Sequential Logic Design Practices

Doru Todinca

Department of Computers
Politehnica University of Timisoara

Outline

Outline

Latches and Flip-flops

- A *flip-flop* samples its inputs and changes its outputs only at the changing of the CLOCK signal.
- Flip-flops can be:
  - positive edge triggered flip-flops: the flip-flop "works" when CLOCK changes from LOW to HIGH
  - negative edge triggered flip-flops: the CLOCK changes from HIGH to LOW
- A *latch* watches continuously its inputs and changes its outputs as long as the CLOCK signal has either a HIGH value, or a LOW value
- We discussed internal structure of several types of latches and flip-flops in Lecture 6
- Examples of SSI circuits containing flip-flops and latches:
  - 74x74: two independent positive edge-triggered D flip-flops with preset and clear inputs
  - 74x109: two independent positive edge-triggered J-K flip-flops with active-low K input, preset and clear inputs
  - 74x375: four D latches, arranged in pairs, with a common C (i.e. CLOCK) input for each pair

Outline

Multibit Registers and Latches

- *Definition:* a collection of two or more D flip-flops with a common clock input is called a *register*
- Registers are often used to store collections of related bits (like registers in computers), but they can also store unrelated data or control bits
- All bits are stored using the same clock signal
- Figure 1 shows the 74x175 4-bit register
- It contains four edge-triggered D flip-flops with a common clock and asynchronous clear inputs
- It provides both active-high and active-low (i.e. negated) outputs
- From the circuit diagram we can see that both CLK and CLR_L are buffered
- Fig 2 shows the 74x374 8-bit register with tri-state outputs and an active-low *output enable* (OE_L) signal that puts all 8 outputs in Hi-Z if it has a HIGH value

Multibit Registers and Latches

- The 74x373 (fig 3) is similar to '374, but contains D latches instead of flip-flops
- 74x273 (from fig 4) is like '374, but its outputs are not tri-state, and it has an asynchronous clear input (CLR_L) instead of the output enable input
- Figure 5 shows the 74x377 8 bit register with gated clock (the outputs are not tri-state)
- Remember from Lecture 6 (fig 24) that the clock signal is not gated, but the active-low EN_L input is used as a selection signal for the MUX situated at the D input of each flip-flop, in order to choose between the new D value (8D in fig 5) and the previous value stored in the flip-flop (the Q output of the flip-flop)
- The MUX consists of 3 gates: two AND gates and one OR gate

Multibit Registers and Latches: 74x175

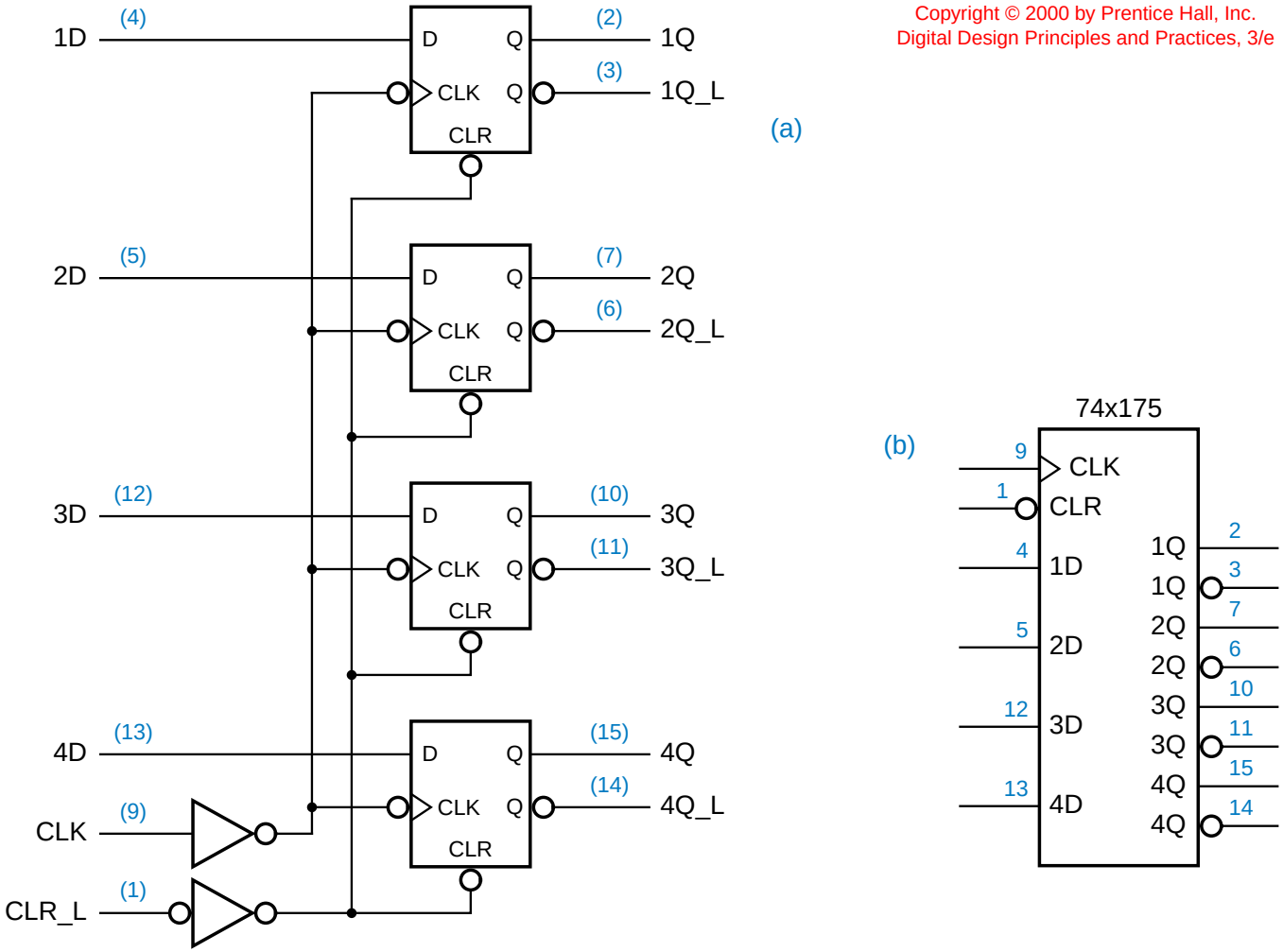

Figure 1: The 74x175 4-bit register: (a) logic diagram, including pin-numbers for a standard 16-pin dual-in-line package; (b) logic symbol

Multibit Registers and Latches: 74x374

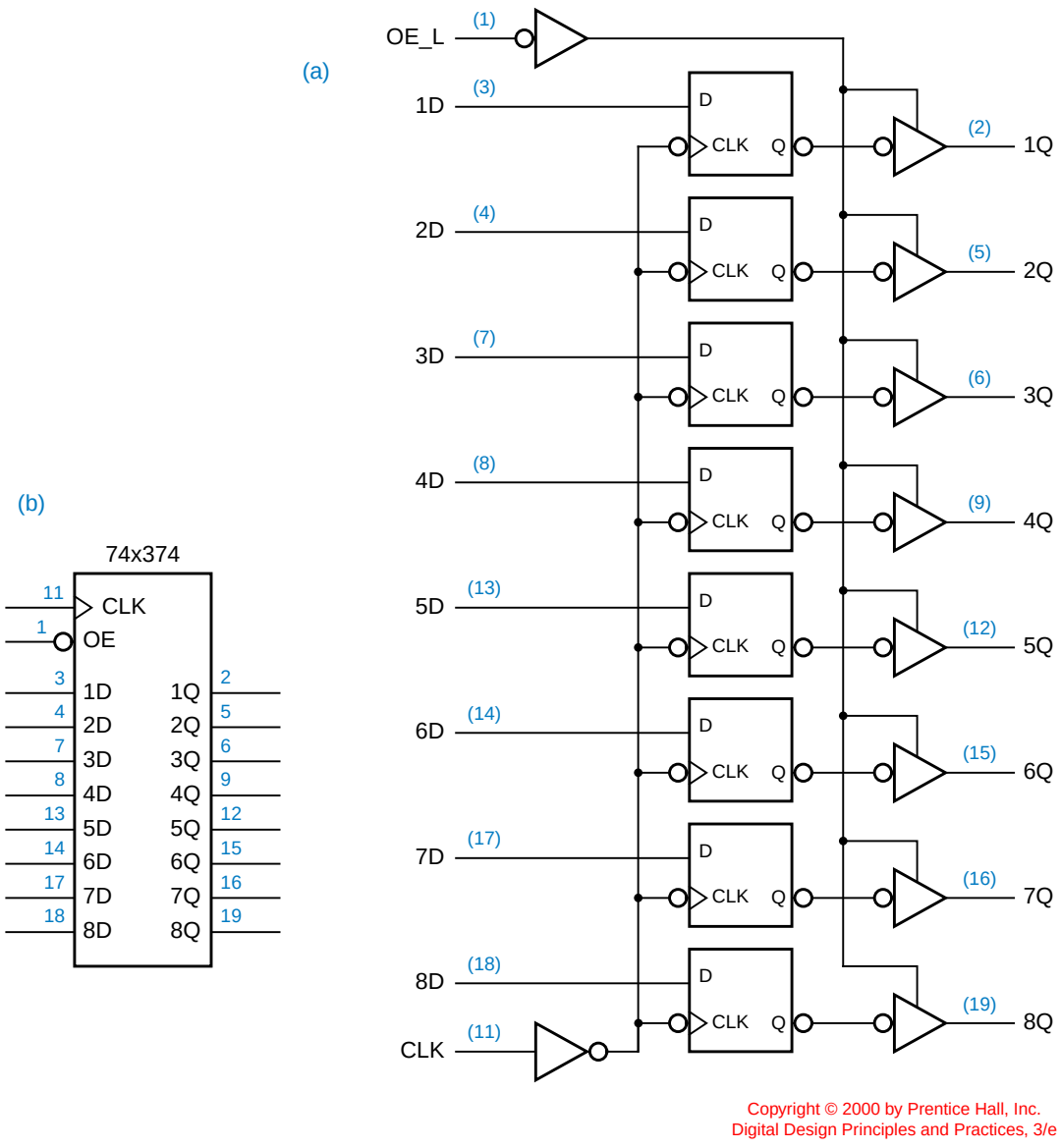

Figure 2: The 74x374 8-bit register: (a) logic diagram, including pin-numbers for a standard 20-pin dual-in-line package; (b) logic symbol

Multibit Registers and Latches: 74x373 and 74x273

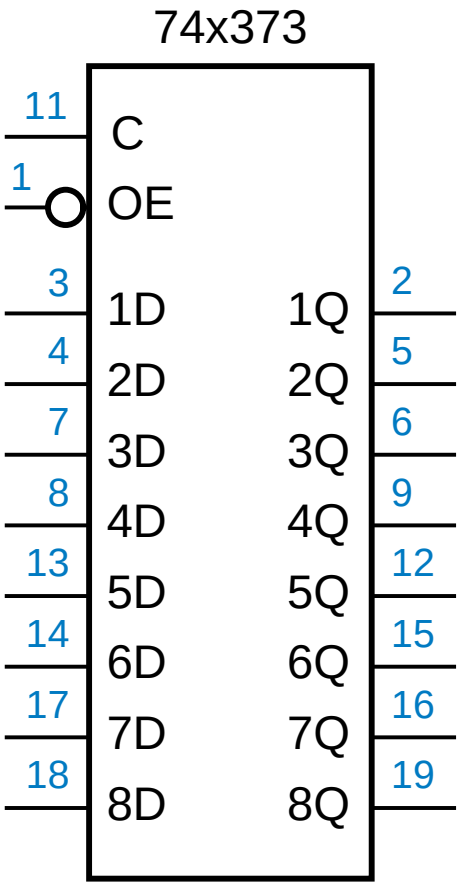

Figure 3: Logic symbol for the 74x373 8-bit latch

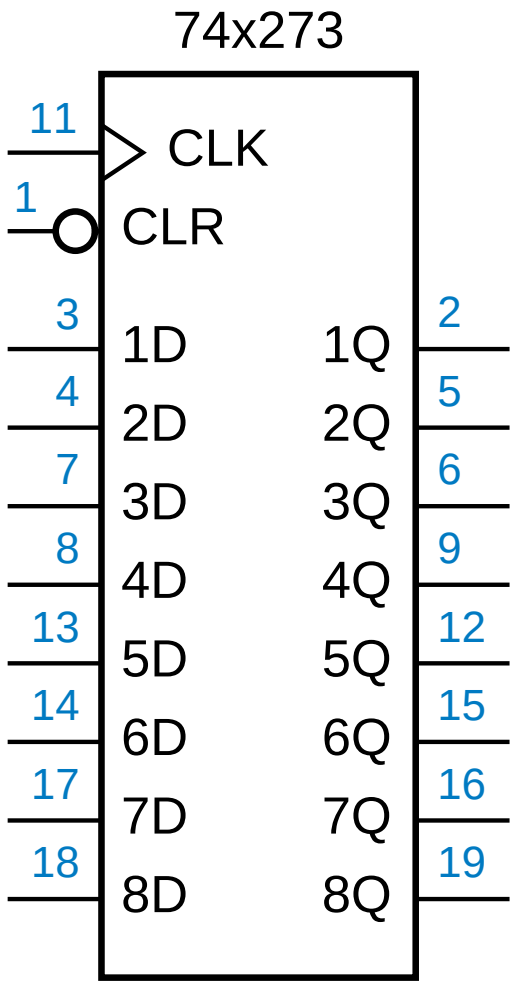

Figure 4: Logic symbol for the 74x273 8-bit register

Multibit Registers and Latches: 74x377

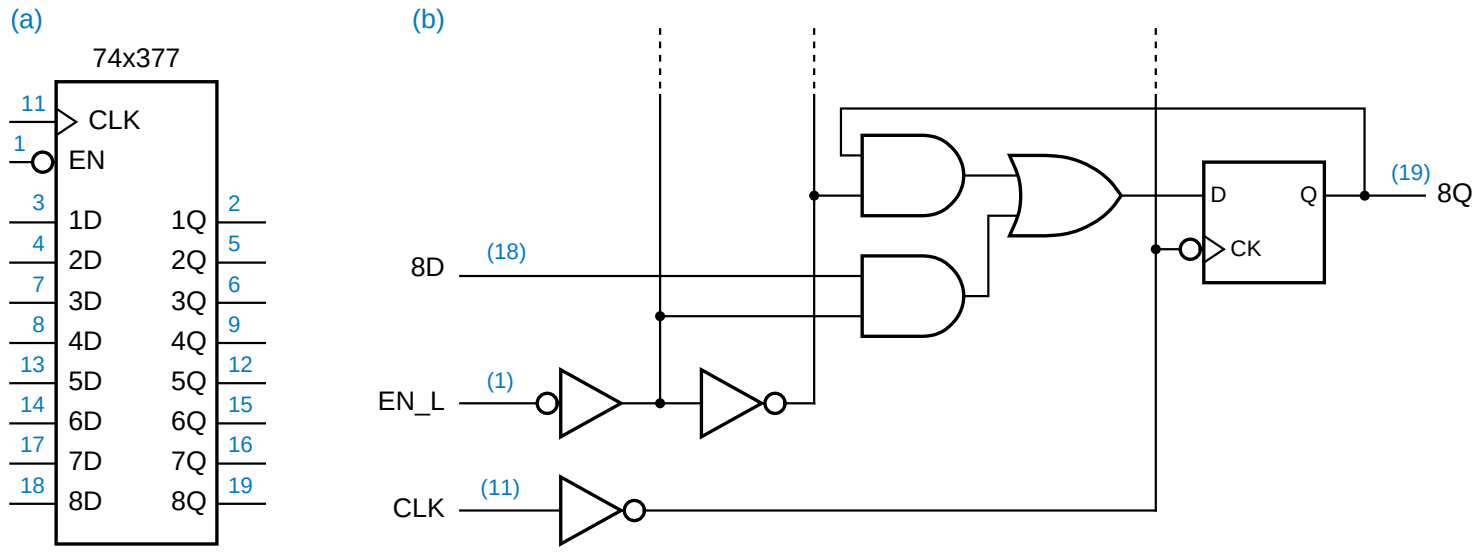

Figure 5: The 74x377 8-bit register with gated clock: (a) logic symbol; (b) logic behavior of one bit

Outline

Counters

- A *counter* is a clocked sequential circuit whose state diagram contains s single cycle, like in figure 6
- The *modulus* of a counter is the number of states in the cycle
- A counter with m states is called a *modulo-m counter* or a *divide-by-m counter*
- If m is not a power of 2, then there are states that are not used in normal operation
- Probably the most commonly used counter type is an *n-bit binary counter*, that has n flip-flops and 2^n states visited in the sequence $0, 1, \ldots, 2^n - 1, 0, 1, \ldots$
- Each of these n states is encoded as the corresponding binary n bit integer

Counters: State Diagram

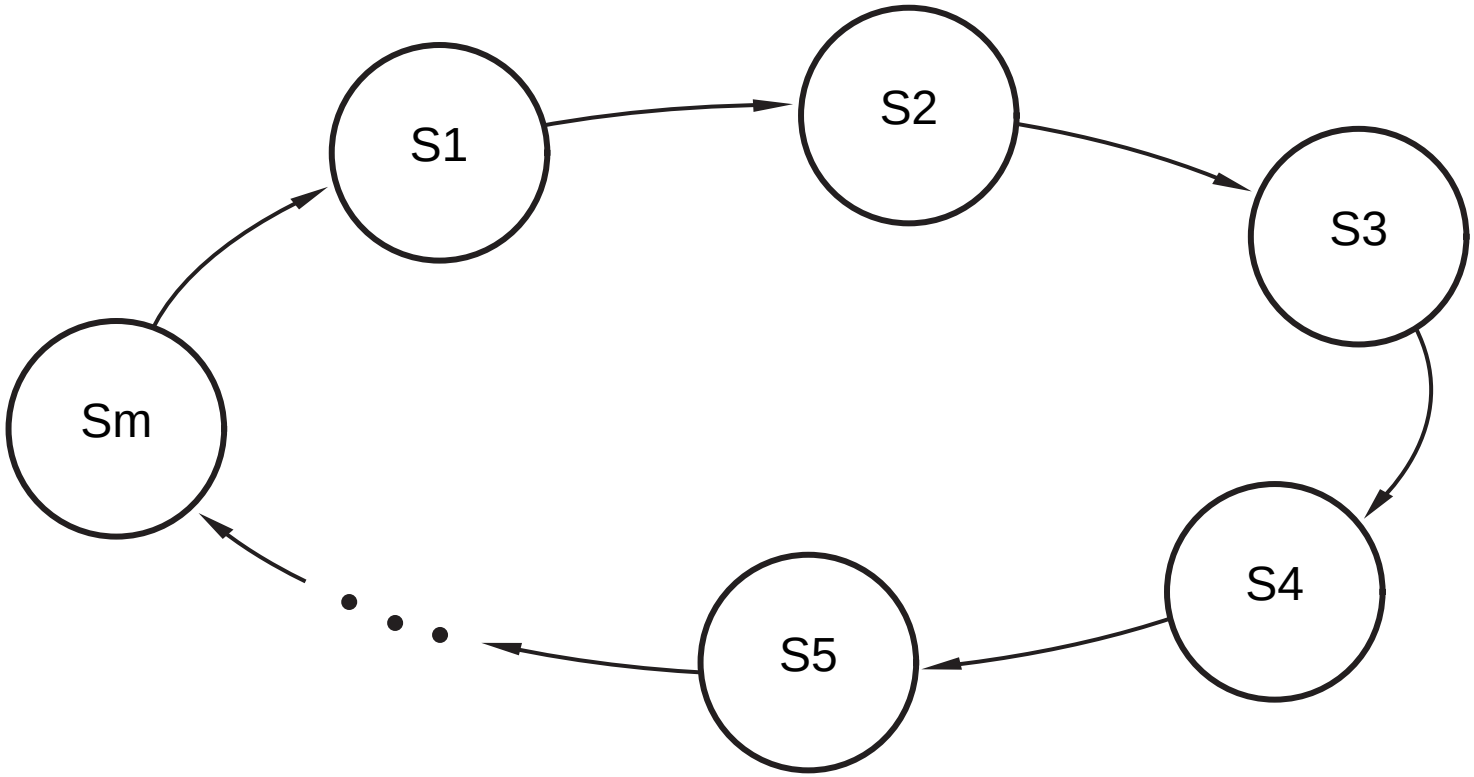

Figure 6: General structure of a counter's state diagram – a single cycle

Outline

Counters: Ripple Counter

- Figure 7 shows an example of n-bit (4-bit in this case) binary counter constructed only with flip-flops, and no other components
- In this case there are four T flip-flops and each T flip-flop toggles (i.e. changes state) on the rising edge of its clock
- In this case, each bit of the counter toggles if and only if the immediately preceding bit changes from 1 to 0
- This corresponds to a normal binary counting sequence: when a bit changes from 1 to 0 it generates a carry to the next more significant bit
- This counter is called a *ripple counter* because the carry information ripples from the less significant bits to the more significant bits, one bit at a time
- The problems of this counter are that it is asynchronous and that it is slower than other types of binary counters (the change must propagate, in the worst case, from the least significant to the most significant bit)

Counters: Ripple Counter

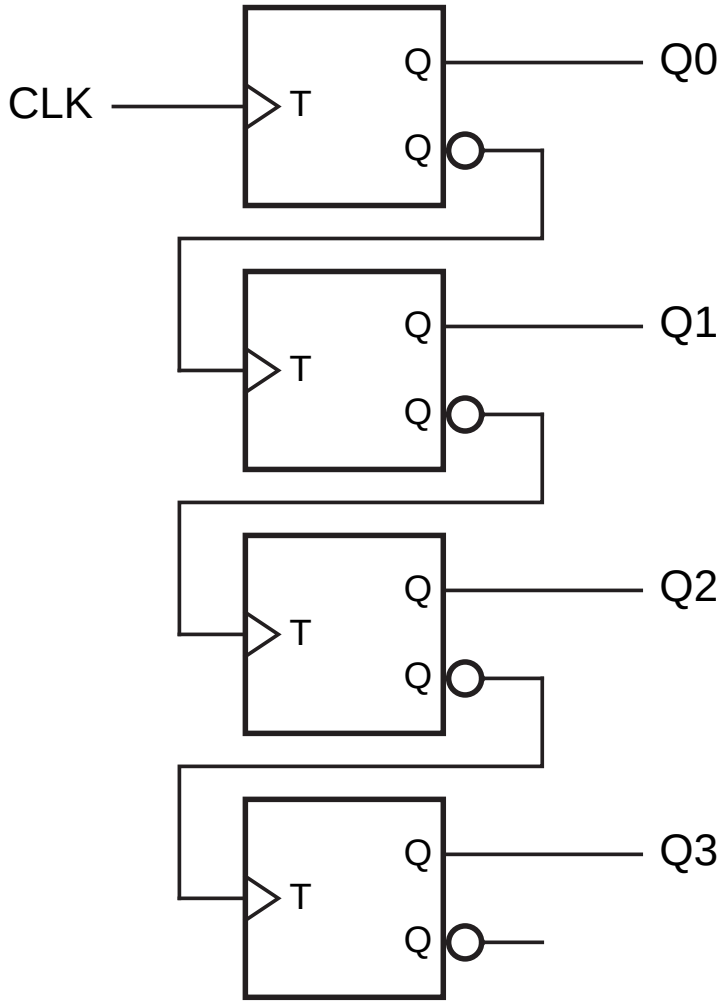

Figure 7: A 4-bit binary ripple counter

Outline

Synchronous Counters

- All flip-flops have a common clock (CLK) signal
- Hence, all flip-flops outputs change in the same time
- Figure 8 shows such a counter constructed with T flip-flops with enable inputs (EN on the figure)
- Combinational logic on the EN inputs determines which (if any) flip-flops toggle on the rising edge of CLK
- CTEN is a master count enable signal
- Each flip-flop toggles if and only if CTEN is asserted and all the lower order bits are 1
- The counter in fig 8 is called *synchronous serial counter* because the combinational enable signals propagate serially from the least significant bit (LSB) to the most significant bit (MSB)
- If the clock period is too short, there may not be enough time for a change to propagate from the LSB to MSB

Synchronous Counters

- This problem is solved at the *synchronous parallel counter* in figure 9
- Here there is a single level of logic (an AND gate) for each EN input
- You will see this approach for binary adder circuits: there are ripple carry adders and parallel adders:
- The condition to have a carry is generated serially, from LSB to MSB at ripple carry adders, and is anticipated using a more complex, but faster combinational logic, at parallel adders

Synchronous Counters: With Serial Enable Logic

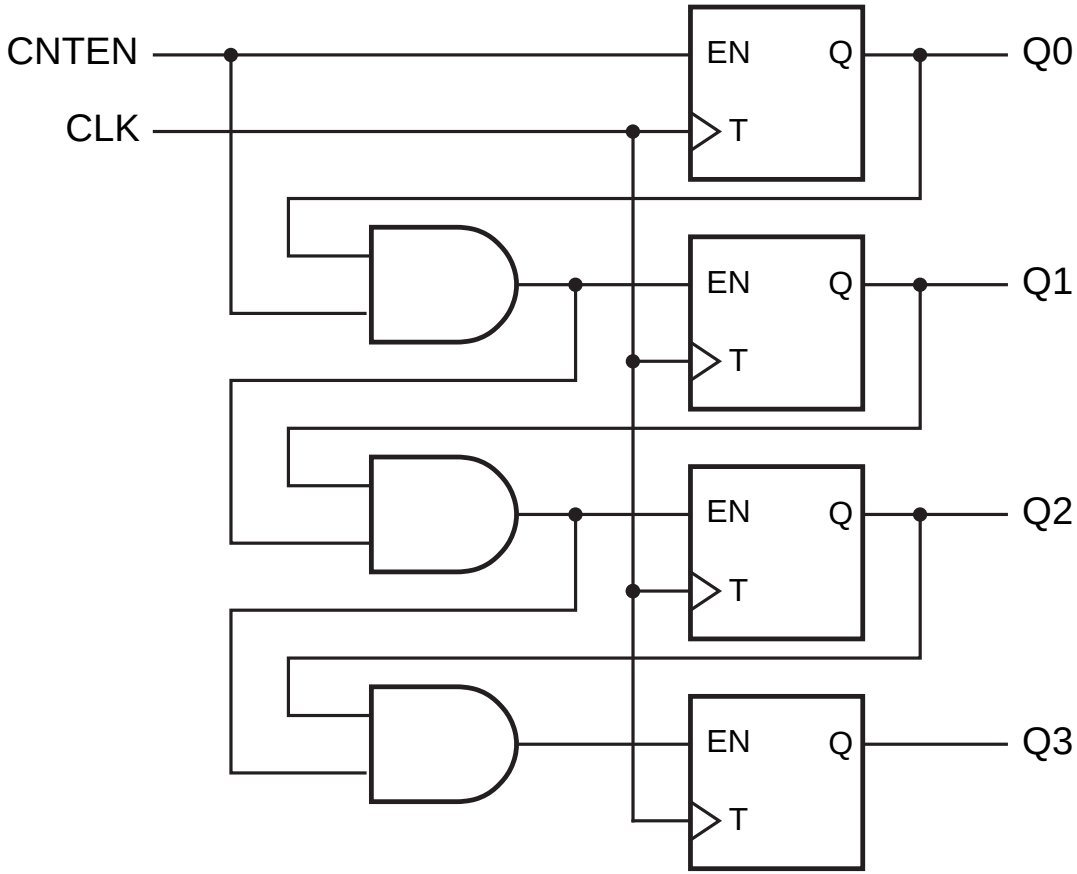

Figure 8: A synchronous 4-bit binary counter with serial enable logic

Synchronous Counters: With Parallel Enable Logic

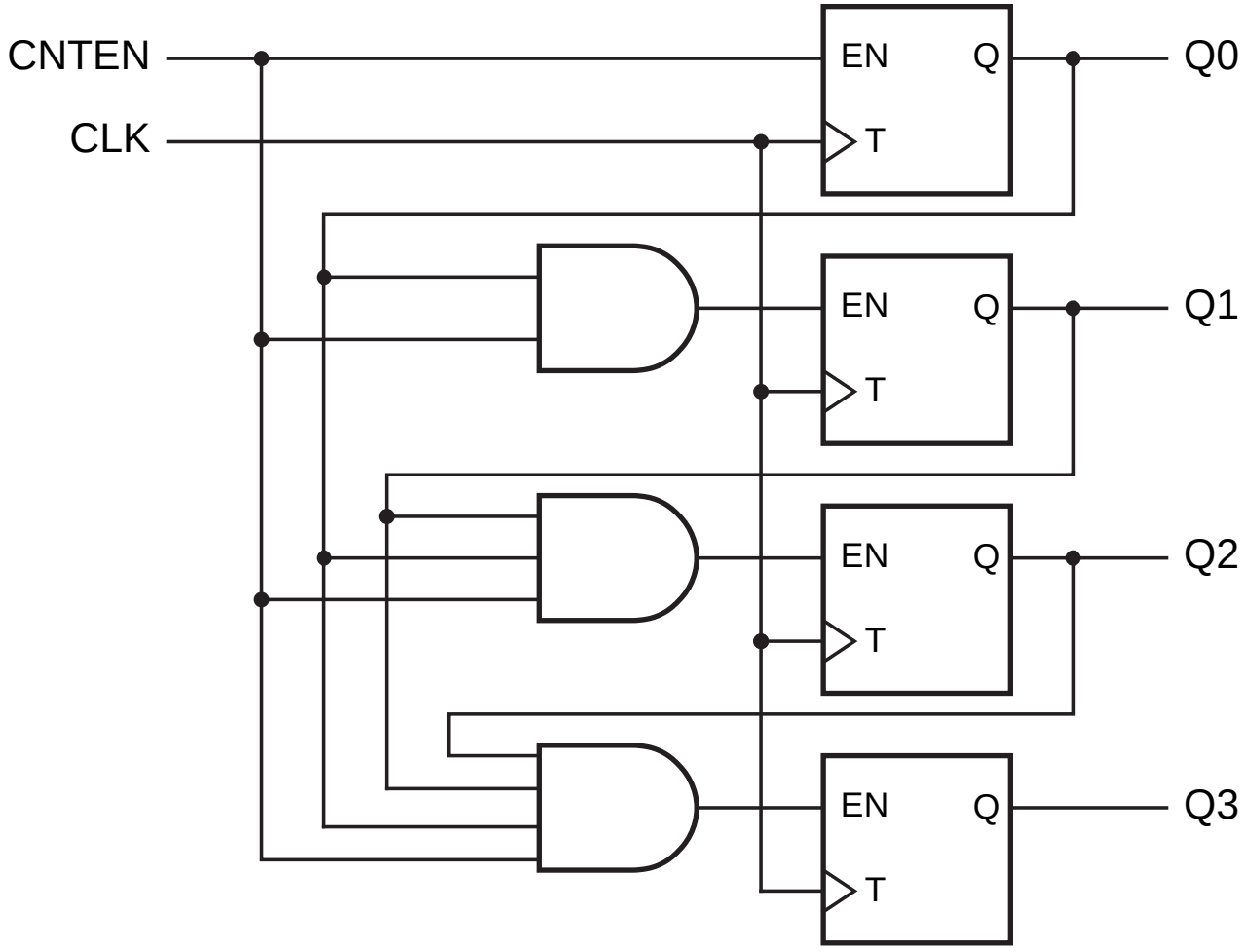

Figure 9: A synchronous 4-bit binary counter with parallel enable logic

Outline

MSI Synchronous Counters: 74x163

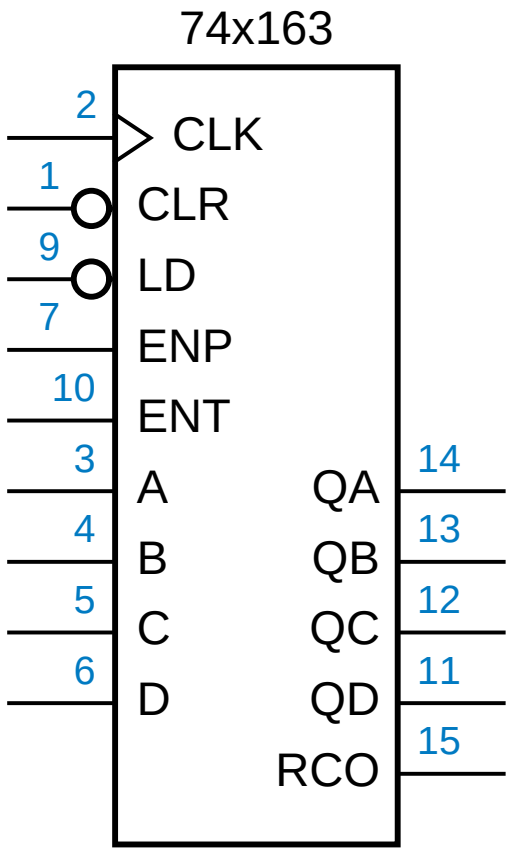

Figure 10: Logic symbol for the 74x163

- A "classic" MSI counter is 74x163: 4-bit binary counter with synchronous load and clear inputs
- Its logic symbol is shown in figure 10, the state table in Table 8-11 (fig 11), and the logic diagram in fig 12
- Because it can be loaded with an input combination, it is constructed from D, not T flip-flops
- At each D input there is a 2:1 MUX, consisting of 2 AND gates and one OR gate

MSI Synchronous Counters: 74x163

- If $CLR\_L = 0$ the output of two NOR gates is 0, hence the output of the AND gates in the MUX is 0, thus the output of the MUX is 0
- If $CLR\_L = 1$ and $LD\_L = 0$ the output of the upper NOR gate is 1, the output of the lower NOR gate is 0, hence the output of the MUX will be the parallel input (A, B, C, or D)
- If $CLR\_L = LD\_L = 1$, the output of the upper NOR gate is 0, the output of the lower NOR gate is 1 and hence output of the XNOR gate will pass to D (the counting function is active)
- The counting function is implemented by the XNOR gate: one input is the count bit (QA, QB, QC, QD), and the other input is 1, which complements the count bit, if and only if both enables ENP and ENT are asserted and all of the lower-order count bits are 1

State Table for a 74x163 Binary Counter

Table 8-11 State table for a 74x163 4-bit binary counter.

Inputs				*Current State*				*Next State*			
CLR_L	*LD_L*	*ENT*	*ENP*	*QD*	*QC*	*QB*	*QA*	*QD**	*QC**	*QB**	*QA**
0	x	x	x	x	x	x	x	0	0	0	0
1	0	x	x	x	x	x	x	D	C	B	A
1	1	0	x	x	x	x	x	QD	QC	QB	QA
1	1	x	0	x	x	x	x	QD	QC	QB	QA
1	1	1	1	0	0	0	0	0	0	0	1
1	1	1	1	0	0	0	1	0	0	1	0
1	1	1	1	0	0	1	0	0	0	1	1
1	1	1	1	0	0	1	1	0	1	0	0
1	1	1	1	0	1	0	0	0	1	0	1
1	1	1	1	0	1	0	1	0	1	1	0
1	1	1	1	0	1	1	0	0	1	1	1
1	1	1	1	0	1	1	1	1	0	0	0
1	1	1	1	1	0	0	0	1	0	0	1
1	1	1	1	1	0	0	1	1	0	1	0
1	1	1	1	1	0	1	0	1	0	1	1
1	1	1	1	1	0	1	1	1	1	0	0
1	1	1	1	1	1	0	0	1	1	0	1
1	1	1	1	1	1	0	1	1	1	1	0
1	1	1	1	1	1	1	0	1	1	1	1
1	1	1	1	1	1	1	1	0	0	0	0

Figure 11: State table for a 74x163 4-bit binary counter

MSI Synchronous Counters: 74x163

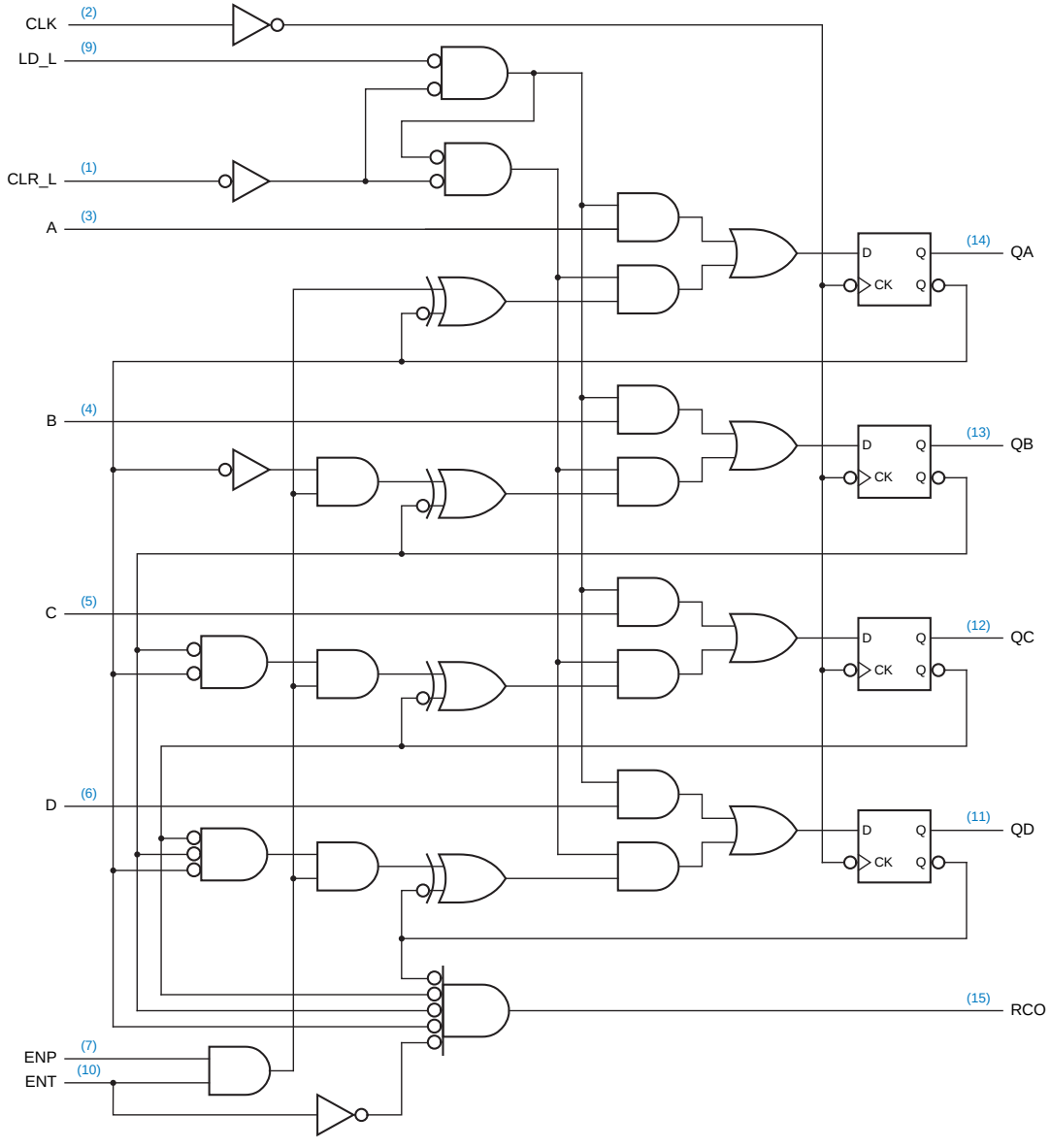

Figure 12: Logic diagram for 74x163

Applications of 74x163

- Even though 74x163, like most MSI counter, has enable inputs, it can be used in a *free-running mode*, in which it is enabled continuously
- Figure 13 shows the connections for the free-running mode, and fig 14 shows the resulting waveform

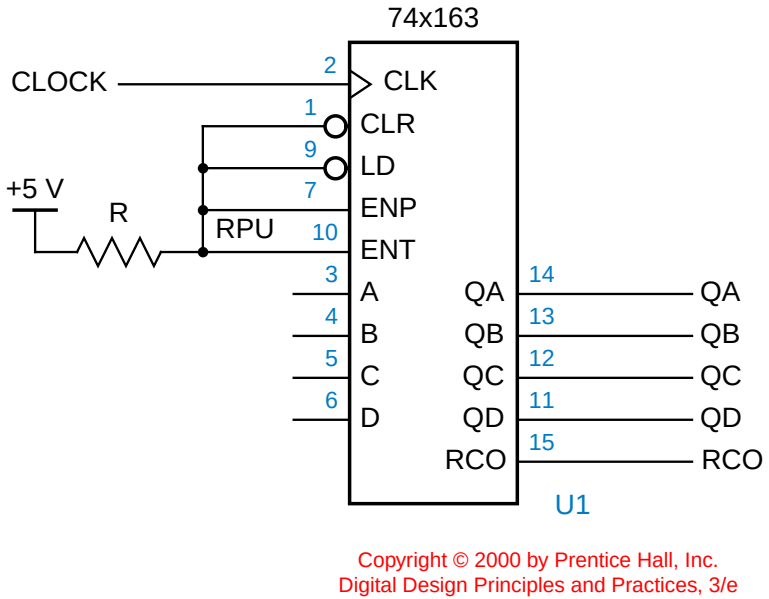

Figure 13: Connections for the 74x163 to operate in a free running mode

Synchronous Counters Applications

- Each output, starting from QA, has the half frequency of the preceding one
- Thus, a free-running '163 can be used as a divide-by-2, -by-4, -8, or -16 counter
- The RCO (ripple carry out) signal of '163 indicates a carry from the most significant bit
- RCO is 1 when all of the count bits are 1 and ENT is asserted
- RCO can be used to "cascade" two 74x163 circuits in order to obtain a 8-bit binary counter: RCO of the "least significant circuit" will connect to the enable input of the "most significant circuit"
- The 74x161 has the same pinout as '163, and the same functions, but it has an asynchronous clear CLR_L
- 74x162 is like '163, but it counts modulo 10 (from 0 to 9 and again $0, 1, \ldots$) and it is called a *decade counter*
- 74x160 is a decade counter with asynchronous clear
- Figure 15 shows the output waveforms of a free-running '160 or '162

Synchronous Counters Applications

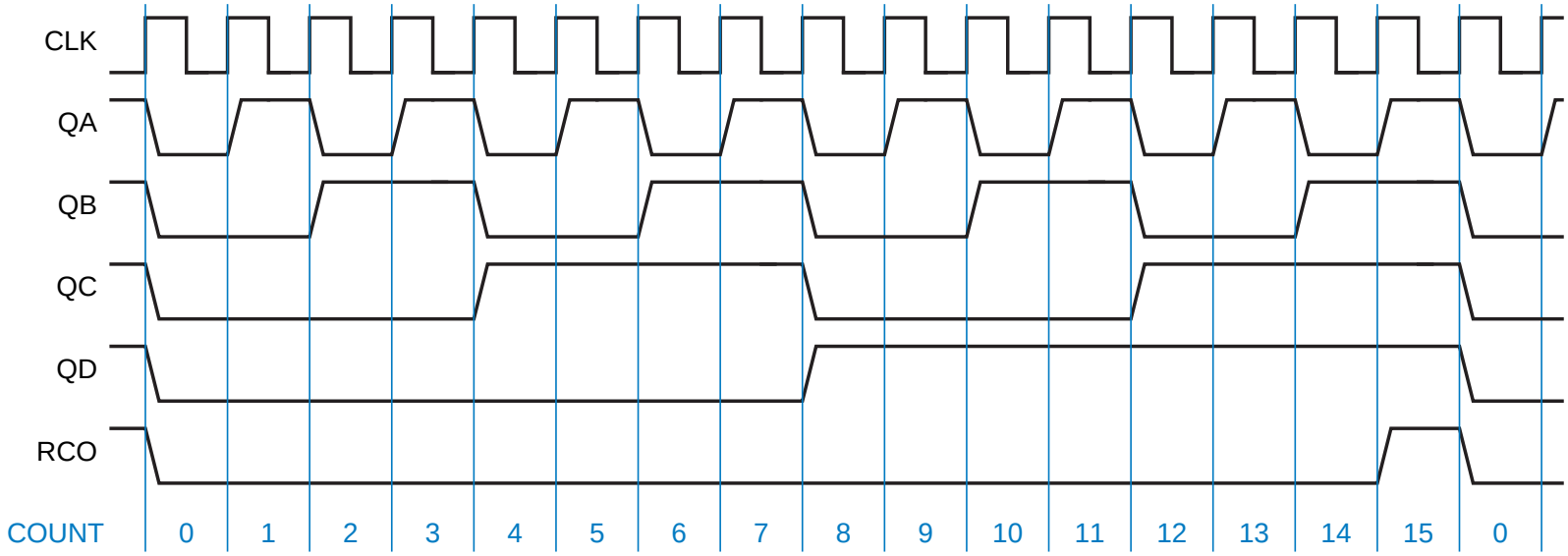

Figure 14: Clock and output waveforms for a free-running divide-by-16 counter

Synchronous Counters Applications

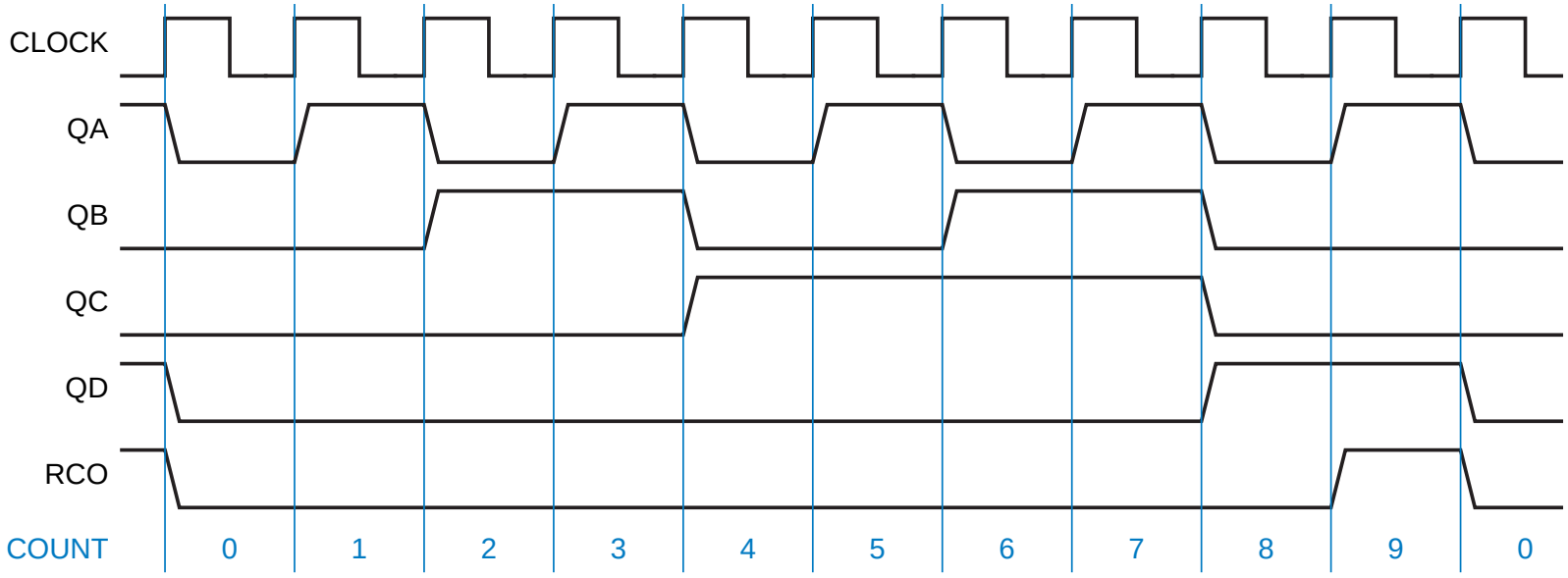

Figure 15: Clock and output waveforms for a free-running divide-by-10 counter

74x169

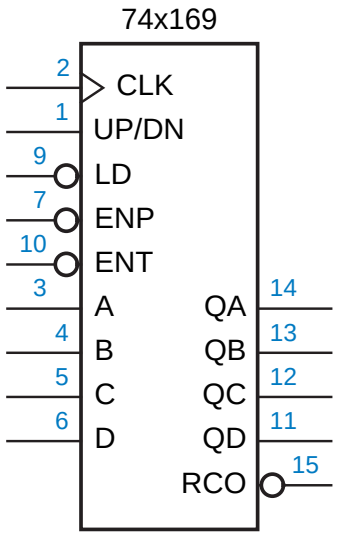

Figure 16: Logic symbol for the 74x169 up/down counter

- 74x169 is a synchronous up/down counter: it counts in ascending or descending order, depending on the input signal UP/DN
- Its symbol is shown in fig 16

Outline

Decoding Binary-Counter States

- Sometimes counters are used to control a set of devices, where a different device is enabled in each counter state
- Then, the outputs of a binary counter are decoded by a decoder, like in fig 17, resulting a timing diagram like in fig 18
- Glitches from fig 18 appear because the outputs of the counter don't change in exactly the same time and, more important, because on the decoder there are different delays on different signal paths
- Many times the glitches are not harmful, i.e., if the decoder outputs are used as control inputs to registers, counters, edge-triggered devices, when they are used *after* the clock tick, so they are gone before the next tick
- If the application requires, a solution for glitch removal is shown in fig 19: the output of the decoder is connected to another register, that samples the stable decoder outputs in the next clock tick

Decoding Binary-Counter States

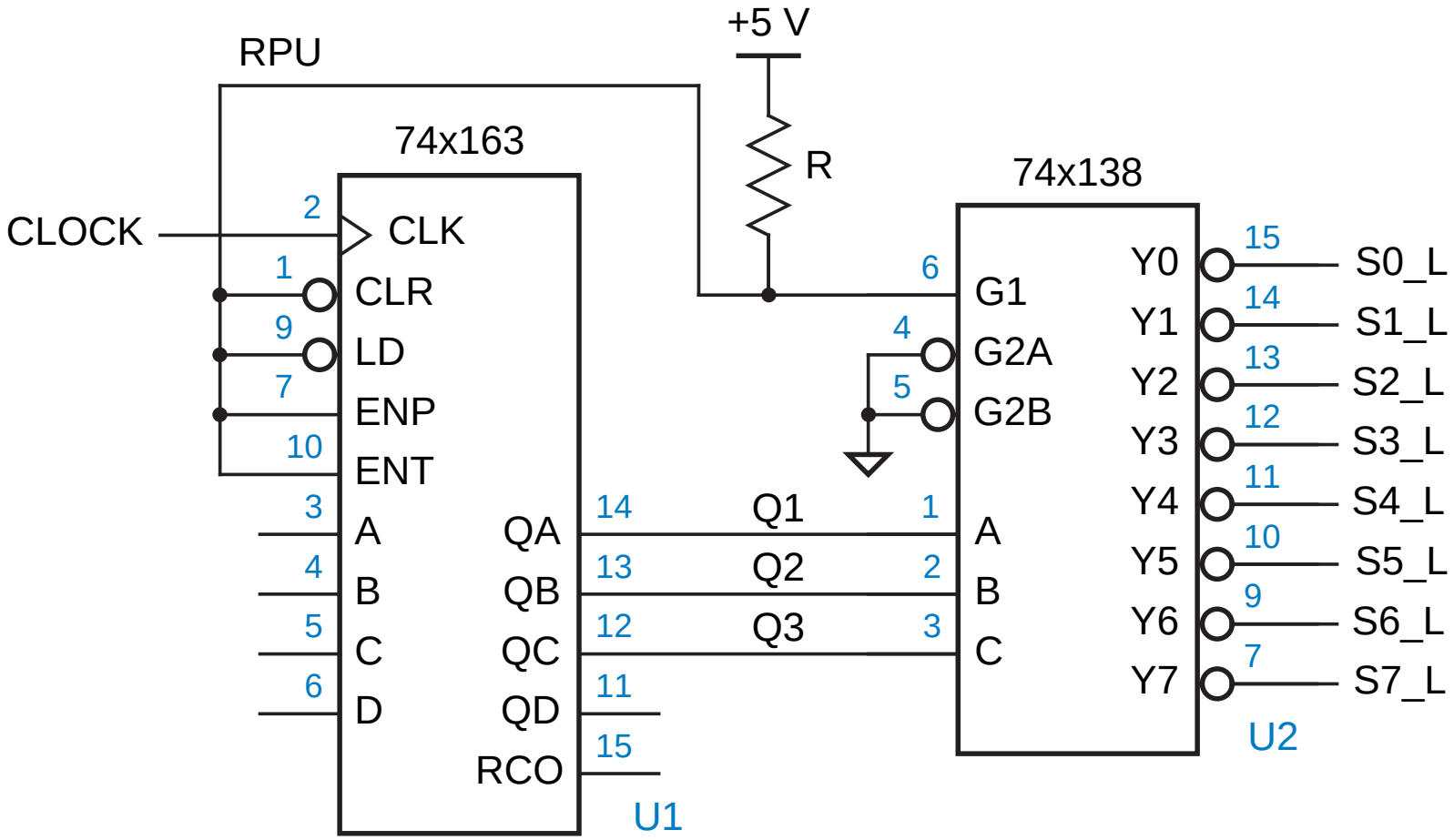

Figure 17: A modulo-8 binary counter and decoder

Decoding Binary-Counter States

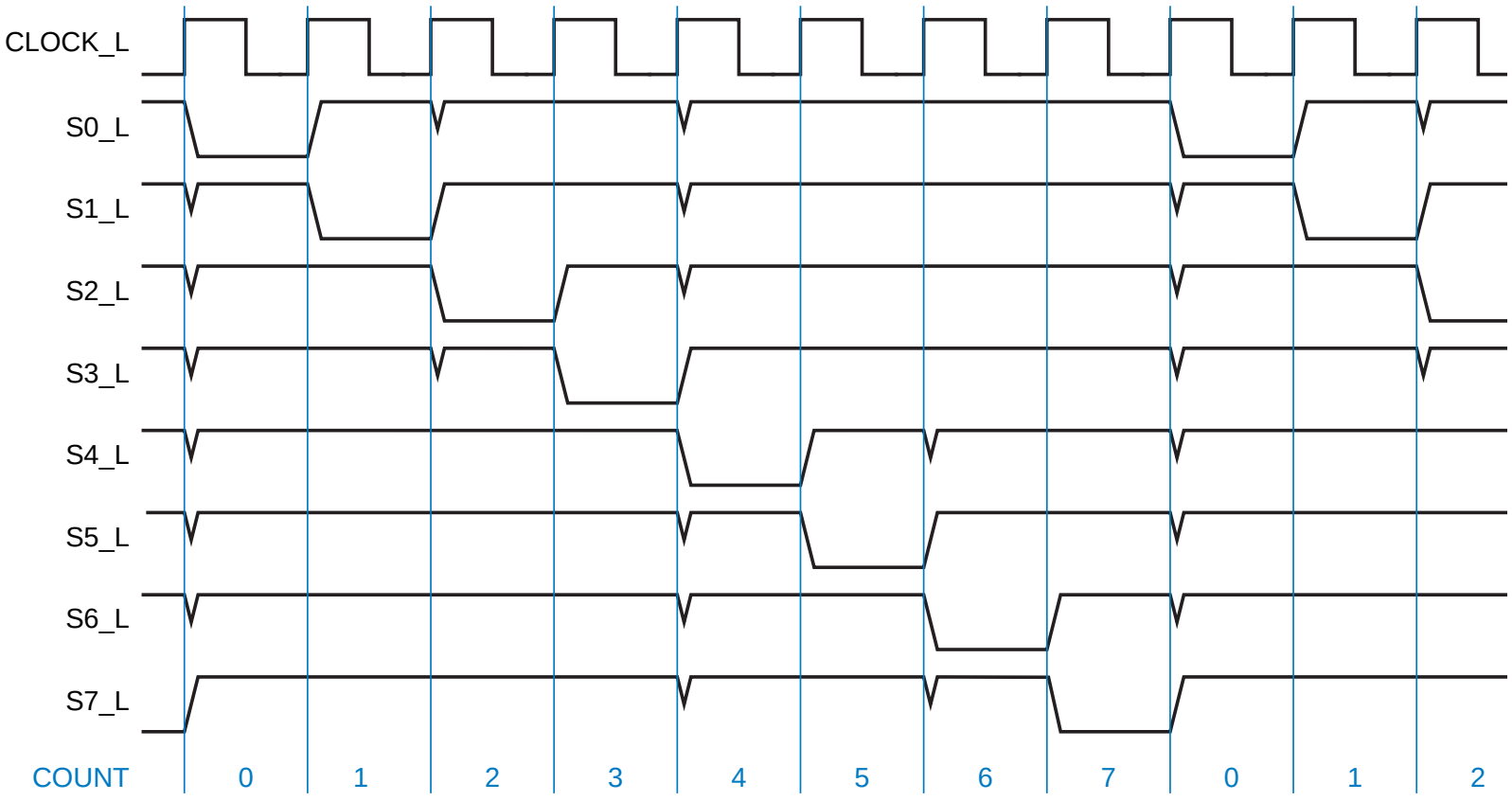

Figure 18: Timing diagram for a modulo-8 binary counter and decoder, showing decoding glitches

Decoding Binary-Counter States

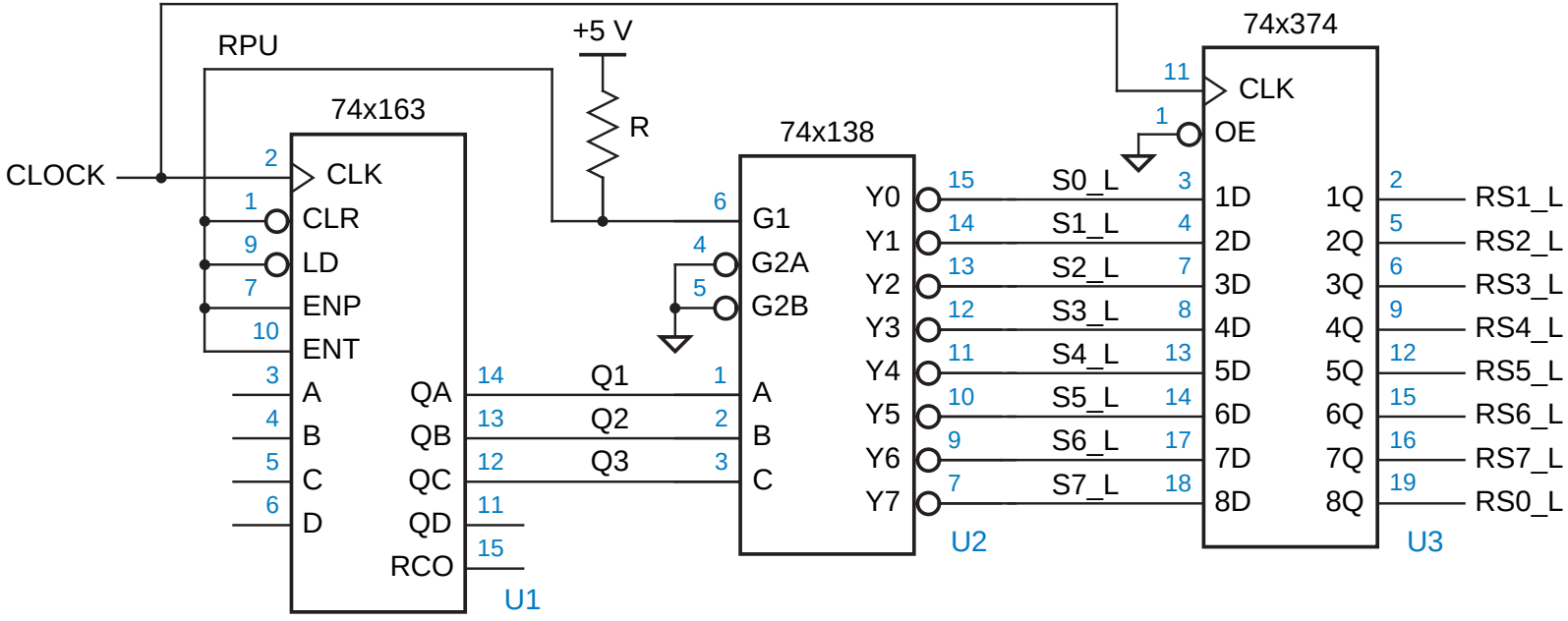

Figure 19: A modulo-8 binary counter and decoder with glitch-free outputs

Outline

Outline

Shift-Register Structure

- A *shift register* is an n-bit register that shifts its stored data by one bit at each tick of the clock
- Fig 20 shows the structure of a serial-in serial-out shift register
- The *serial input*, SERIN, specifies a new bit to be shifted at each clock tick
- This bit appears at the *serial output*, SEROUT, after n clock ticks, and is lost one tick after
- Hence, a serial-in shift register can be used to delay a signal by n clock ticks
- A *serial-in, parallel-out shift register*, shown in fig 21, has outputs for all its stored bits
- Such a shift register can be used for *serial-to-parallel conversion*

Shift-Register Structure

- Conversely, it is possible to build a *parallel-in, serial-out shift register*, like in fig 22
- The circuit uses a 2-input multiplexer on each flip-flop's D input to select between parallel and serial data
- At each clock tick, the register either loads new data from the parallel inputs 1D-ND, or it shifts its current content, depending on the value of the LOAD/SHIFT control input
- A parallel-in, serial-out shift register can be used to perform *parallel-to-serial conversion*
- The parallel-in, parallel-out shift register from fig 23 encompasses the functions of all previous shift registers

Serial-In, Serial-Out Shift Register

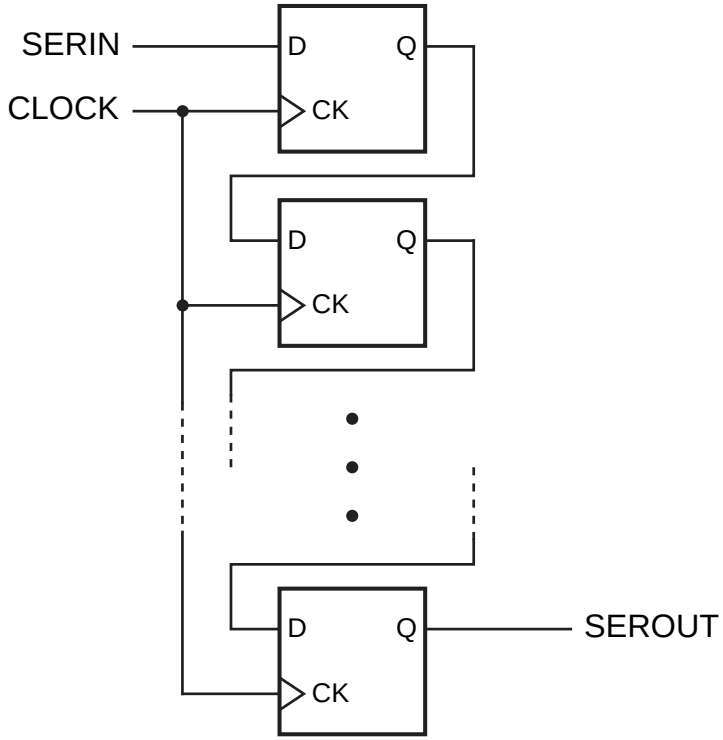

Figure 20: Structure of a serial-in, serial-out shift register

Serial-In, Parallel-Out Shift Register

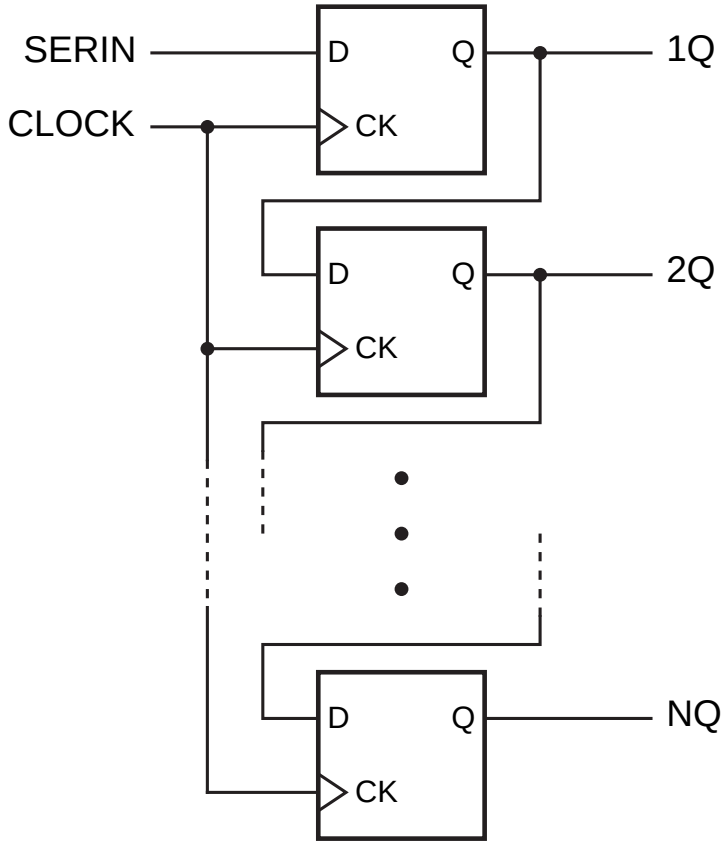

Figure 21: Structure of a serial-in, parallel-out shift register

Parallel-In, Serial-Out Shift Register

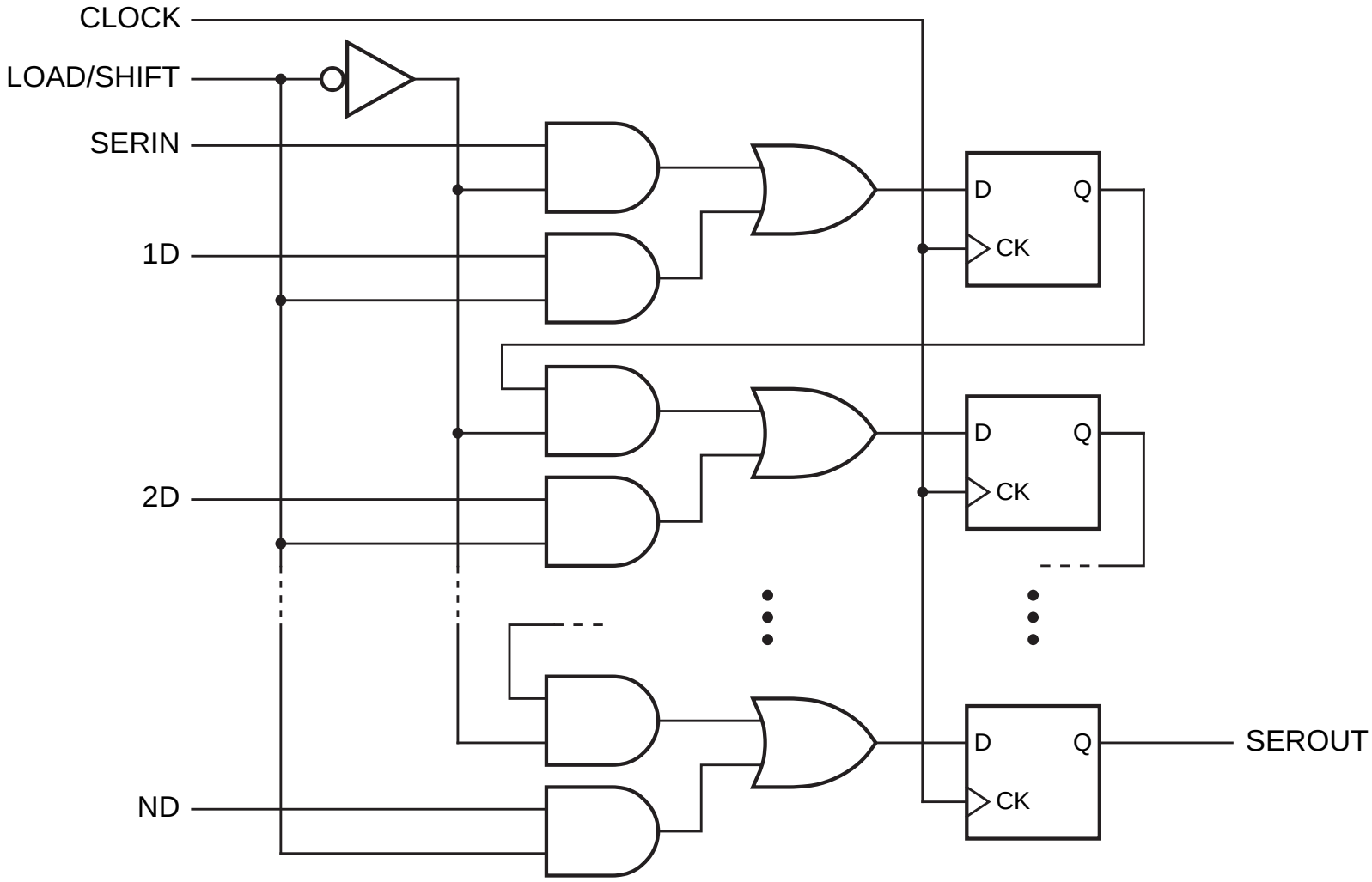

Figure 22: Structure of a parallel-in, serial-out shift register

Parallel-In, Parallel-Out Shift Register

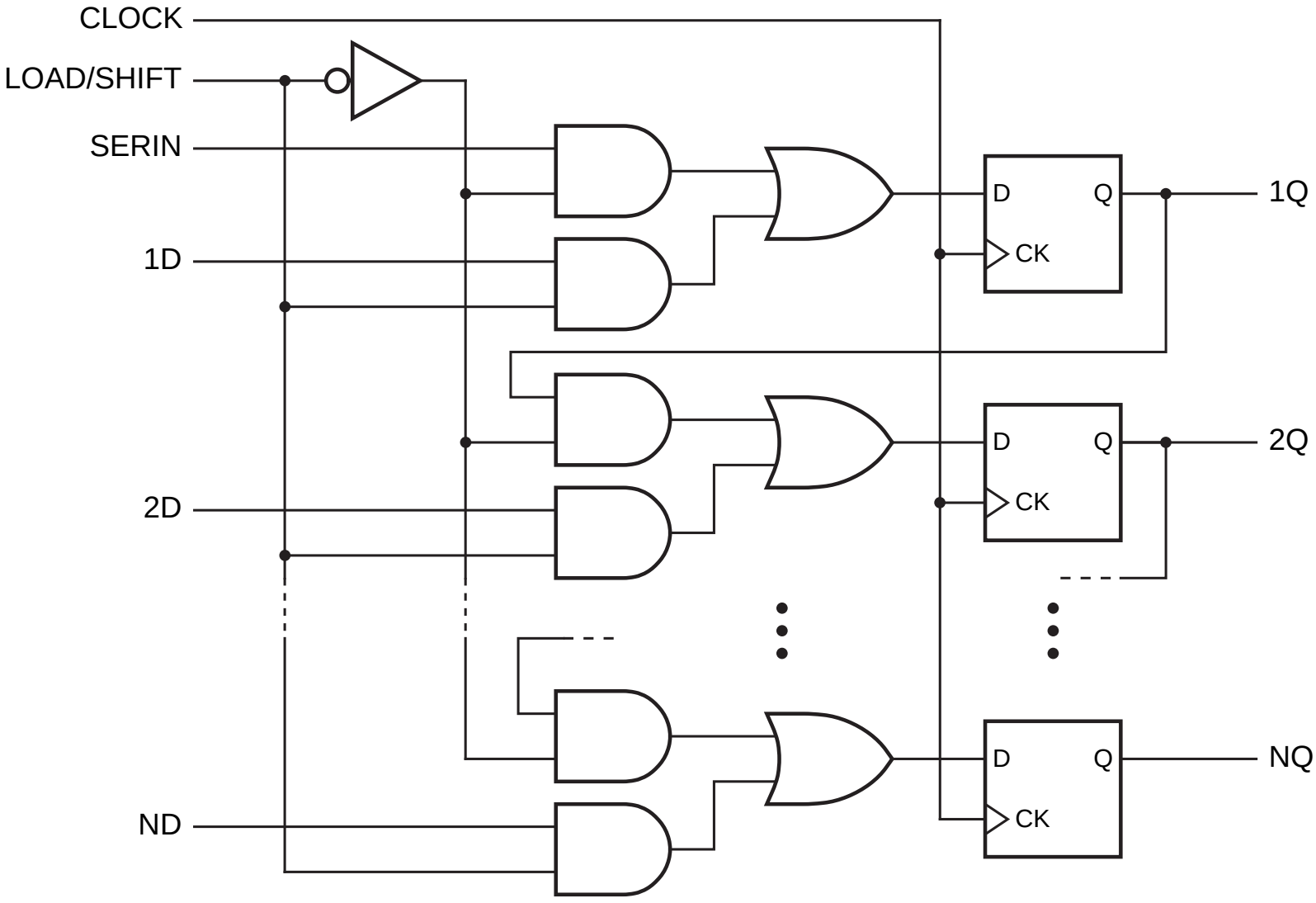

Figure 23: Structure of a parallel-in, parallel-out shift register

Outline

74x194 Universal Shift Register

- The 74x194 is an MSI 4-bit bidirectional, parallel-in, parallel-out shift register
- Its logic diagram is shown in figure 24
- The previous shift registers are called *unidirectional* because they shift in only one direction
- The '194 is called *bidirectional shift register*, because it can shift in either of two directions, depending on the control input(s)
- The two directions are called *right* (from QA to QD) and *left* (from QD to QA)
- Table 8-18 (fig 25) shows the function table for '194
- The 74x194 is called *universal* shift register because it can work in either of previous modes (serial-in; parallel-in; serial-out; parallel-out)

74x194 Universal Shift Register

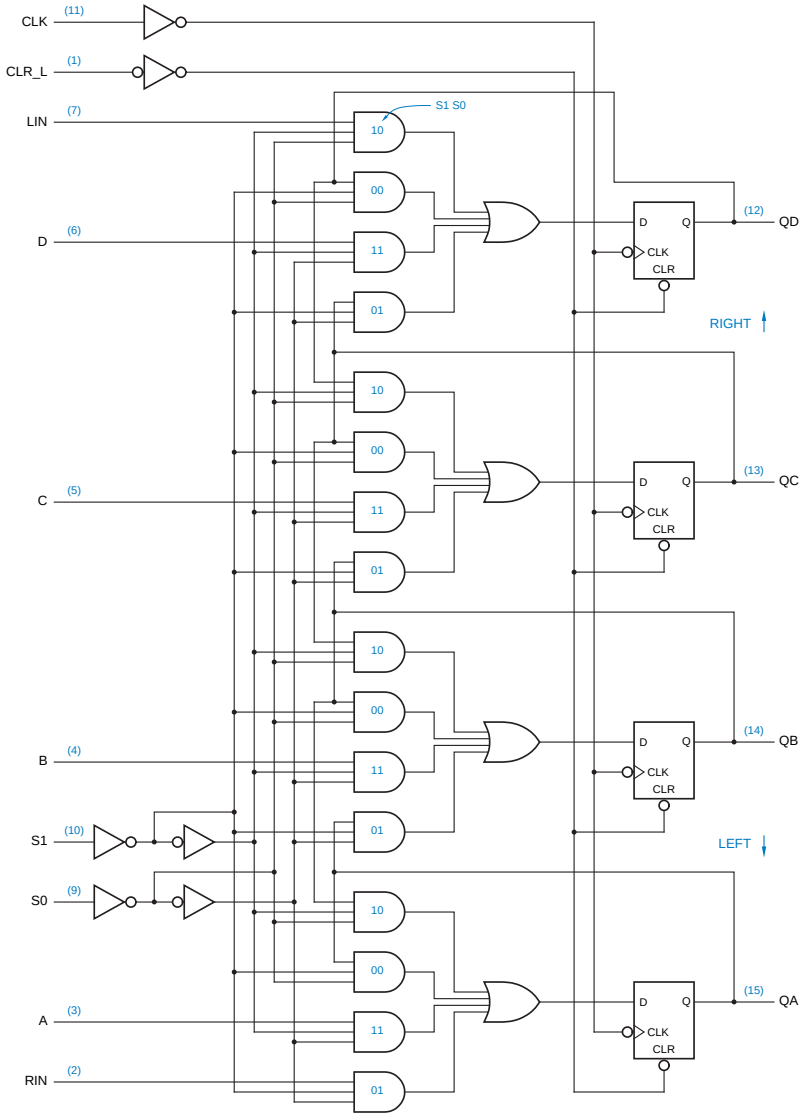

Figure 24: Logic diagram for the 74x194 4-bit universal shift register, including pin numbers for a standard 16-pin dual in-line package

Function Table for the 74x194 Universal Shift Register

Table 8-18
Function table for the 74x194 4-bit universal shift register.

	Inputs		Next state			
Function	**S1**	**S0**	**QA***	**QB***	**QC***	**QD***
Hold	0	0	QA	QB	QC	QD
Shift right	0	1	RIN	QA	QB	QC
Shift left	1	0	QB	QC	QD	LIN
Load	1	1	A	B	C	D

Figure 25: Function table for the 74x194 4-bit universal shift register

Outline

Shift Register Counters. Ring Counters

- A shift register with additional combinational logic can form a state machine whose diagram is cyclic
- Such a circuit is called a *shift-register counter*
- Unlike a binary counter, a shift register counter does not count in ascending or descending order, but it can be useful in many applications
- The simplest shift-register counter uses an n-bit shift register to obtain a counter with n states, called *ring counter*
- Figure 26 is a logic diagram for a 4-bit ring counter
- The 74x194 is wired so that it performs a left shift
- When RESET is asserted, it loads 0001
- When RESET is negated, the circuit shifts at each clock tick
- The LIN serial input is connected to the "leftmost" output (i.e., QA), so that next states are 0010, 0100, 1000, 0001, 0010, ...
- The counter cycles over four distinctive states before repeating
- A timing diagram is shown in figure 27
- In general, a ring counter visits n states in a cycle

Ring Counters

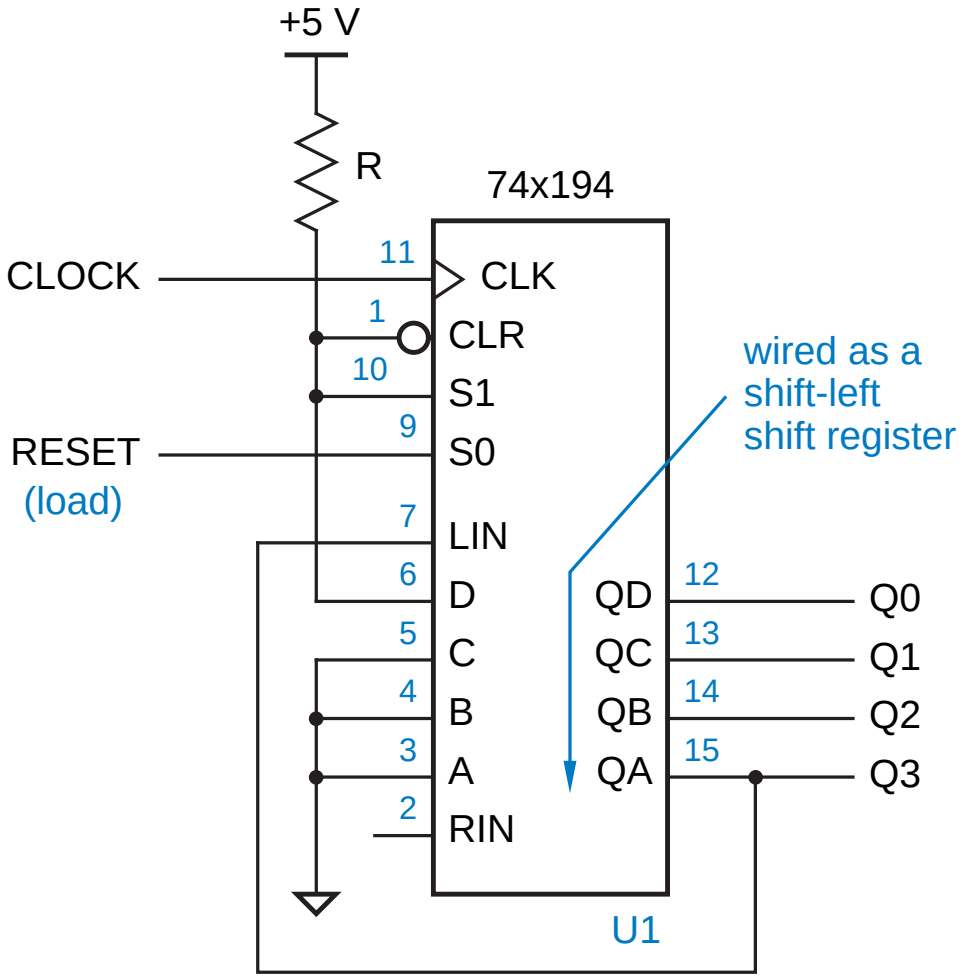

Figure 26: Simplest design for a 4-bit, 4-state ring counter with a single circulating 1

Ring Counters: A Timing Diagram

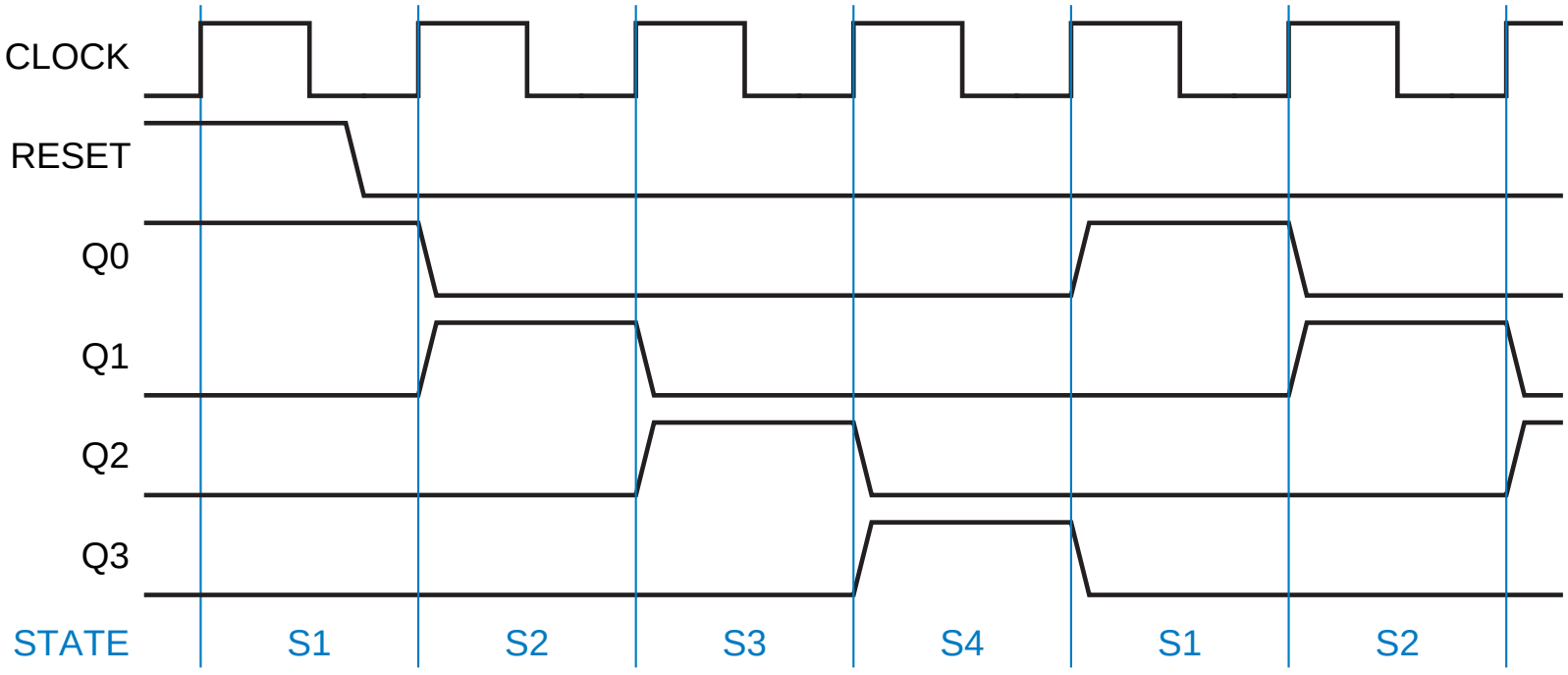

Figure 27: Timing diagram for a 4-bit ring counter

Self-Correcting Ring Counters

- The ring counter from figure 26 has a problem: it is not robust.
- The complete state diagram of this ring counter, from figure 28, illustrates this problem: if the counter enters into a state that is not part of the normal counting cycle, it can not go back to the normal counting sequence.
- The problem is solved by the self-correcting ring counter, shown in figure 29.
- This circuit shifts a 1 into LIN only when the 3 least significant bits are 0. This results in the state diagram from figure 30
- All abnormal states lead back into the normal counting cycle within 3 clock ticks and they reach the state 0001 within 4 clock ticks.
- In the general case, an n-bit self-correcting ring counter uses an $(n-1)$-input NOR gate and corrects the abnormal state within $n-1$ clock ticks.

Ring Counters: A State Diagram

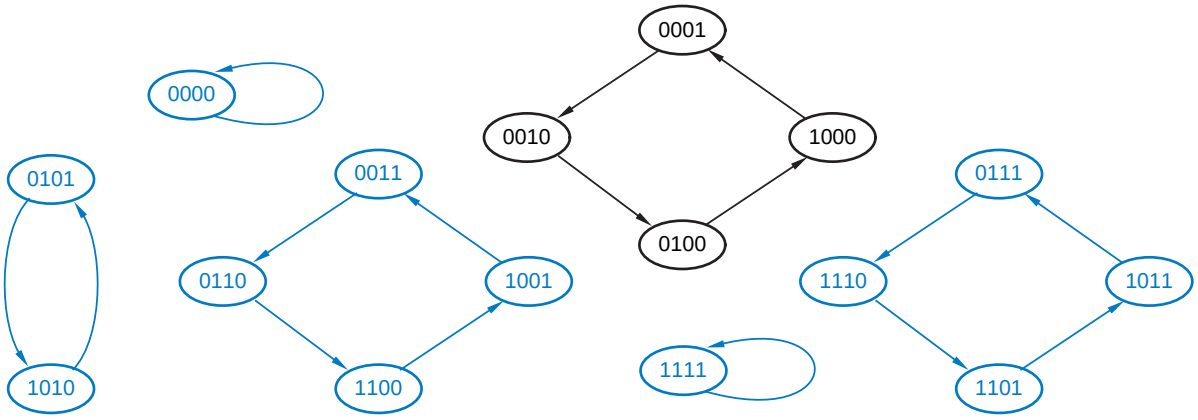

Figure 28: State diagram for a simple ring counter

Self-Correcting Ring Counter

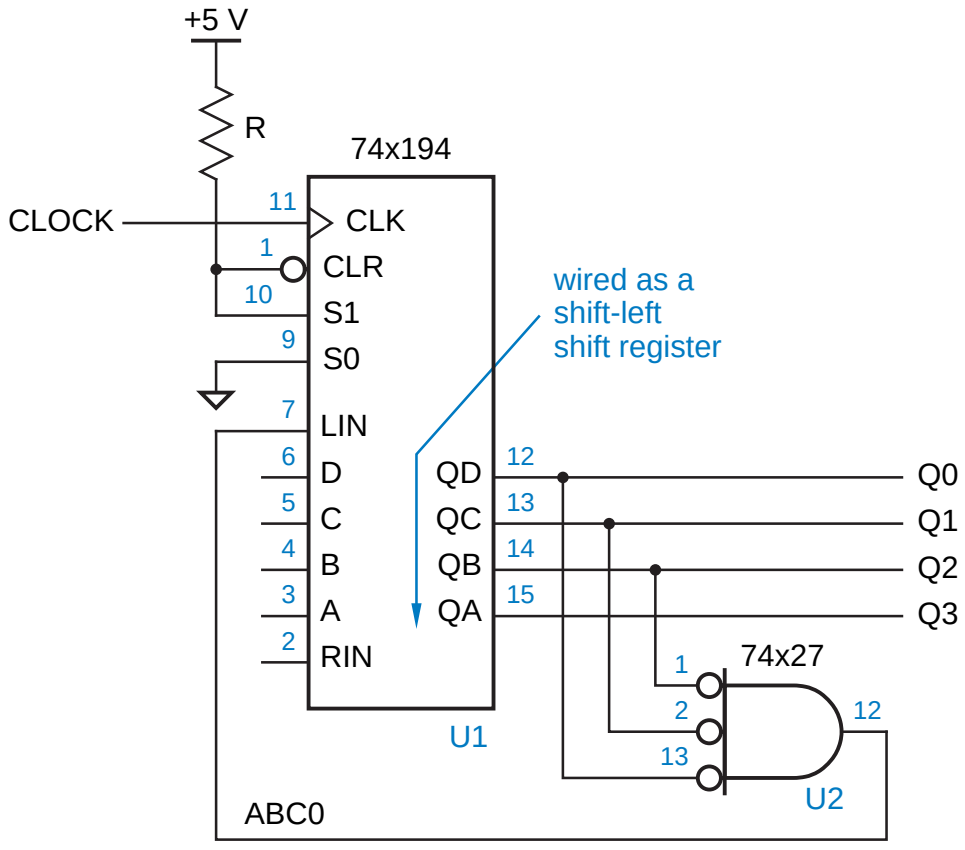

Figure 29: Self-correcting 4-bit, 4-state ring counter with a single circulating 1

State Diagram for a Self-Correcting Ring Counter

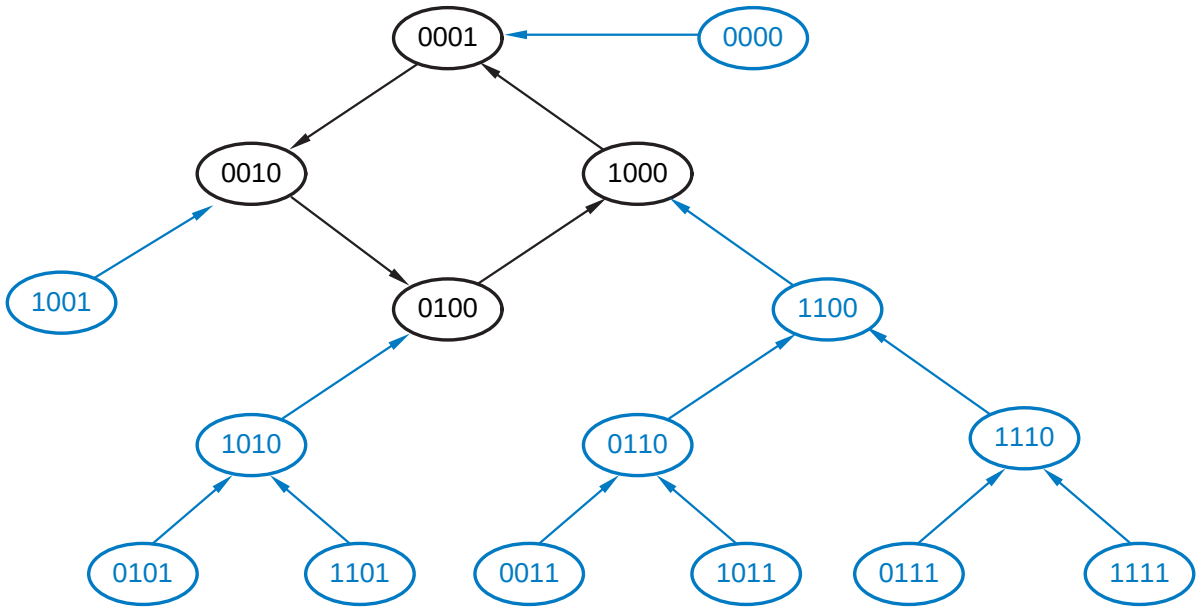

Figure 30: State diagram for a self-correcting ring counter