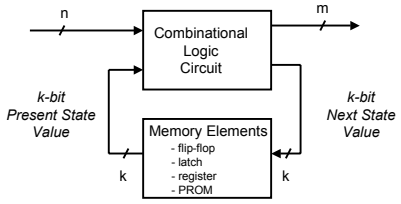
Sequential Systems Review

- 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

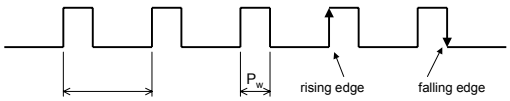
Sequential System Diagram



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

Slide by Prof Mitch Thorton

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 %)

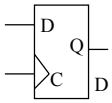
duty cycle = P_w / τ

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

Slide by Prof Mitch Thorton

Memory Elements

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

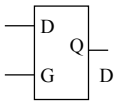


D flip flop (DFF)

D	Q(t+1)
0	0
1	1

Q(t+1) is Q next state

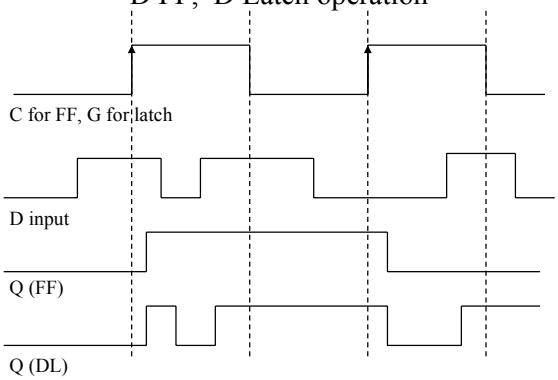
Flip-flops are edge triggered (either rising or falling edge).



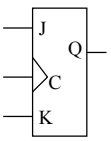
D latch (DL)

Latches are level sensitive. Q follows D when G=1, latches when G goes from 1 to 0.

D FF, D Latch operation

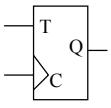


Other State Elements



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

JK useful for single bit flags with separate set(J), reset(K) control.



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

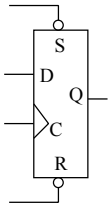
Useful for counter design.

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
- We will use DFFs almost exclusively in this class
  - Will always used edge-triggered state elements (FFs), not level sensitive elements (latches).

Synchronous vs Asynchronous Inputs

Synchronous input: Output will change after active clock edge
Asychronous input: Output changes independent of clock

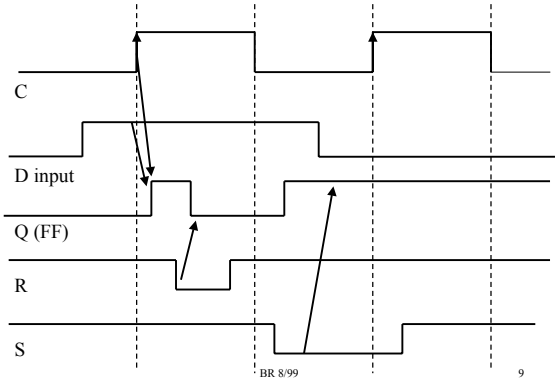


State elements often have async set, reset control.

D input is synchronous with respect to Clk

S, R are asynchronous. Q output affected by S, R independent of C. Async inputs are dominant over Clk.

D FF with async control



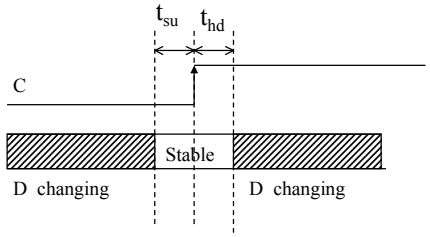
FF Timing

- 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, 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.

Setup, Hold Time

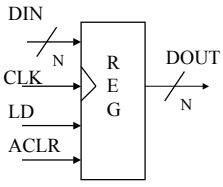


If changes on D input violate either setup or hold time, then correct FF operation is not guaranteed.

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.

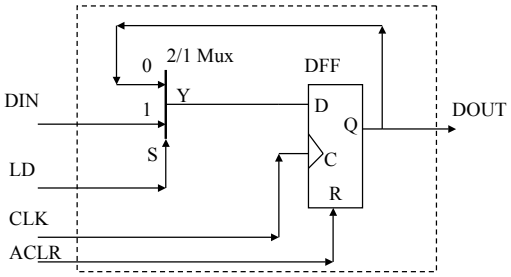


Register contents do not change unless LD = 1 on active edge of clock.

A DFF is NOT a register! DFF contents change every clock edge.

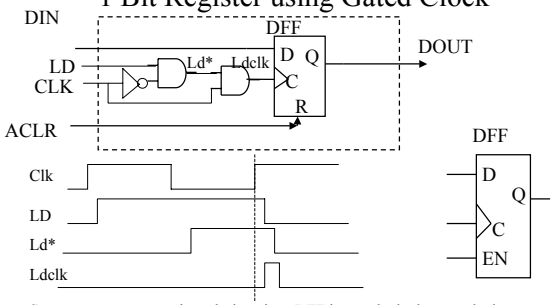
ACLR used to asynchronously clear the register

1 Bit Register using DFF, Mux



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

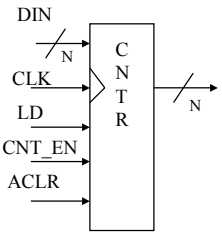
1 Bit Register using Gated Clock



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.

Counter

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

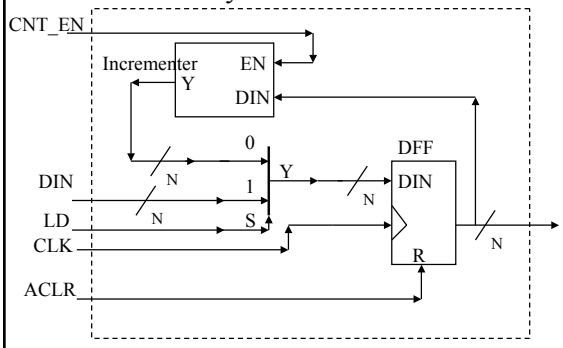


LD asserted loads counter with DIN value.

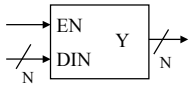
CNT_EN asserted will increment counter on next active clock edge.

ACLR will asynchronously clear the counter.

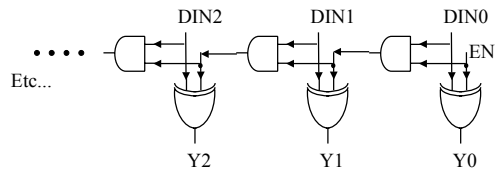
One way to build a Counter



Incrementer: Combinational Building Block



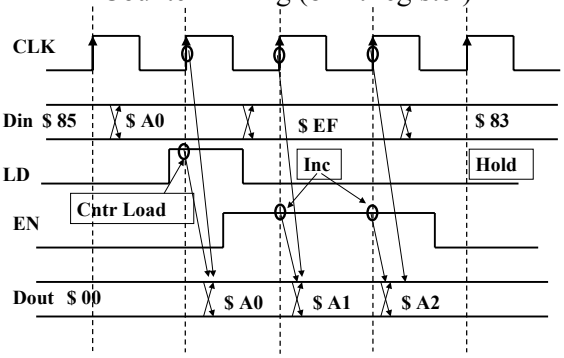
When EN=1, Y = DIN + 1
When EN=0, Y = DIN



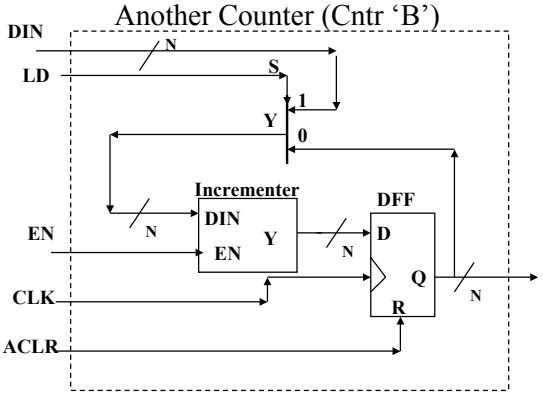
Counter Operation

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 (Cntr 'B')



Counter Operation

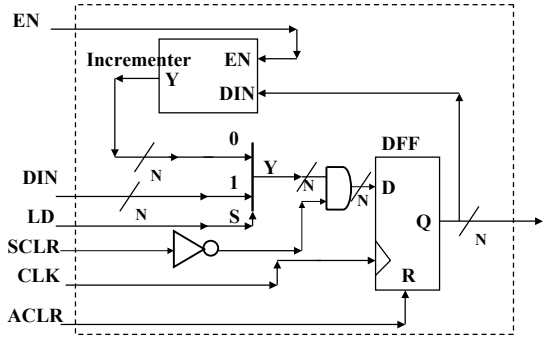
Counter B						
Aclr	Clk	En	LD	Q	Q+	Op
H	X	X	X	X	0	Async Clr
L	↑	L	H	X	Din	Load
L	↑	H	L	Q	Q+1	Increment
L	X	L	L	Q	Q	Hold
L	↑	H	H	Q	Din+1	Load Inc

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.

Cntr 'A' with SCLR Input

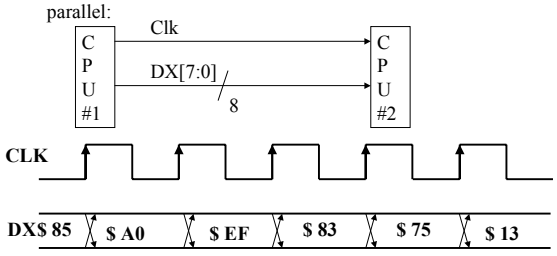


Counter Operation

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

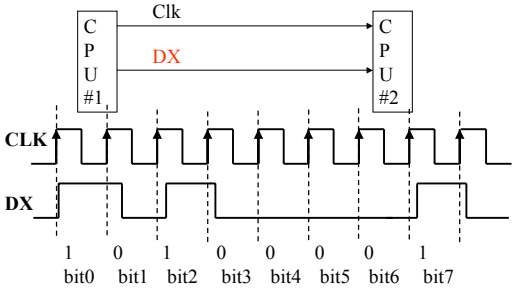
To transfer data between two computers, we can do it in parallel:



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.



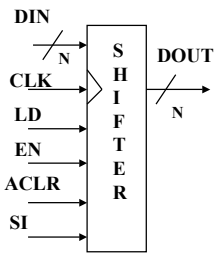
$ 85 = 10000101, data transmitted LSB to MSB

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 Register

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.

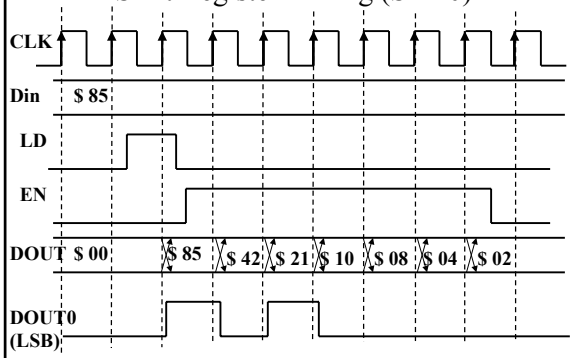
EN asserted will shift data on next active clock edge.

ACLR is async clear.

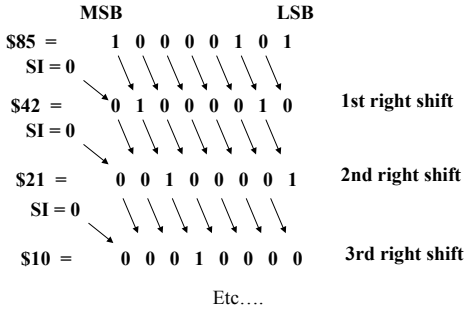
SI is serial data in.

Look at LSB of DOUT for serial data out.

Shift Register Timing (SI = 0)

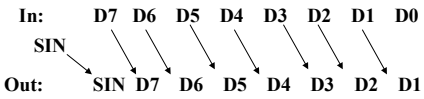


Understanding the shift operation

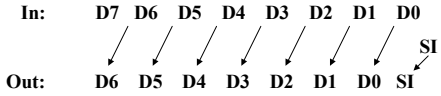


Right Shift vs Left Shift

A right shift is MSB to LSB

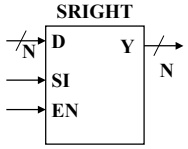


A left shift is LSB to MSB



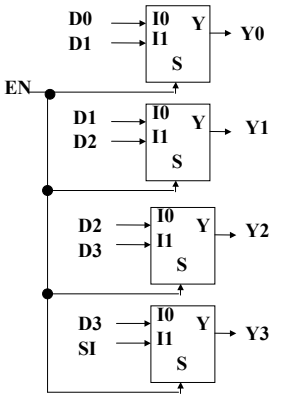
Combinational Right Shifter

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



4-bit Combinational RIGHT Shifter Implementation

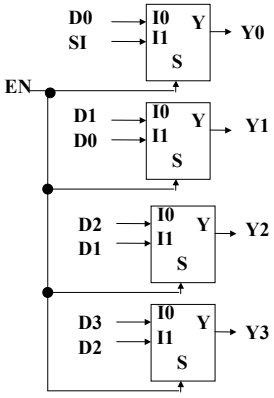
When EN = 0, then:

Y = D3 D2 D1 D0

When EN = 1, then:

Y = SI D3 D2 D1

(right shifted by one position)



4-bit Combinational LEFT Shifter Implementation

When EN = 0, then:

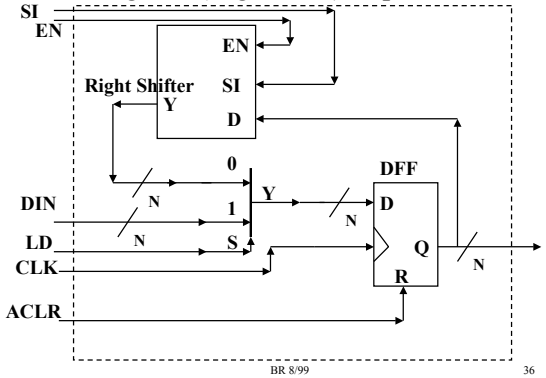
Y = D3 D2 D1 D0

When EN = 1, then:

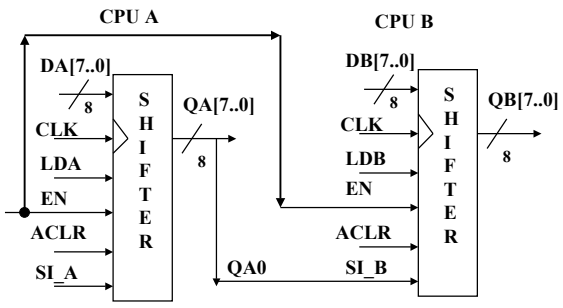
Y = D2 D1 D0 SI

(left shifted by one position)

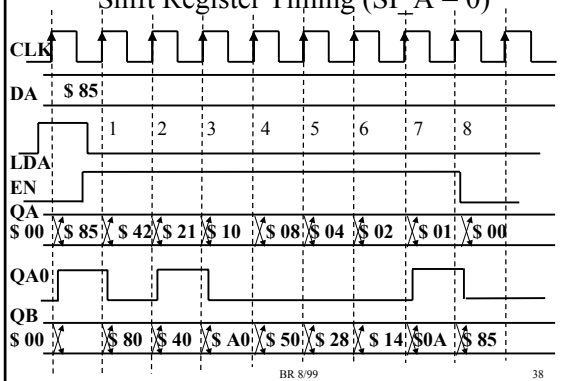
Shift Register (Right shift) Implementation



Serial Communication



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.

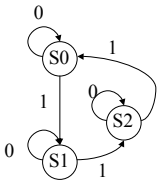
Sequential System Description

- The Q outputs of the flip-flops form a state vector
- A particular set of outputs is the Present State (PS)
- 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 as FSMs yet a register is a sequential circuit.

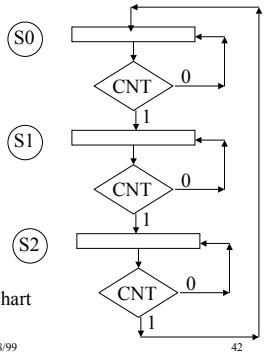
Describing FSMs

- State Tables
- State Equations
- State Diagrams
- Algorithmic State Machine (ASM) Charts
  - Preferred method in this class
- HDL descriptions

Example State Machine



State Diagram
(Bubble Diagram)



ASM Chart

State Assignment

State assignment is the binary coding used to represent the states

Given N states, need at least $\log_2(N)$ FFs to encode the states

(i.e. 3 states, need at least 2 FFs for state information).

S0 = 00, S1 = 01, S2 = 10 (FSM is now a modulo 3 counter)

Do not always have to use the fewest possible number of FFs.
A common encoding is One-Hot encoding - use one FF per state.

S0 = 001, S1 = 010, S2 = 100

State assignment affects speed, gate count of FSM

FSM Implementation

Use DFFs, State assignment: S0 = 00, S1 = 01, S2 = 10

	PS		NS			
Inc	Q1	Q0	Q1+	Q0	D1	D0
0	0	0	0	0	0	0
0	0	1	0	1	0	1
0	1	0	1	0	1	0
0	1	1	x	x	x	x
1	0	0	0	1	0	1
1	0	1	1	0	1	0
1	1	0	0	0	0	0
1	1	1	x	x	x	x

State Table

Equations

D1 = Inc'Q1Q0' + IncQ1'Q0

D0 = Inc'Q1'Q0 + IncQ1'Q0'

Minimize Equations (if desired)

D1

Inc \ Q1Q0	00	01	11	10
0	0	0	(x	1)
1	0	(1	x)	0

D1 = Inc' Q1 + Inc Q0

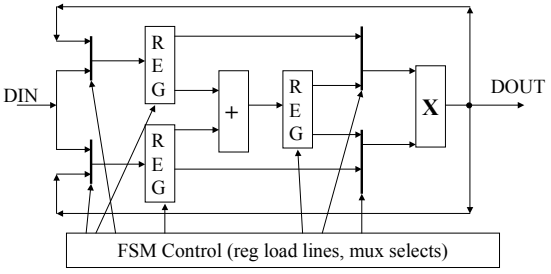
D0

Inc \ Q1Q0	00	01	11	10
0	0	(1	x)	0
1	(1)	0	x	0

D1 = Inc' Q0 + Inc Q1'Q0'

FSM Usage

- Custom counters
- Datapath control



Summary

- We will be describing sequential systems via VHDL and ASM charts
  - Use ASM chart for human reader, VHDL to allow synthesis of the design
  - Synthesis will perform combinational minimization, but not state reduction.
- Will use common sequential building blocks extensively
  - Registers, Counters, Shift registers, Memories
- Basic storage element will be DFF
- Synchronous (edge-triggered) design methodology