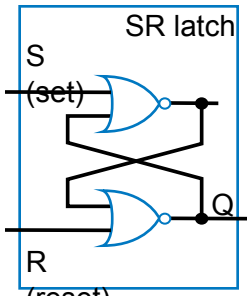
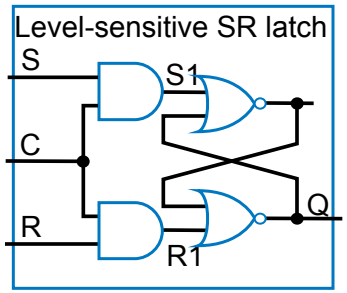
Sequential Logic Design

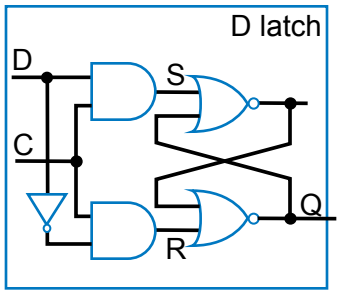
Bit Storage Summary



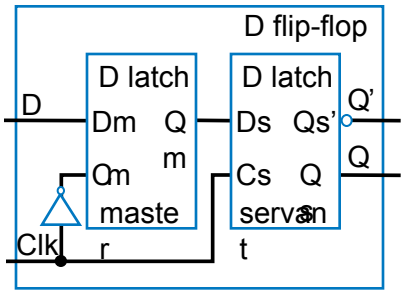
Feature: S=1 sets Q to 1, R=1 resets Q to 0. Problem: SR=11 yield undefined Q.



Feature: S and R only have effect when C=1. We can design outside circuit so SR=11 never happens when C=1. Problem: avoiding SR=11 can be a burden.



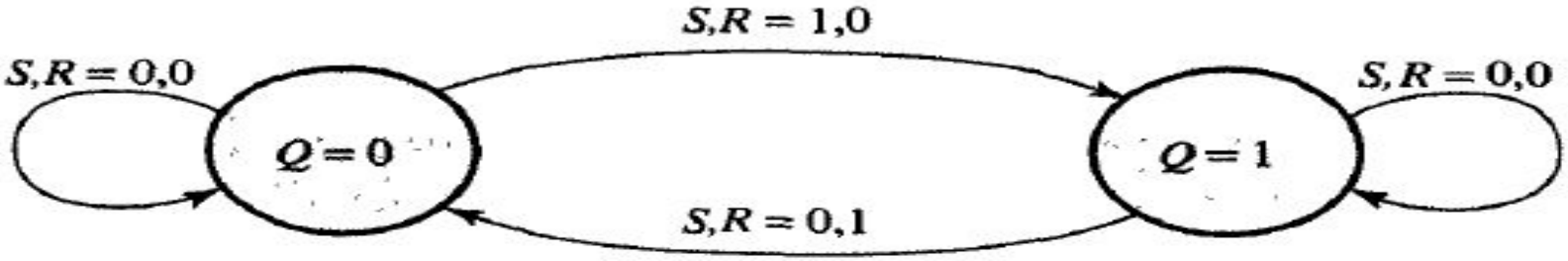
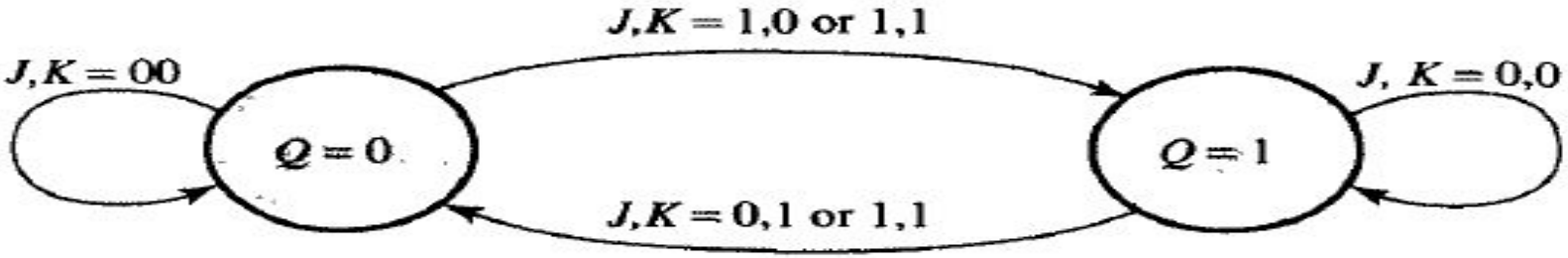
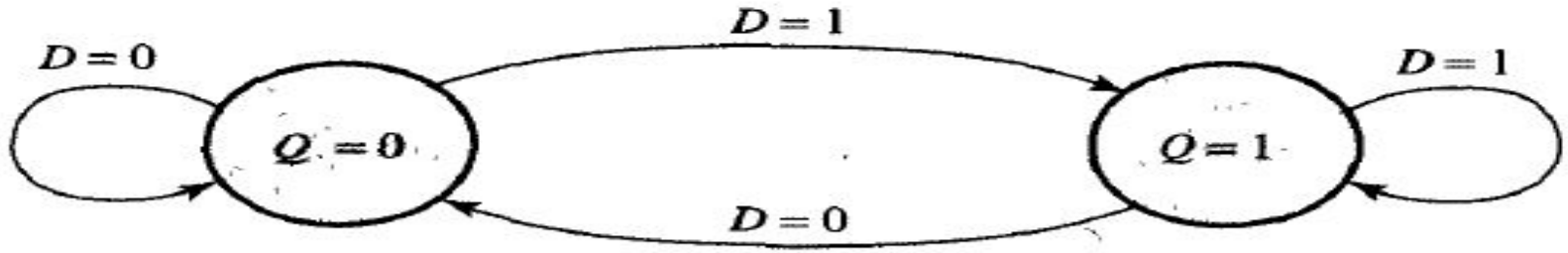
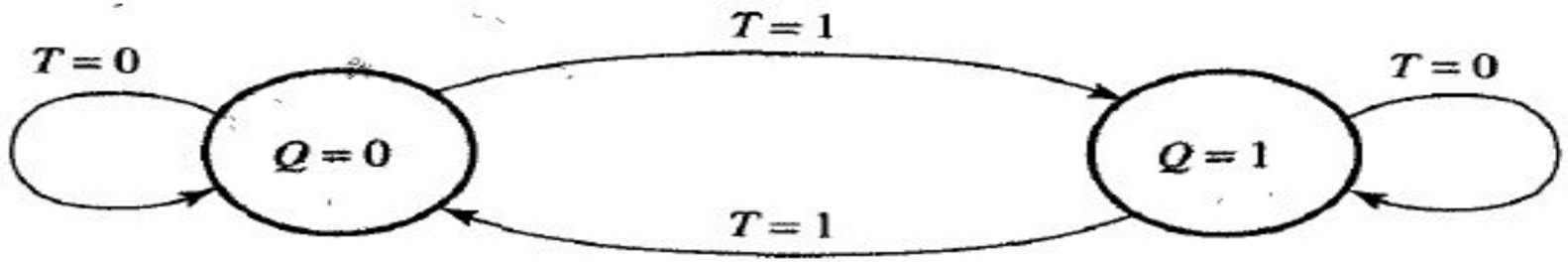
Feature: SR can't be 11 if D is stable before and while C=1, and will be 11 for only a brief glitch even if D changes while C=1. Problem: C=1 too long propagates new values through too many latches: too short may not enable a store.

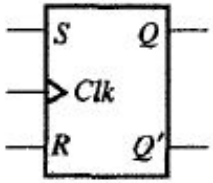
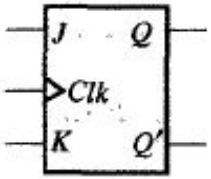
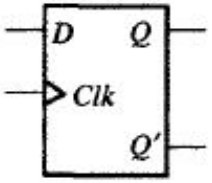
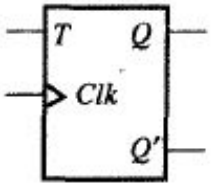


Feature: Only loads D value present at rising clock edge, so values can't propagate to other flip-flops during same clock cycle. Tradeoff: uses more gates internally than D latch, and requires more external gates than SR – but gate count is less of an issue today.

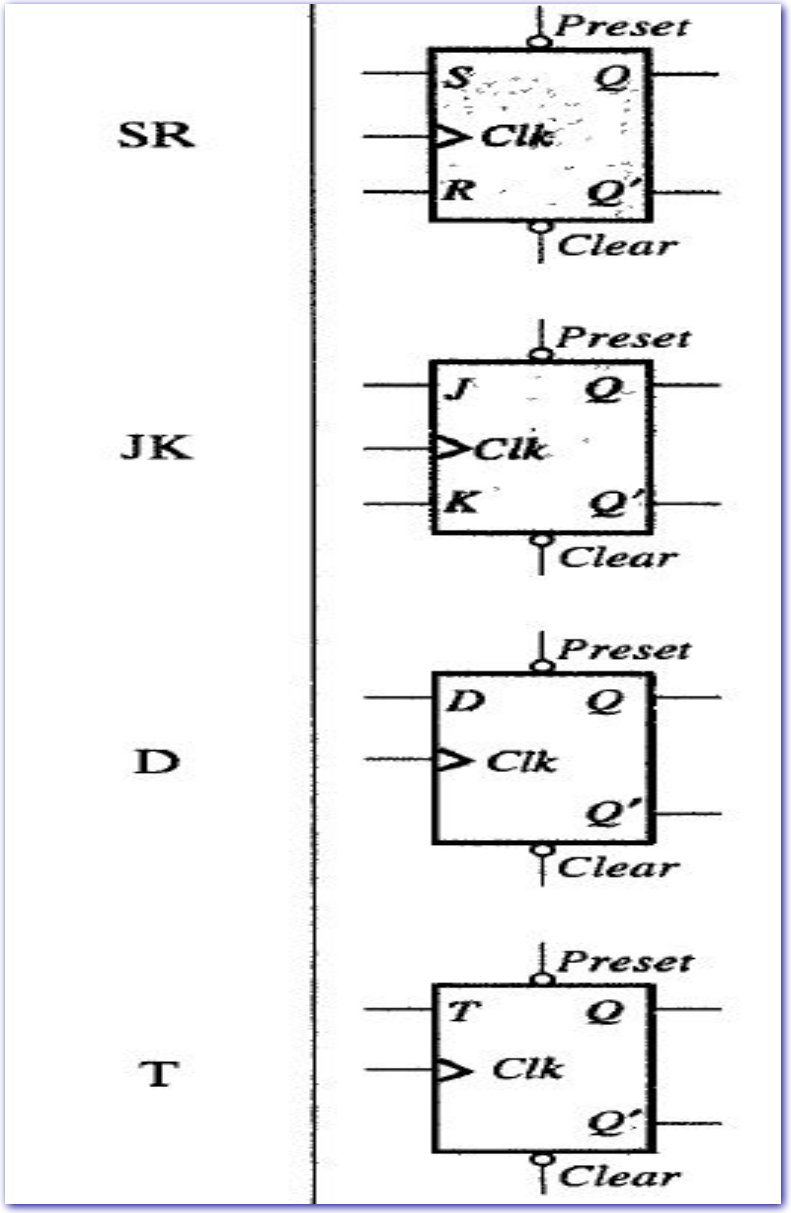
- We considered increasingly better bit storage until we arrived at the robust D flip-flop bit storage

State Diagrams for Various Flip-Flops

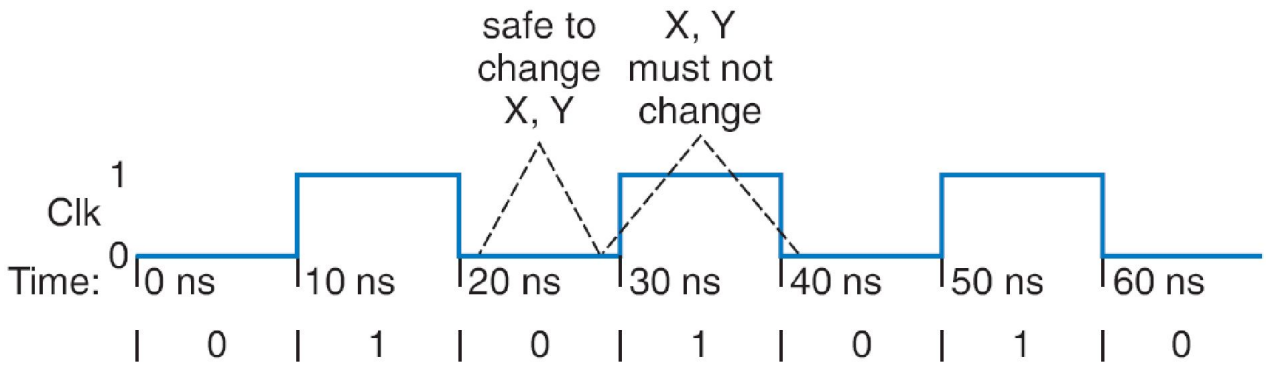
NAME	STATE DIAGRAM
SR	
JK	
D	
T	

FLIP-FLOP NAME	FLIP-FLOP SYMBOL	CHARACTERISTIC TABLE	CHARACTERISTIC EQUATION	EXCITATION TABLE
SR		S R $Q(next)$ 0 0 Q 0 1 0 1 0 1 1 1 NA	$Q(next) = S + R'Q$ $SR = 0$	Q $Q(next)$ S R 0 0 0 X 0 1 1 0 1 0 0 1 1 1 X 0
JK		J K $Q(next)$ 0 0 Q 0 1 0 1 0 1 1 1 Q′	$Q(next) = JQ' + K'Q$	Q $Q(next)$ J K 0 0 0 X 0 1 1 X 1 0 X 1 1 1 X 0
D		D $Q(next)$ 0 0 1 1	$Q(next) = D$	Q $Q(next)$ D 0 0 0 0 1 1 1 0 0 1 1 1
T		T $Q(next)$ 0 Q 1 Q′	$Q(next) = TQ' + T'Q$	Q $Q(next)$ T 0 0 0 0 1 1 1 0 1 1 1 0

Preset and Clear



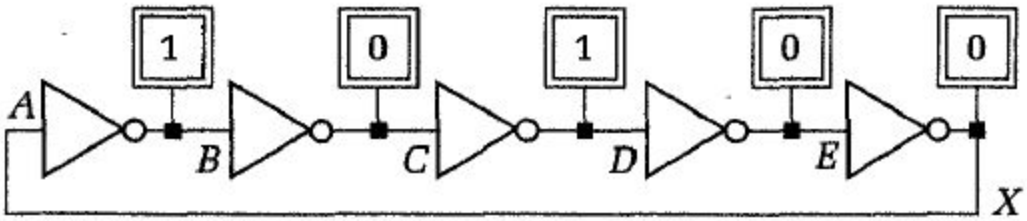
Clocks



- ***Clock period***: time interval between pulses
  - Above signal: period = 20 ns
- ***Clock cycle***: one such time interval
  - Above signal shows 3.5 clock cycles
- ***Clock frequency***: 1/period
  - Above signal: frequency = 1 / 20 ns = 50 MHz
    - 1 Hz = 1/s

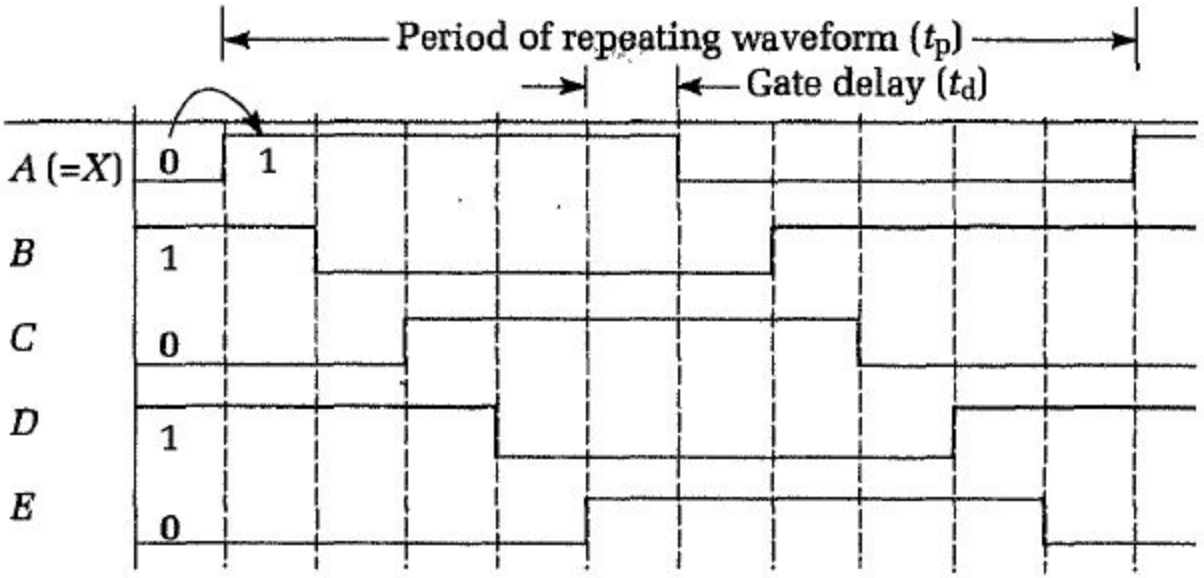
Freq	Period
100 GHz	0.01 ns
10 GHz	0.1 ns
1 GHz	1 ns
100 MHz	10 ns
10 MHz	100 ns

Clock Generation-Ring Oscillator



(a) Ring oscillator

1ns delay for inverte



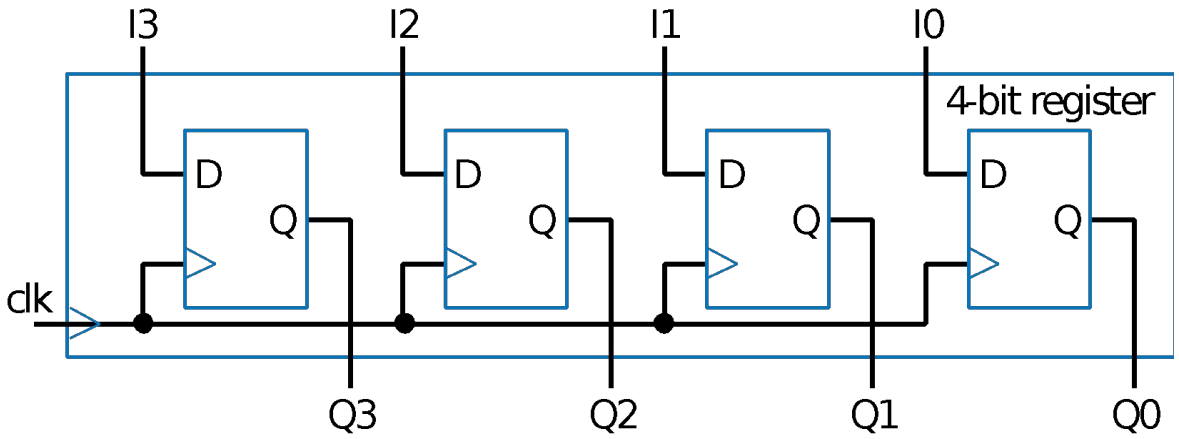
(b) Timing waveform

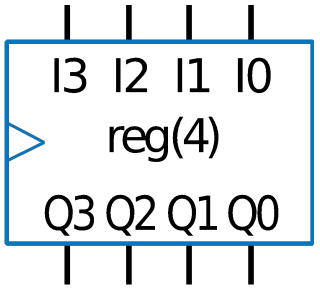
Frequency=1/2*n*td

Frequency=1/2*5*1n= 100Mhz

Basic Register

- Typically, we store multi-bit items
  - e.g., storing a 4-bit binary number
- ***Register***: multiple flip-flops sharing clock signal
  - From this point, we'll use registers for bit storage
    - No need to think of latches or flip-flops
    - But now you know what's inside a register



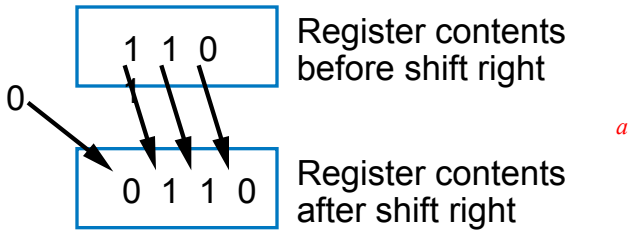


Shift Registers

- Shift registers
- Parallel/serial format conversion
  - SISO (serial in serial out)
  - SIPO (serial in parallel out)
  - PISO (parallel in serial out)
  - PIPO (parallel in parallel out)

Shift Register

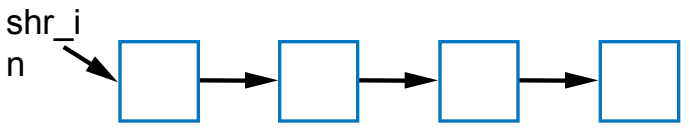
- Shift right
  - Move each bit one position right
  - Shift in 0 to leftmost bit



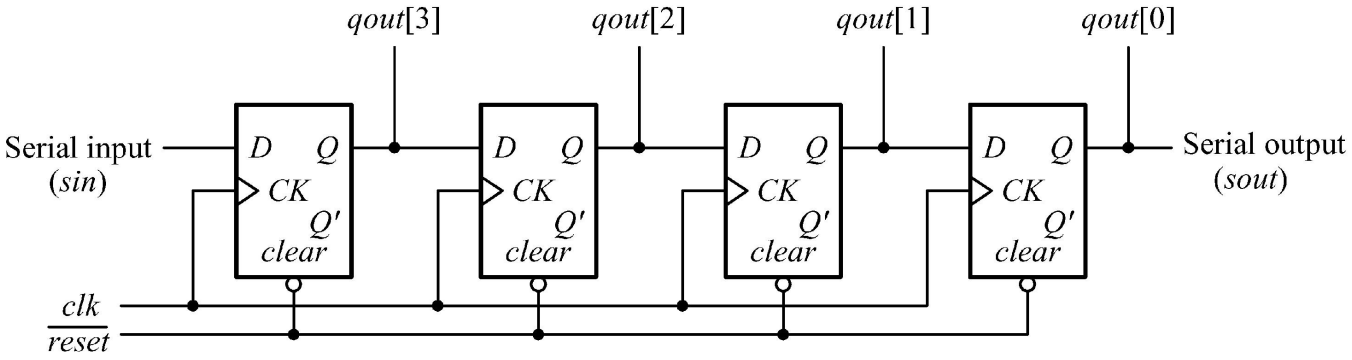
Q: Do four right shifts on 1001, showing value after each shift

A: 1001 (original)
0100
0010
0001
0000

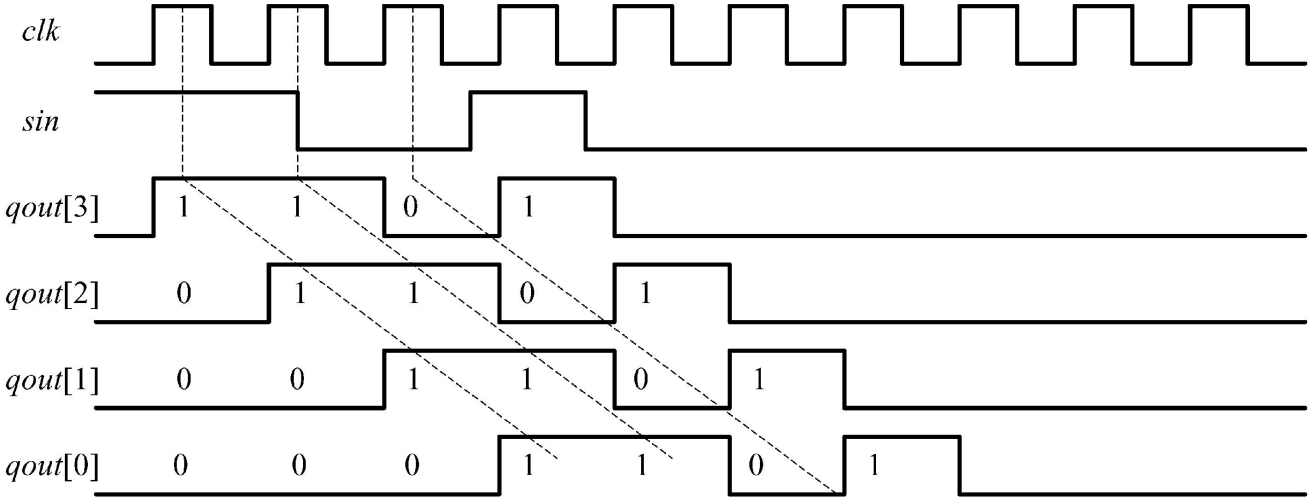
- Implementation: Connect flip-flop output to next flip-flop's input



Shift Registers



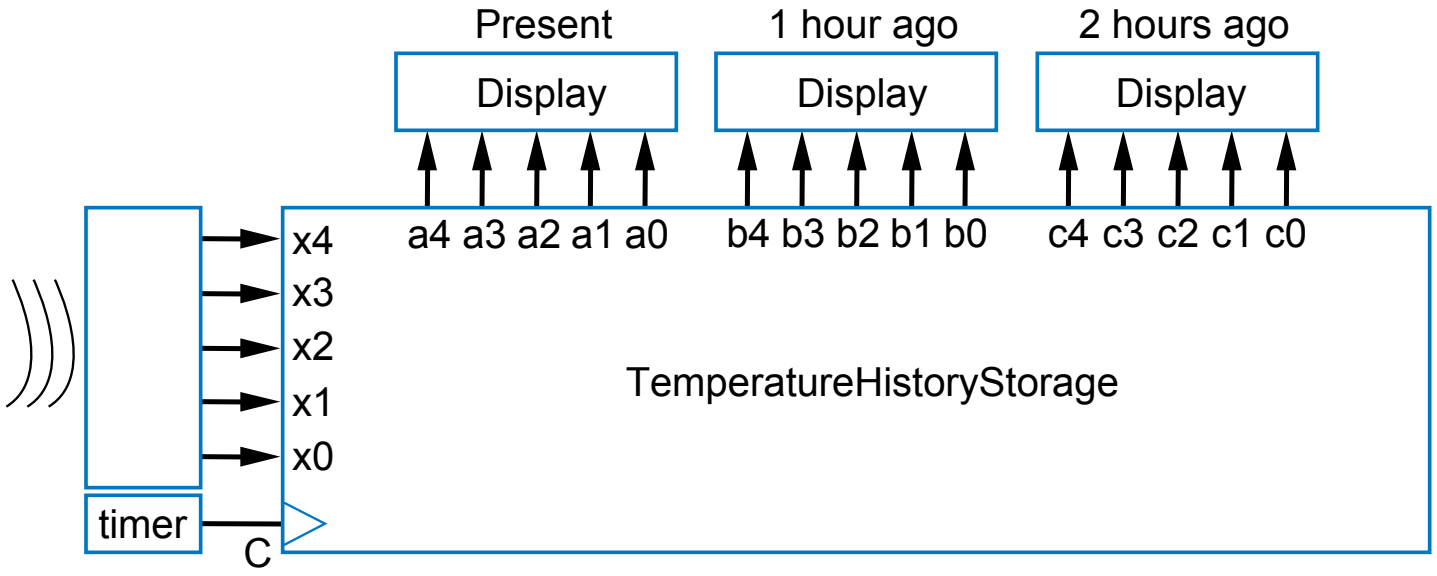
(a) Logic circuit



(b) Timing

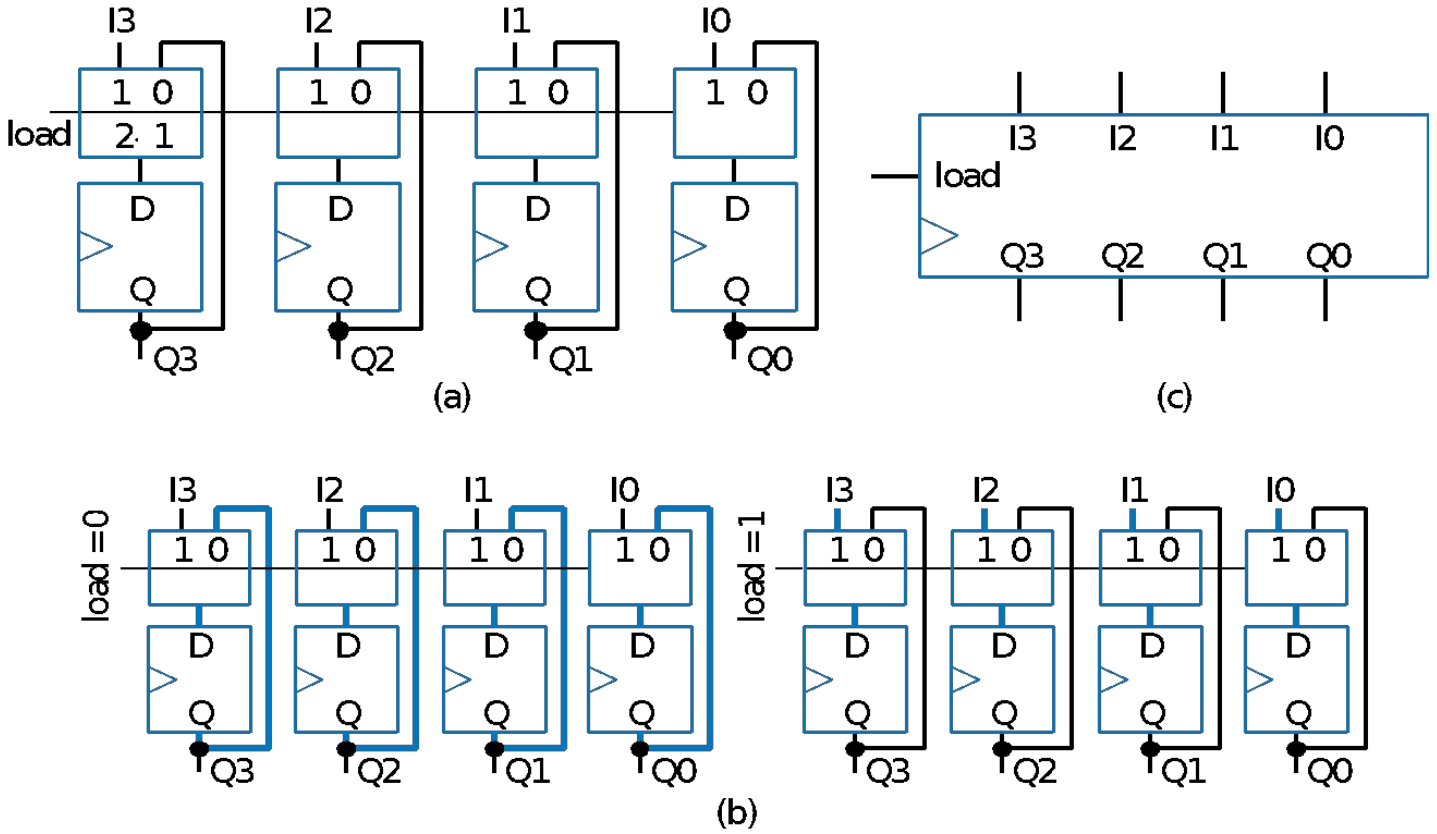
Example Using Registers: Temperature Display

- Temperature history display
  - Sensor outputs temperature as 5-bit binary number
  - Timer pulses C every hour
  - Record temperature on each pulse, display last three recorded values



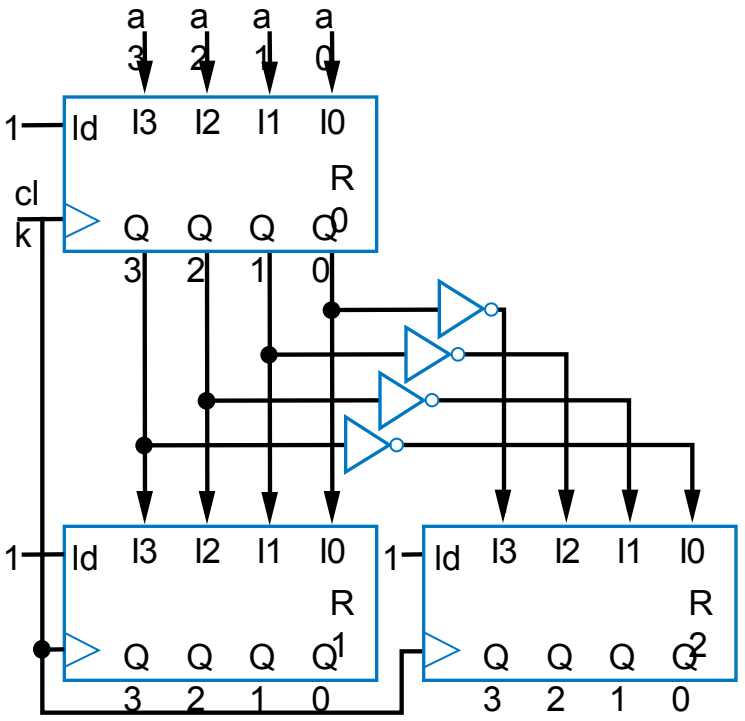
Register with Parallel Load

- Add 2x1 mux to front of each flip-flop
- Register's load input selects mux input to pass
  - Either existing flip-flop value, or new value to load

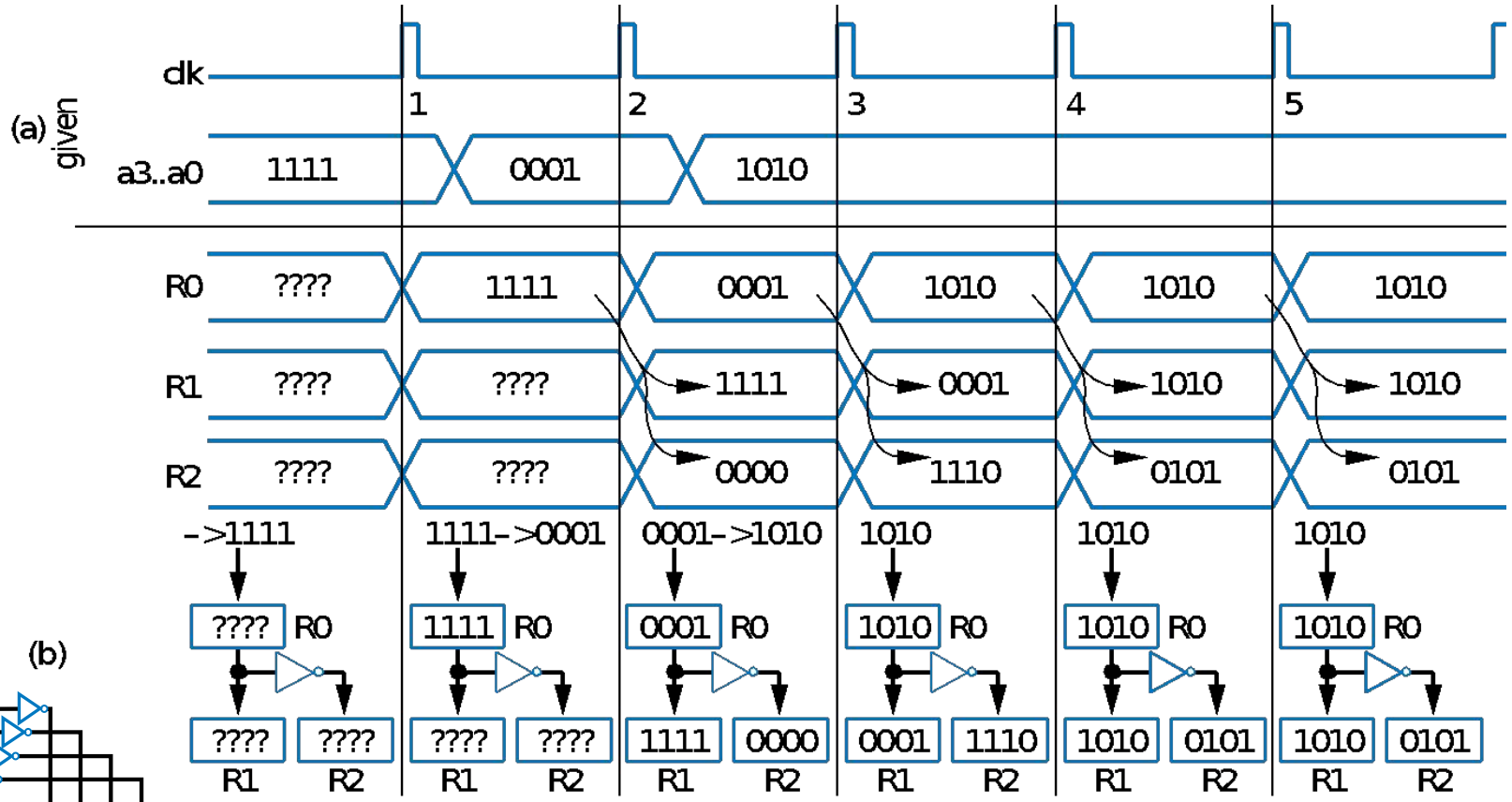


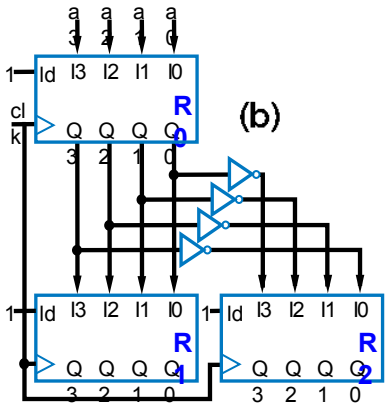
Basic Example Using Registers

- This example will show how registers load simultaneously on clock cycles
  - Notice that all load inputs set to 1 in this example -- just for demonstration purposes



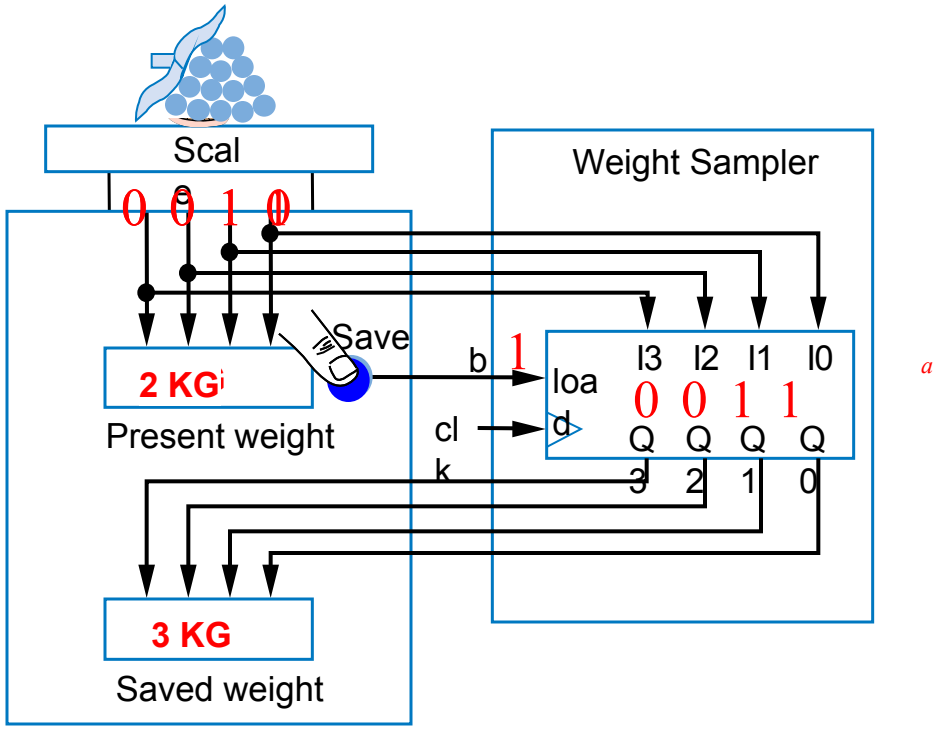
Basic Example Using Registers





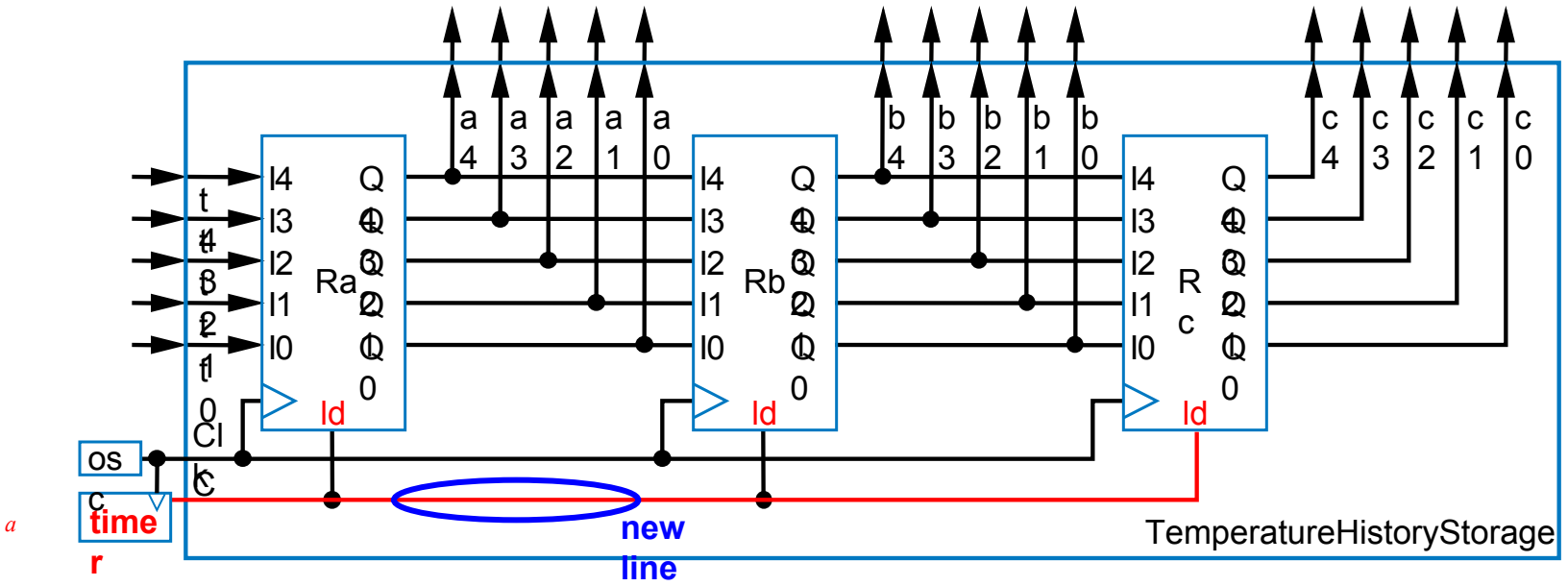
Register Example using the Load Input: Weight Sampler

- Scale has two displays
  - Present weight
  - Saved weight
  - Useful to compare present item with previous item
- Use register to store weight
  - Pressing button causes present weight to be stored in register
    - Register contents always displayed as “Saved weight,” even when new present weight appears



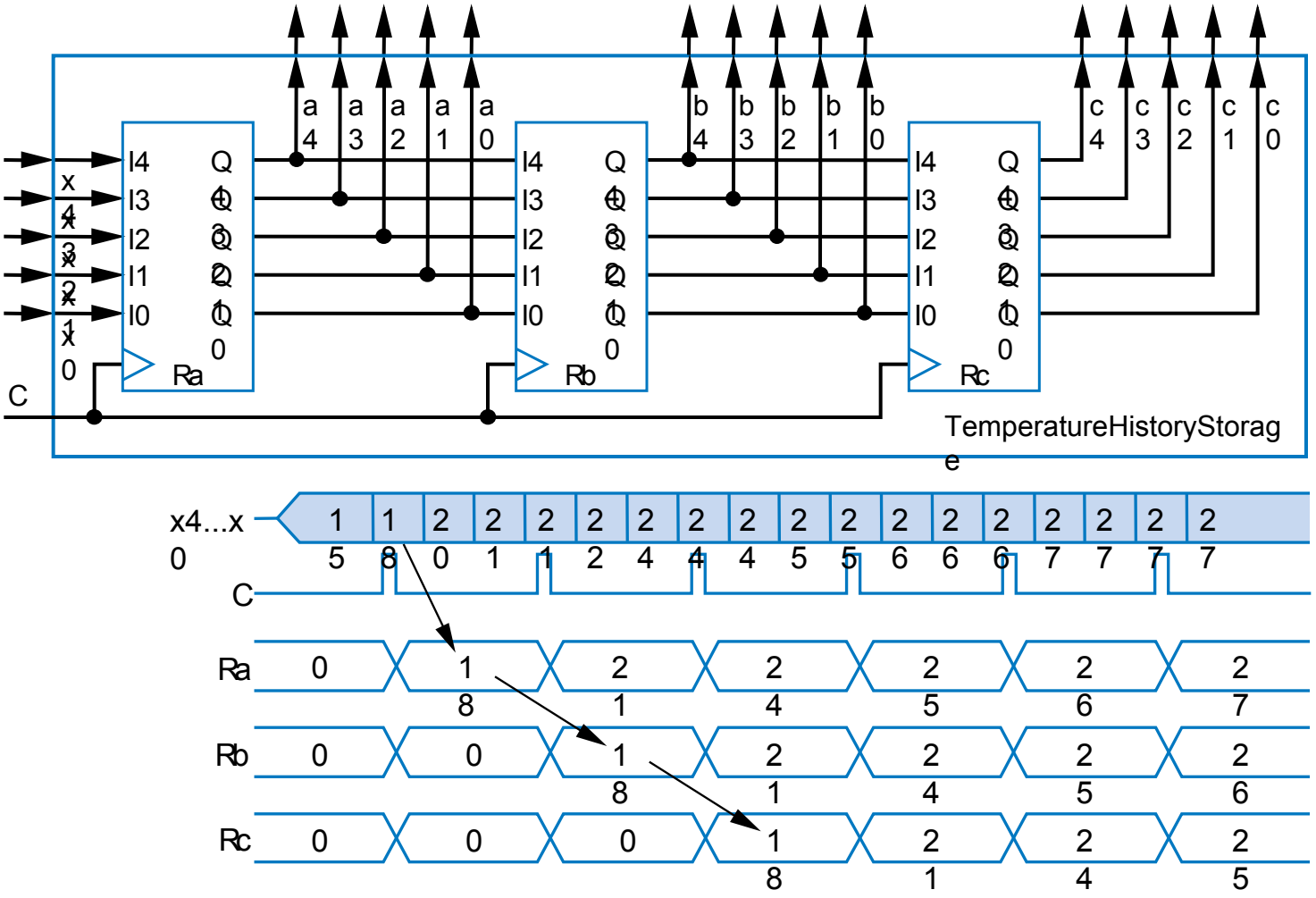
Register Example: Temperature History Display

- example
  - Timer pulse every hour
  - Previously used as clock. Better design only connects oscillator to clock inputs -- use registers with load input, connect to timer pulse.

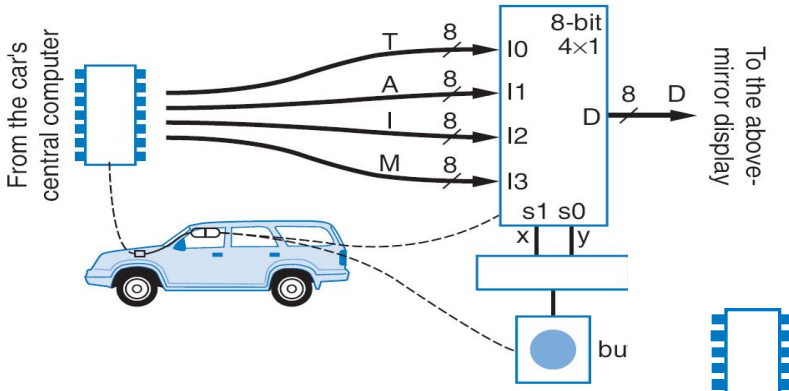


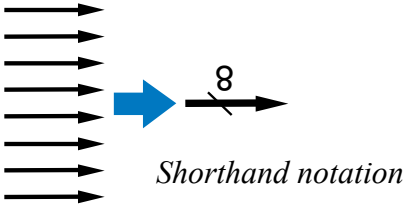
Example Using Registers: Temperature Display

- Use three 5-bit registers

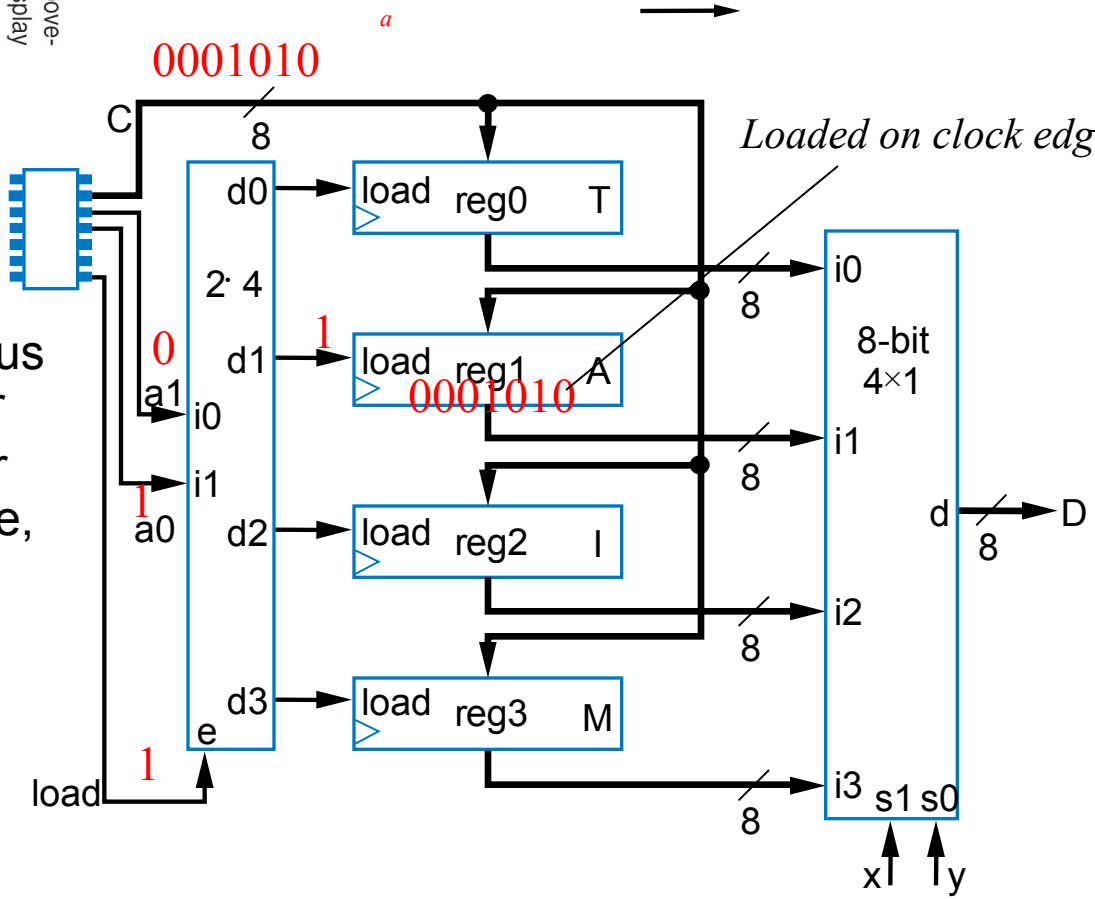


Register Example: Above-Mirror Display



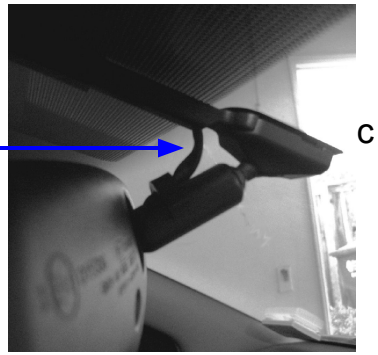


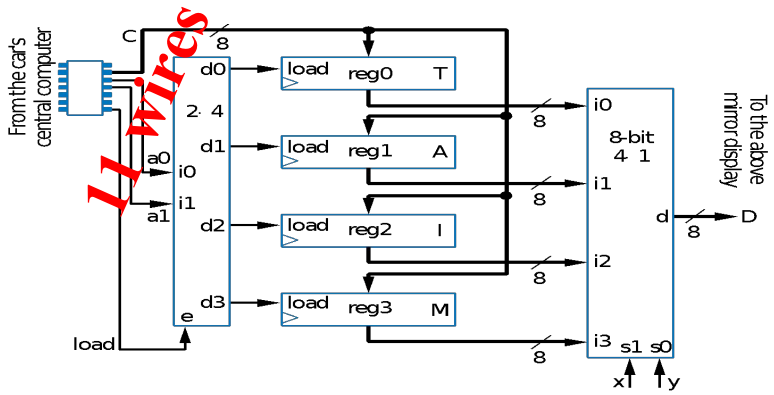
- example: Four simultaneous values from car's computer
- To reduce wires: Computer writes only 1 value at a time, loads into one of four registers
  - Was: 8+8+8+8 = 32 wires
  - Now: 8 +2+1 = 11 wires

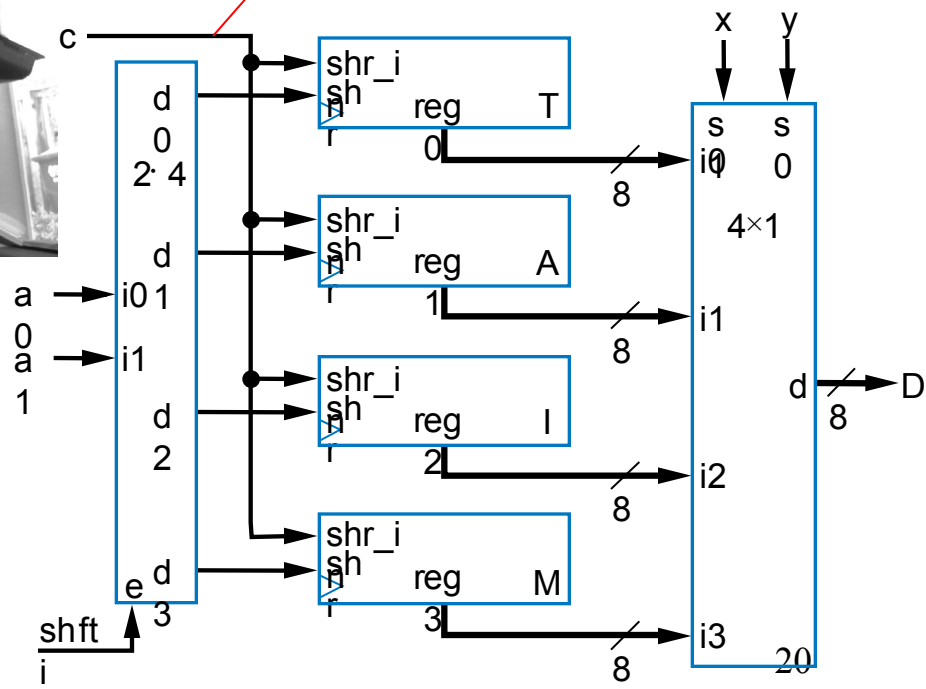


Shift Register Example: Above-Mirror Display

- Earlier example: 8 +2+1 = 11wires from car's computer to above-mirror display's four registers
  - Better than 32 wires, but 11 still a lot -- want fewer for smaller wire bundles
- Use shift registers
  - Wires: 1+2+1=4
  - Computer sends one value at a time, one bit per clock cycle

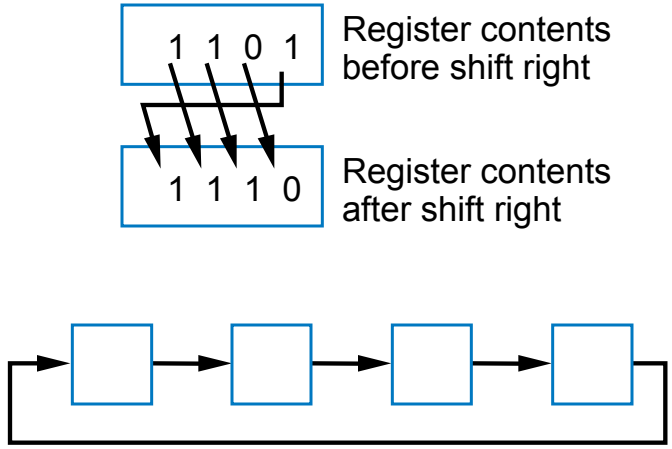






Rotate Register

- Rotate right: Like shift right, but leftmost bit comes from rightmost bit

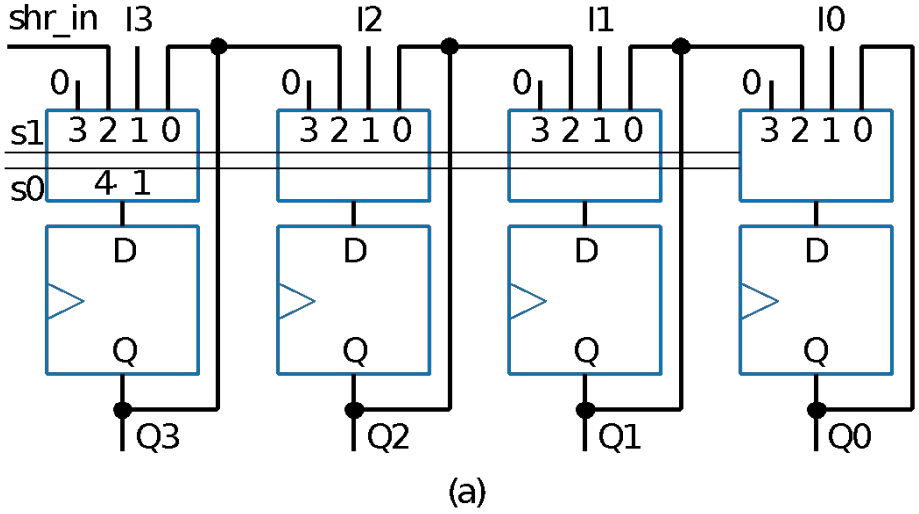


Multifunction Registers

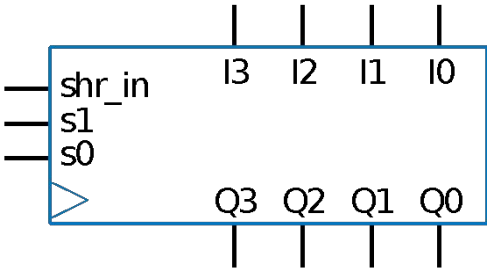
- Many registers have multiple functions
  - Load, shift, clear (load all 0s)
  - And retain present value, of course
- Easily designed using muxes
  - Just connect each mux input to achieve desired function

Functions:

s1	s0	Operation
0	0	Maintain present value
0	1	Parallel load
1	0	Shift right
1	1	(unused - let's load 0s)

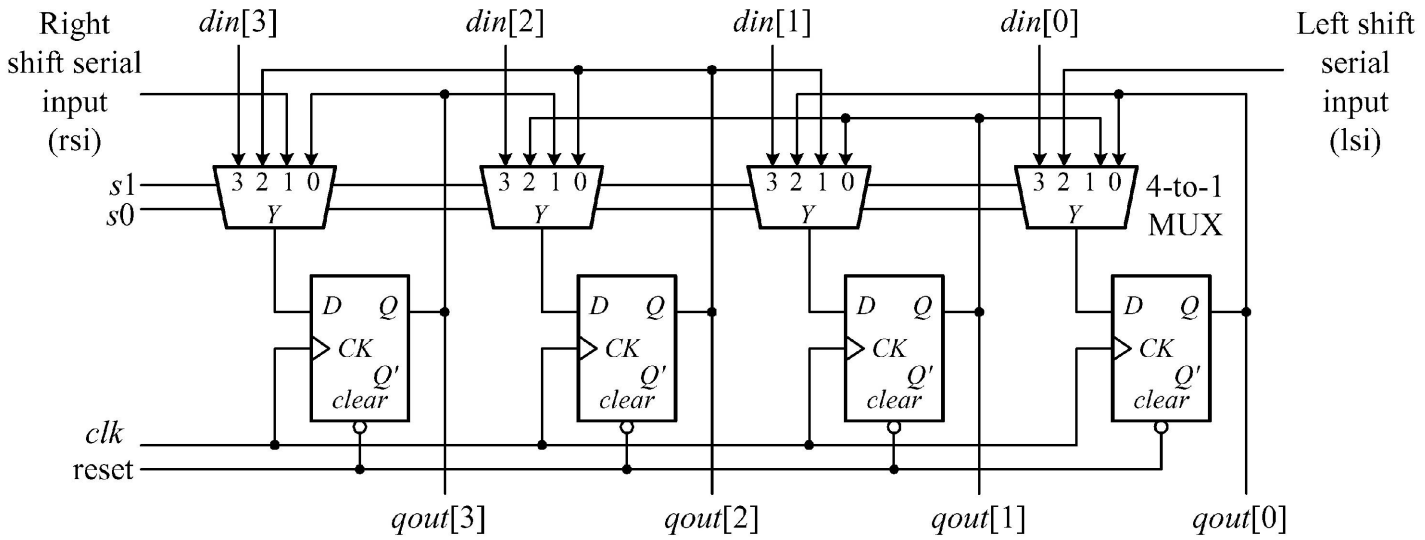


(a)

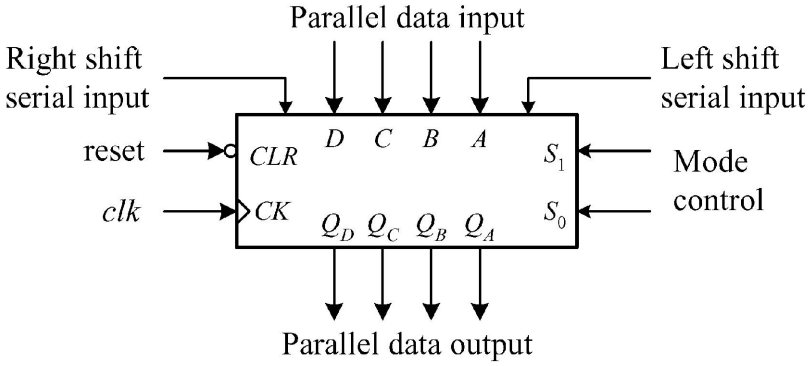


(b)

Universal Shift Registers



(a) Logic diagram



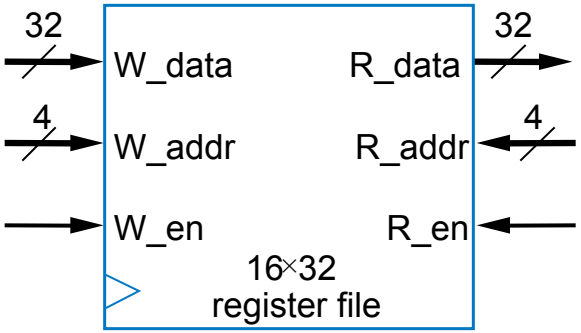
(b) Logic symbol

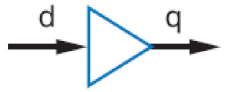
$s1$	$s0$	Function
0	0	No change
0	1	Right shift
1	0	Left shift
1	1	Load data

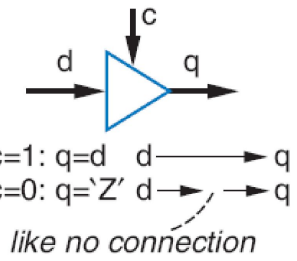
(c) Function table

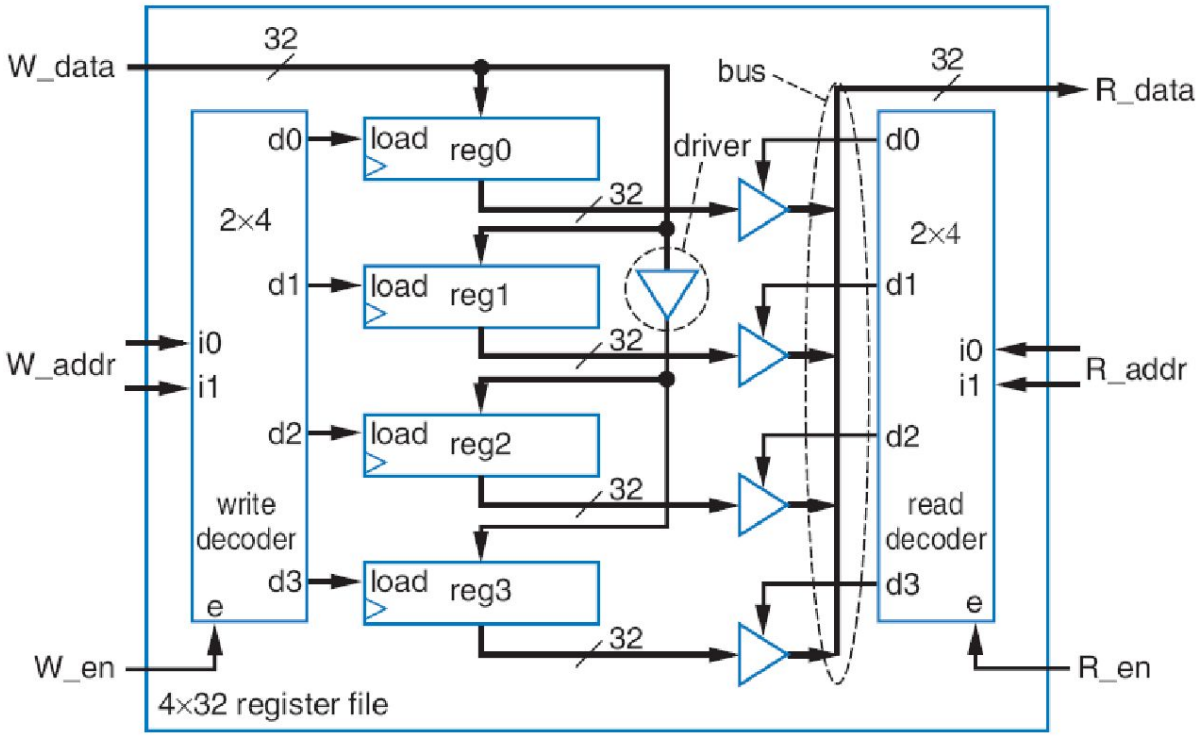
Register File

- Instead, want component that has one data input and one data output, and allows us to specify which internal register to write and which to read









Register File Timing Diagram

- Can write one register and read one register each clock cycle
  - May be same register