

# REGISTERS

Sequential circuit used to store binary word

Consists of a set of flip-flops (each flip-flop stores one bit of information)

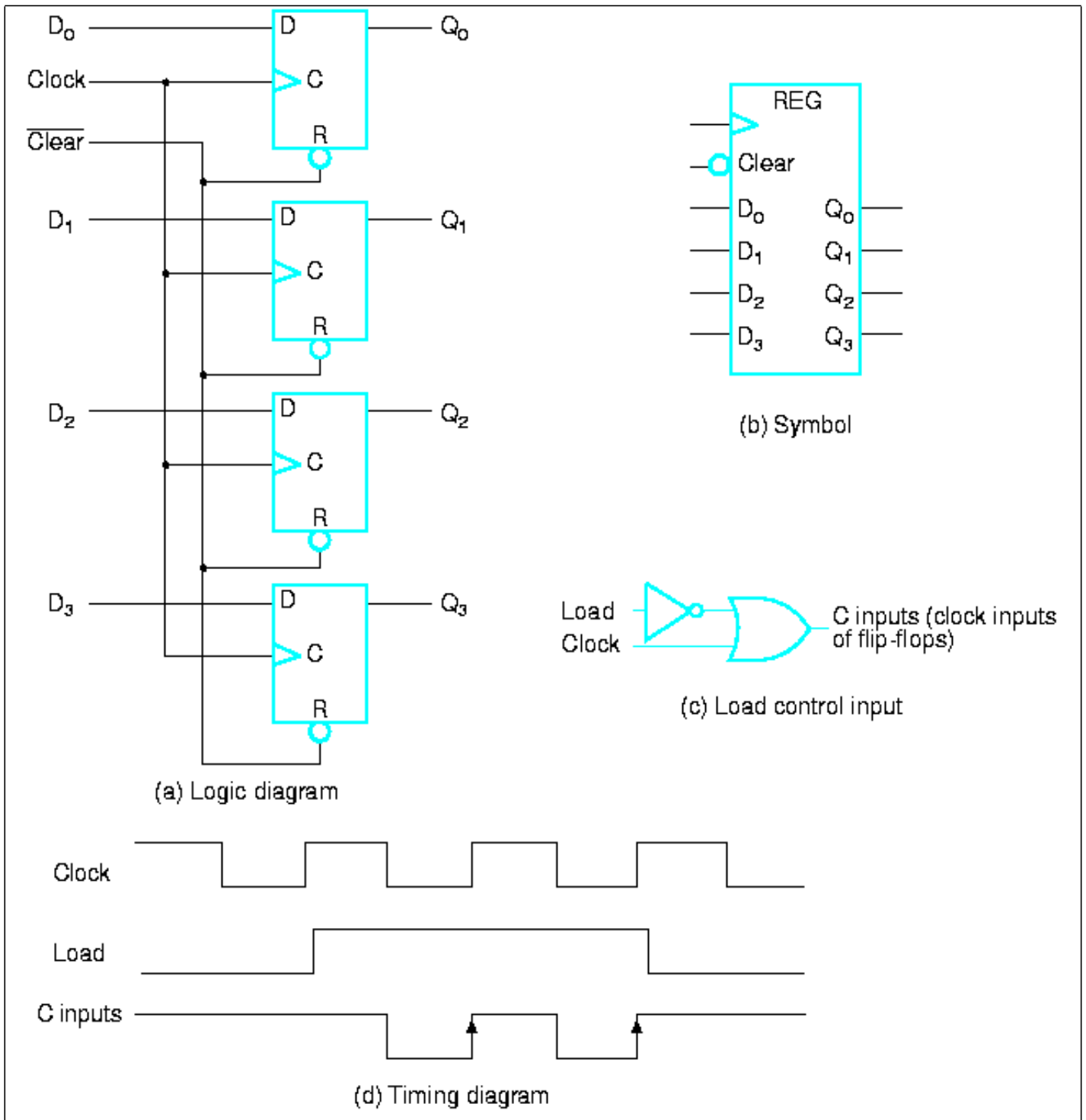
External gates may be used to control the inputs of the flip-flops: that is when and how new binary information is transferred to the flip-flops

All flip-flops of the register are triggered by the same clock pulse

All registers are built by flip-flops that are sensitive to *pulse transition* rather than *pulse duration*

The flip-flops in a register are all of the same type

# Registers (continued)



4-bit Register with *D* Flip-flops

## Loading

Transfer of new binary information into a register

**Serial load:** (transfer in serial mode) Transfer of new binary information into a register bit by bit (# of clock pulses needed to load the register = # of flip-flops in the register)

**Parallel load:** (transfer in parallel mode) Transfer of new binary information into a register as a whole word (# of clock pulses needed to load the register = 1)

Unique master clock supplies clock pulses to all flip-flops and registers in the computer

**Problem:** What to do if we do not want to change the contents of a *master-clocked register*?

**Solution:** Clock gating

$$C \text{ inputs} = \overline{Load} + Clock$$

## Parallel Loading

**Clock gating problem:** Introduces different propagation delays between *Clock* and the inputs of registers (or flip-flops) having and not having clock gating attached

**Solution:** Avoid clock gating.

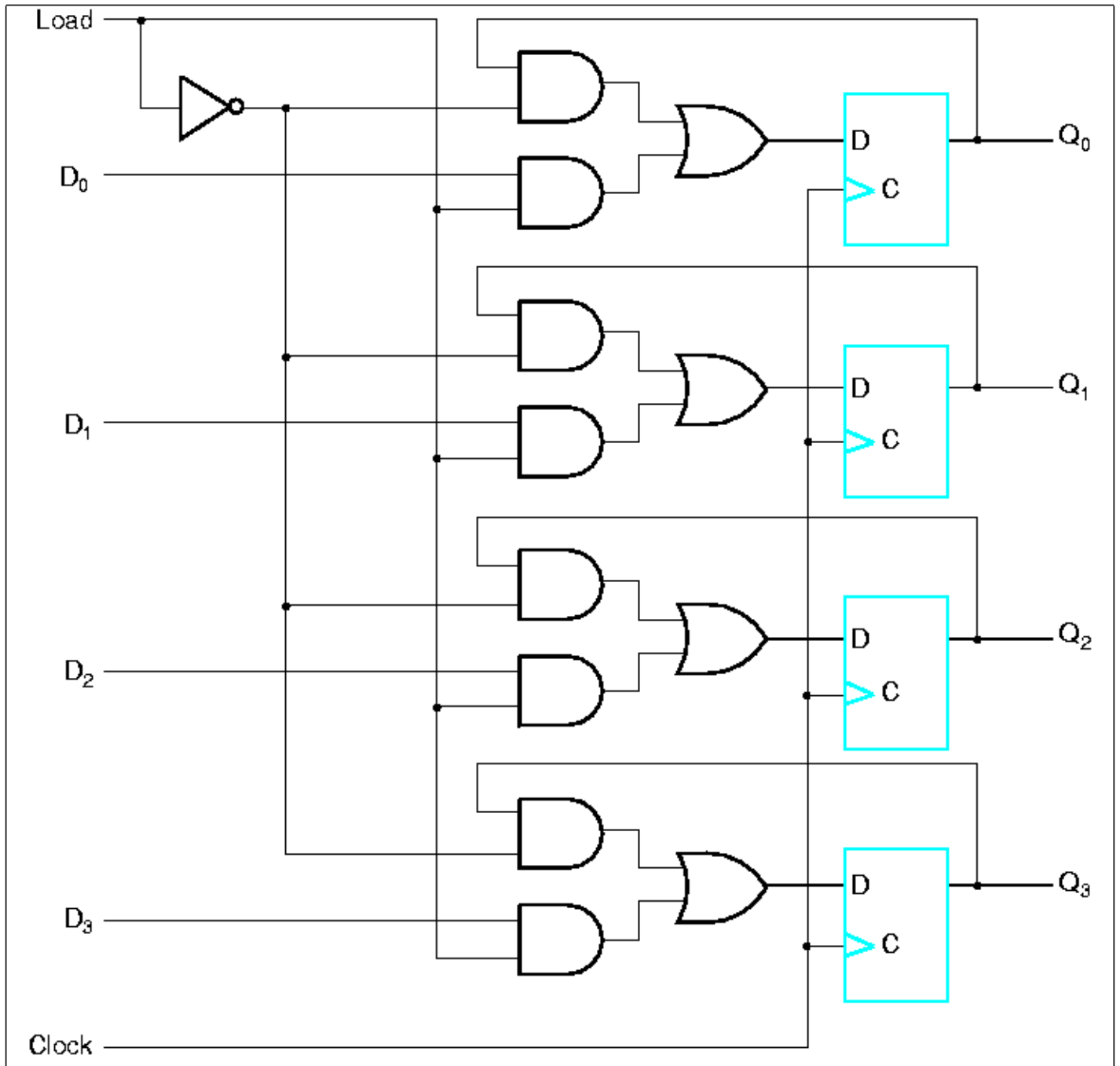
*Load* must determine whether the next clock pulse will accept new information or leave the information in the register intact

If  $Load = 1$  then the data inputs will be transferred into the register with the next positive transition of a clock pulse

If  $Load = 0$  then the data inputs must be blocked and each  $D$  input must be equal to the previous  $Q$  output of the associated flip-flop

Clock pulses are applied continuously to the *Cinputs*. The transfer of information from inputs to register is done simultaneously for all data inputs during a single positive clock pulse

## Parallel Loading (continued)



4-bit  $D$  Register with Parallel Load

# Shift Registers

Register capable of shifting its binary information to its left or right

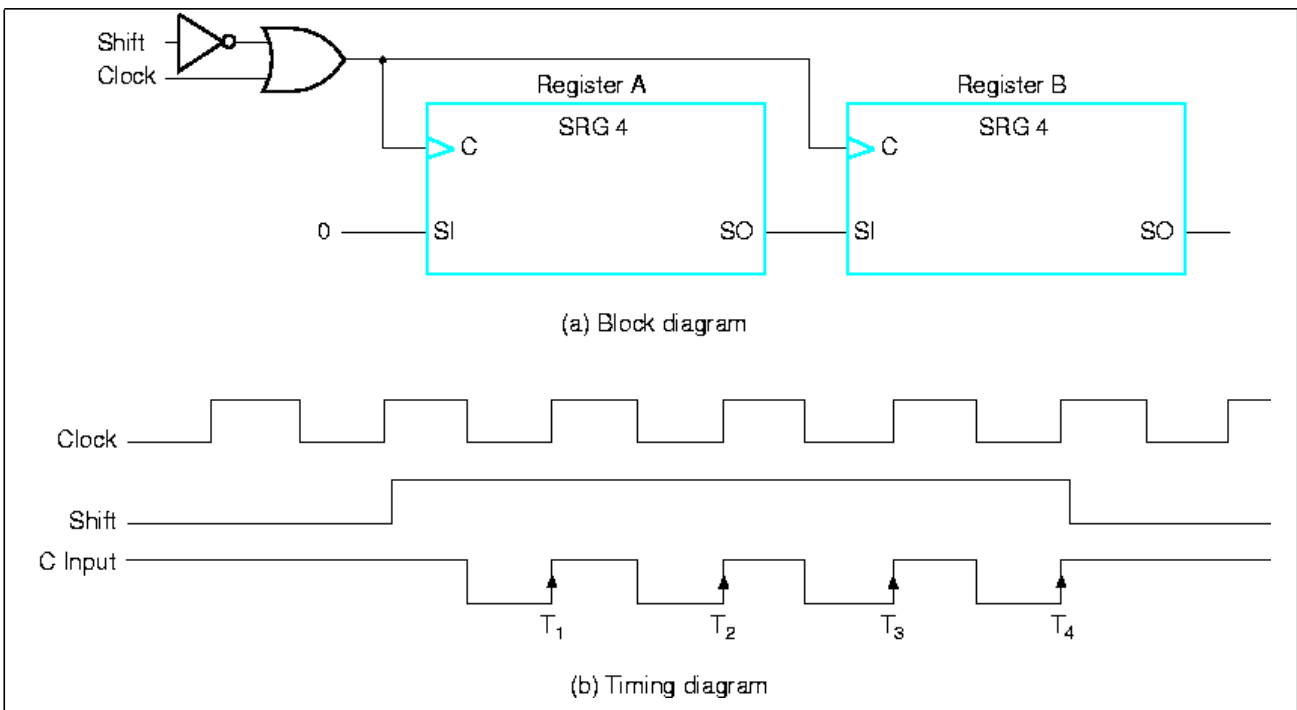
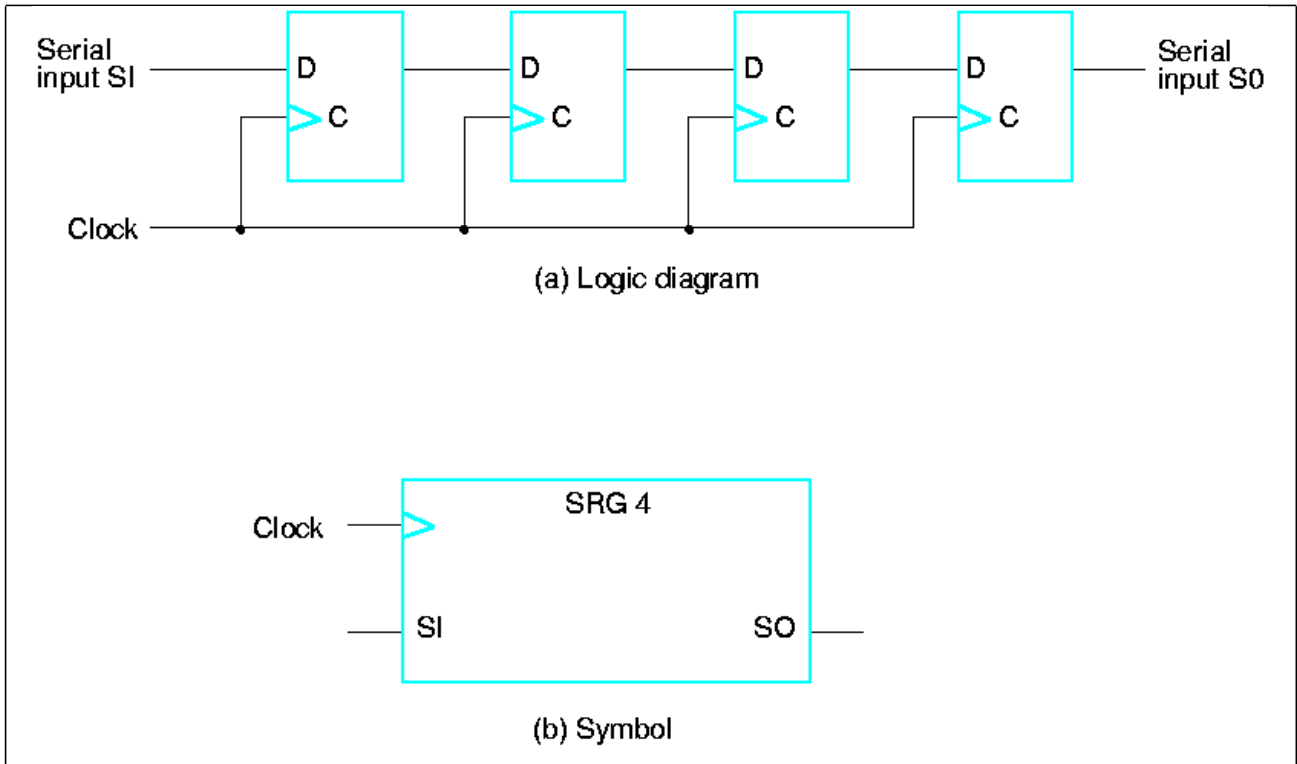
All register's flip-flops are connected in cascade

Used for converting serial data to parallel data, and vice versa

The most general shift register has

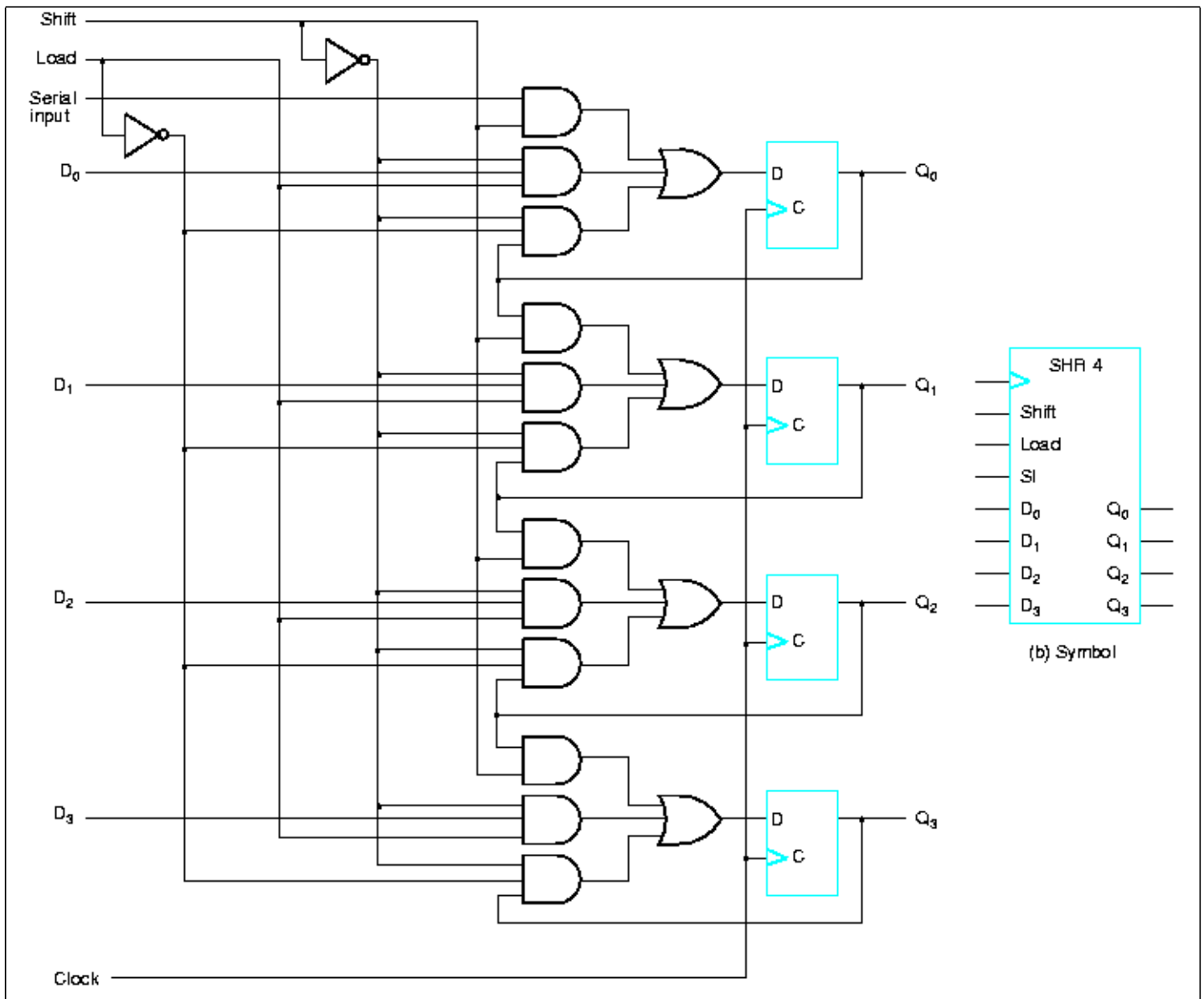
1. a *Clear* control to clear register to 0
2. a *C input* to synchronize all flip-flops
3. a *no change* control
4. a *shift-right* control
5. a *shift-left* control
6. a *parallel-load* control
7.  $n$  parallel input and output lines

# Shift Registers (continued)



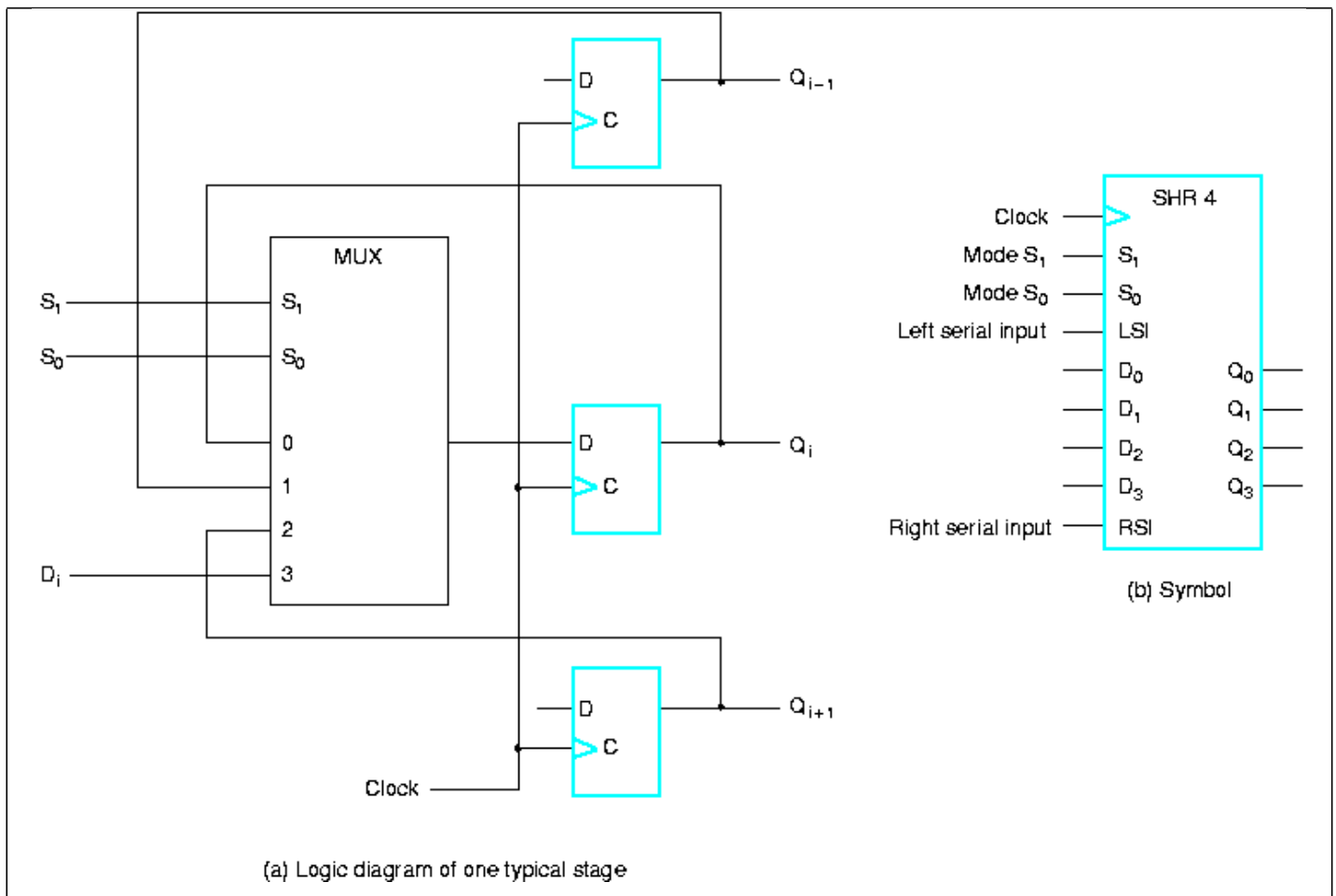
## Unidirectional Shift Register

# Shift Registers (continued)



Unidirectional Shift Register with parallel Load

# Shift Registers (continued)



## Bidirectional Shift Register with Parallel Load

Function Table of B.S.R.P.L		
Mode Control		Register Operation
$S_1$	$S_0$	
0	0	No Change
0	1	Shift Down
1	0	Shift Up
1	1	Parallel Load

# COUNTERS

A sequential circuit that goes through a prescribed sequence of states

Flip-flops are connected in cascade

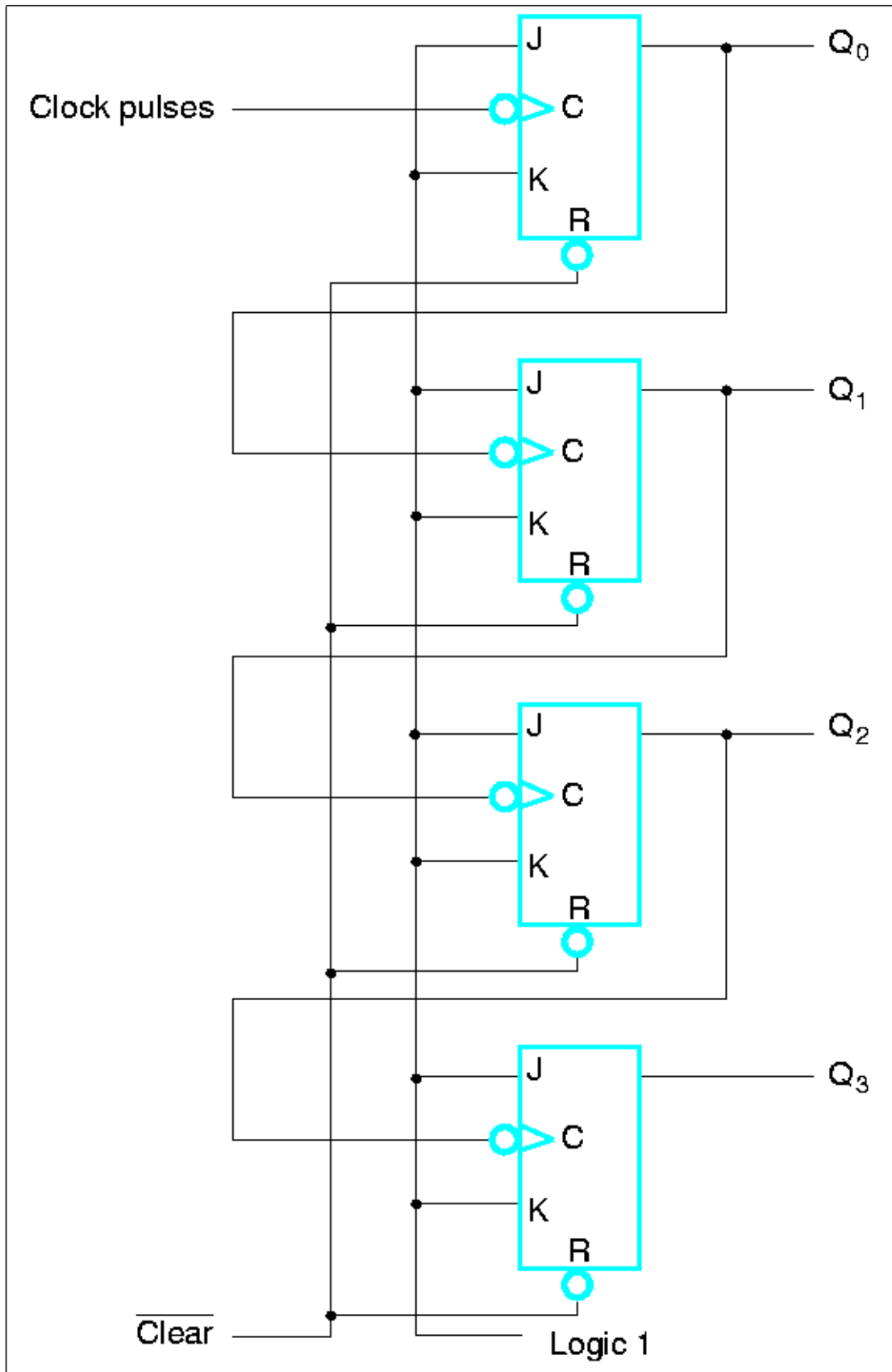
Input pulses may be from clock pulses or other sources

**Binary counter:** Counter that follows the binary number sequence. An  $n$ -bit binary counter has  $n$  flip-flops and can count in binary from 0 to  $2^n - 1$

**Ripple counter:** A flip-flop output is connected to the  $C$  input of the next flip-flop. The change of state is determined from the state of the preceding flip-flop

**Synchronous counter:** The  $C$  inputs of all the flip-flops receive the common clock pulse. The change of state is determined from the present state of the counter

# Ripple Counters



4-bit Binary Ripple Counter with  $JK$  Flip-flops

## Binary Ripple Counter

Count Sequence				
$Q_3$	$Q_2$	$Q_1$	$Q_0$	
0	0	0	0	
0	0	0	1	→ $Q_1$ will complement $Q_2$
0	0	1	0	
0	0	1	1	→ $Q_1$ will complement $Q_2$ $Q_2$ will complement $Q_3$
0	1	0	0	
0	1	0	1	→ $Q_1$ will complement $Q_2$
0	1	1	0	
0	1	1	1	→ $Q_1$ will complement $Q_2$ $Q_2$ will complement $Q_3$ $Q_3$ will complement $Q_4$
1	0	0	0	
⋮	...	⋮		⋮

Counters are best implemented with flip-flops capable of complementing their contents (example:  $T$  or  $JK$  flip-flops)

Content of a flip-flop is complemented if its  $C$  input changes from 1 to 0 (negative transition)

$Q_0$  is complemented after every pulse

$Q_i$  is complemented after every  $2^i$  pulses

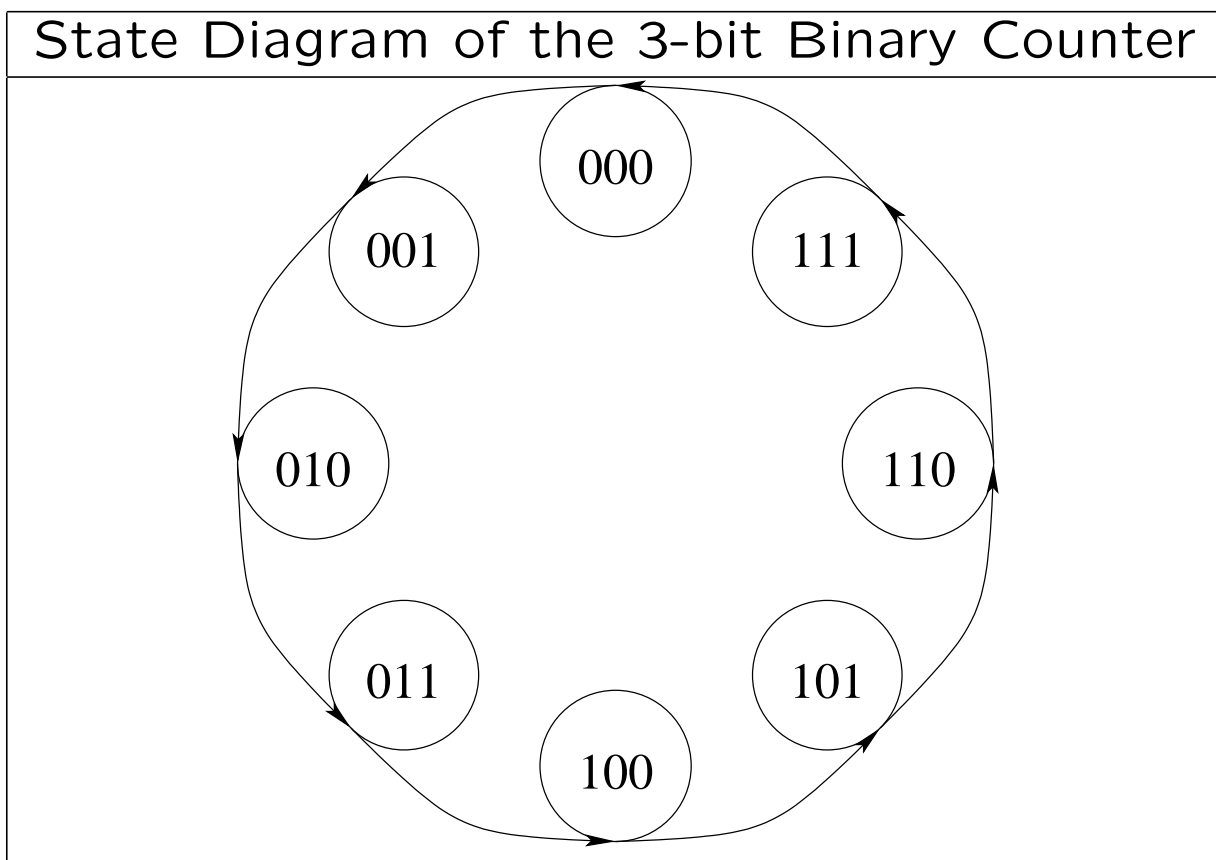
# Synchronous Binary Counters

A synchronous binary counter can be designed in the way any other synchronous sequential circuit is designed

Use the counting sequence as a state diagram

Design the counter from the state diagram

**Example:** Design a 3-bit binary counter with  $T$  flip-flops



# 3-Bit Binary Counter Design

With no external inputs and output we get

State Table of the 3-bit Binary Counter								
Present State			Next State			Flip-Flops Inputs		
$Q_2$	$Q_1$	$Q_0$	$Q_2$	$Q_1$	$Q_0$	$T_{Q_2}$	$T_{Q_1}$	$T_{Q_0}$
0	0	0	0	0	1	0	0	1
0	0	1	0	1	0	0	1	1
0	1	0	0	1	1	0	0	1
0	1	1	1	0	0	1	1	1
1	0	0	1	0	1	0	0	1
1	0	1	1	1	0	0	1	1
1	1	0	1	1	1	0	1	1
1	1	1	0	0	0	1	1	1

After K-map simplifications we get

$$T_{Q_2} = Q_1 Q_0$$

$$T_{Q_1} = Q_0$$

$$T_{Q_0} = 1$$

which gives the circuit below

