CHAPTER IX

REGISTER BLOCKS
COUNTERS, SHIFT, AND ROTATE REGISTERS

READ PAGES 249-275 FROM MANO AND KIME

REGISTER BLOCKS

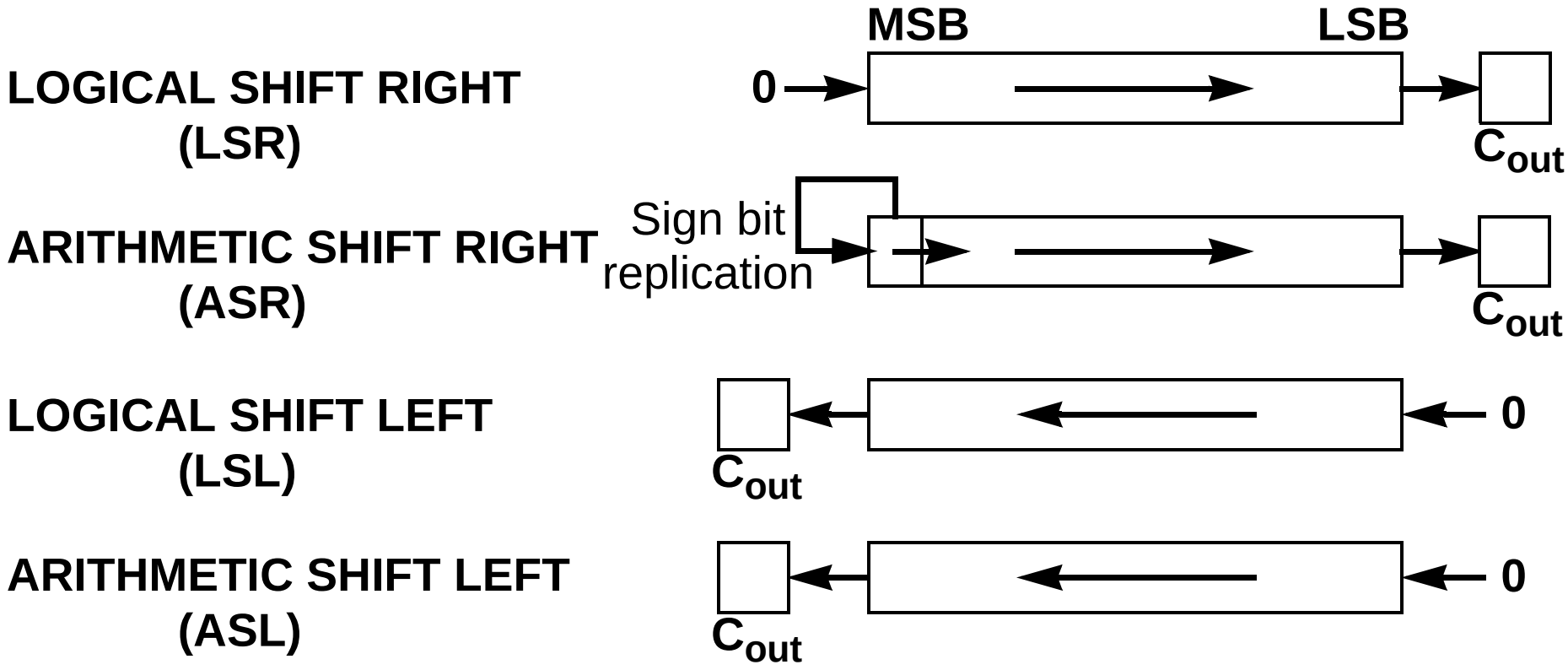
INTRODUCTION

- Like combinational building blocks, we can also develop some simple building blocks using registers. These include:
  - Shift registers
  - Rotate registers
  - Counters
- Implementations of these components can use state machines, but, it is often easier to think of them without the complication of a state machine.

SHIFT REGISTERS

INTRODUCTION

- Logical shift registers take the bits stored and move them up a significant bit or down a significant bit.

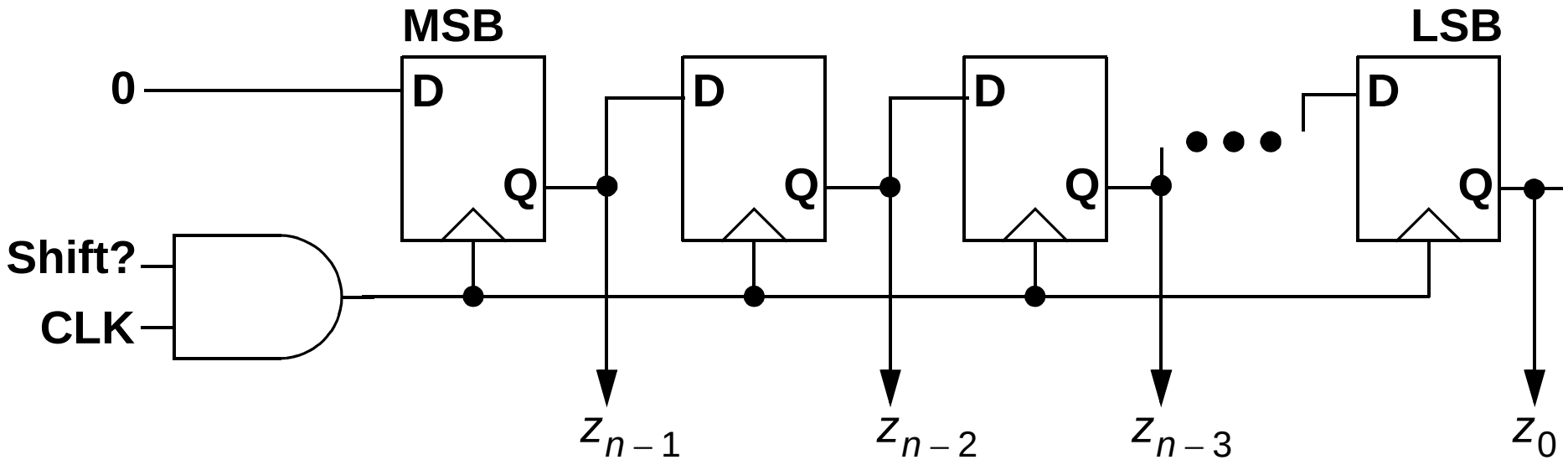


- Notice that logical and arithmetic shift lefts are the same.

SHIFT REGISTERS

LSR SAMPLE

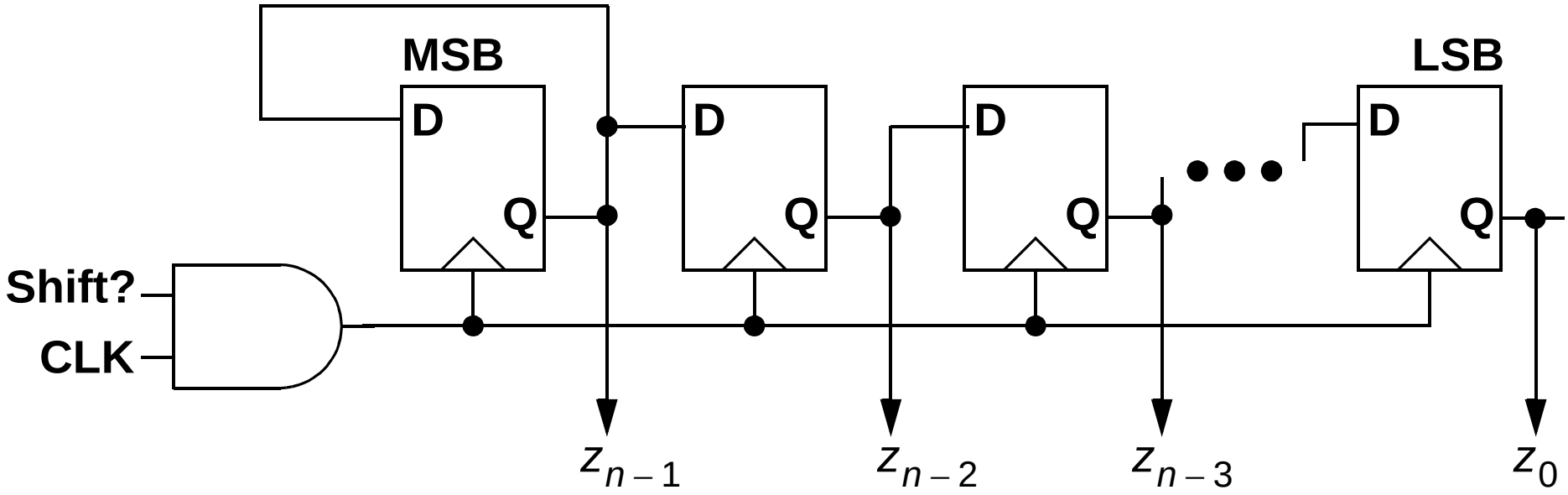
- A simple implementation of a logical right shift register might look like the following.



SHIFT REGISTERS

ASR SAMPLE

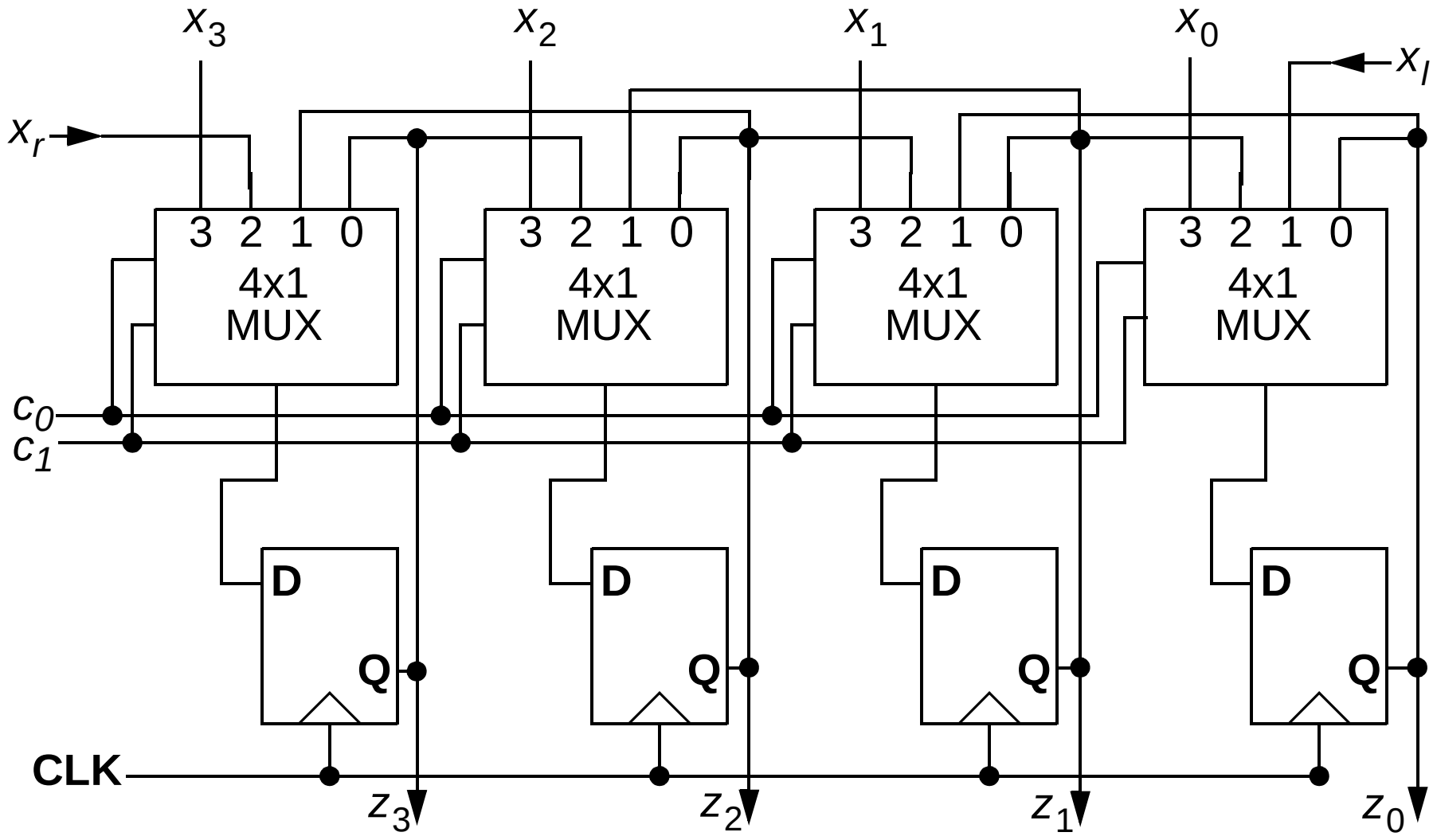
- An arithmetic right shift register might look like the following.



SHIFT REGISTERS

4-BIT BIDIRECTIONAL

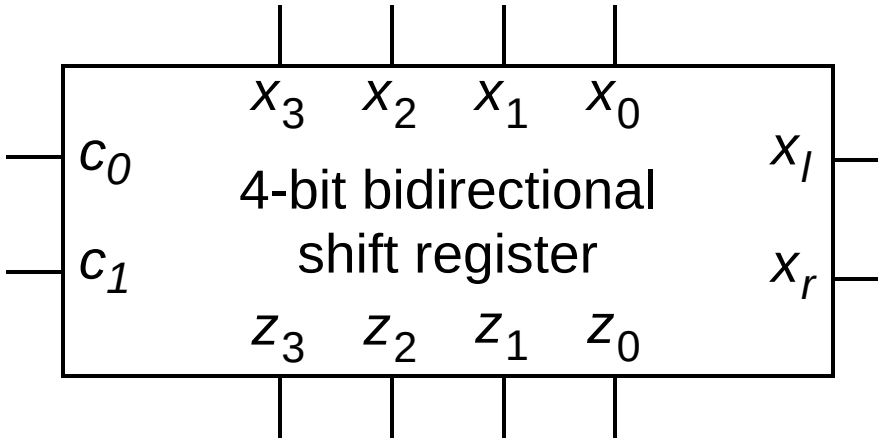
- The following is a 4-bit bidirectional shift register with parallel load.



SHIFT REGISTERS

CASCADING (1)

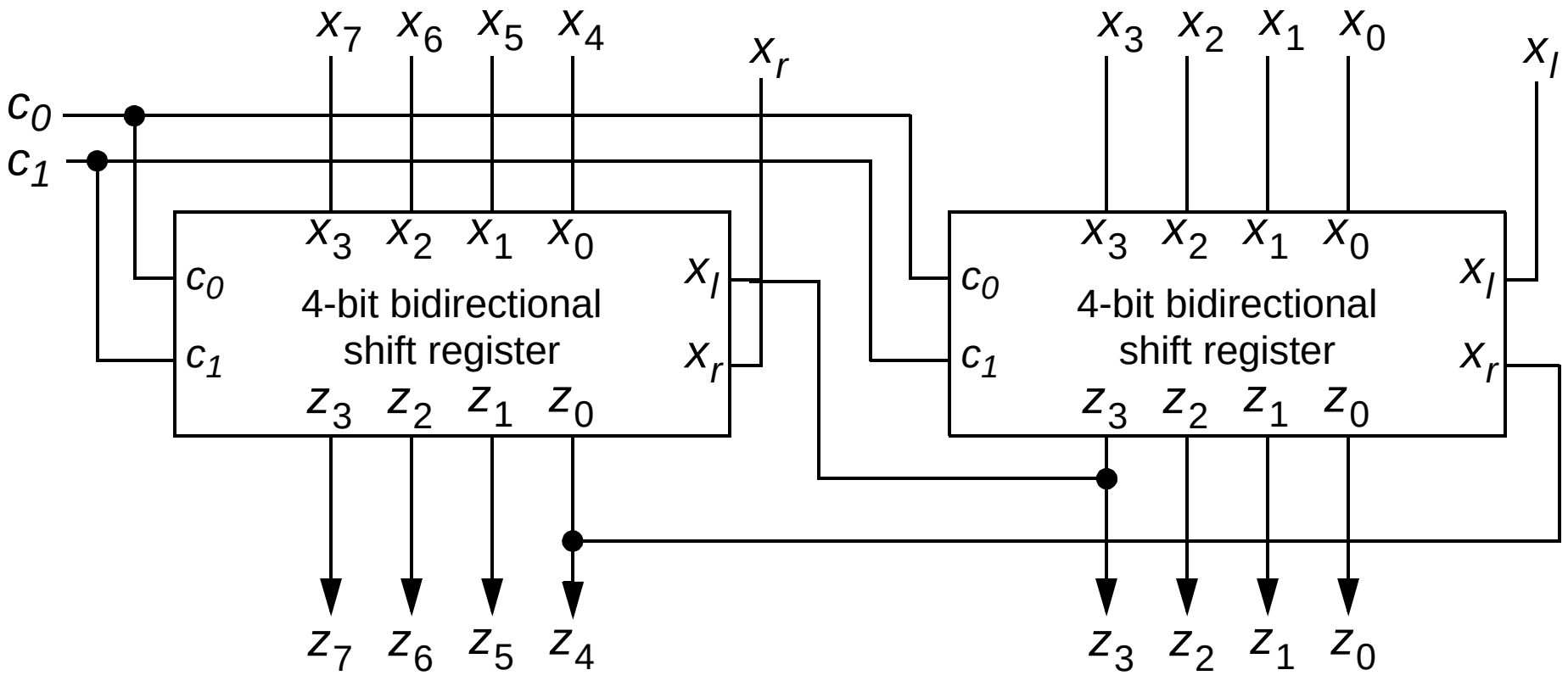
- Cascading of shift registers can also be done if the discarded bit is used to shift into another shift register module.
- For instance, the 4-bit bidirectional shift register previously presented can be easily cascaded using the
  - x_r (right shift data input) and
  - x_l (left shift data input)



SHIFT REGISTERS

CASCADING (2)

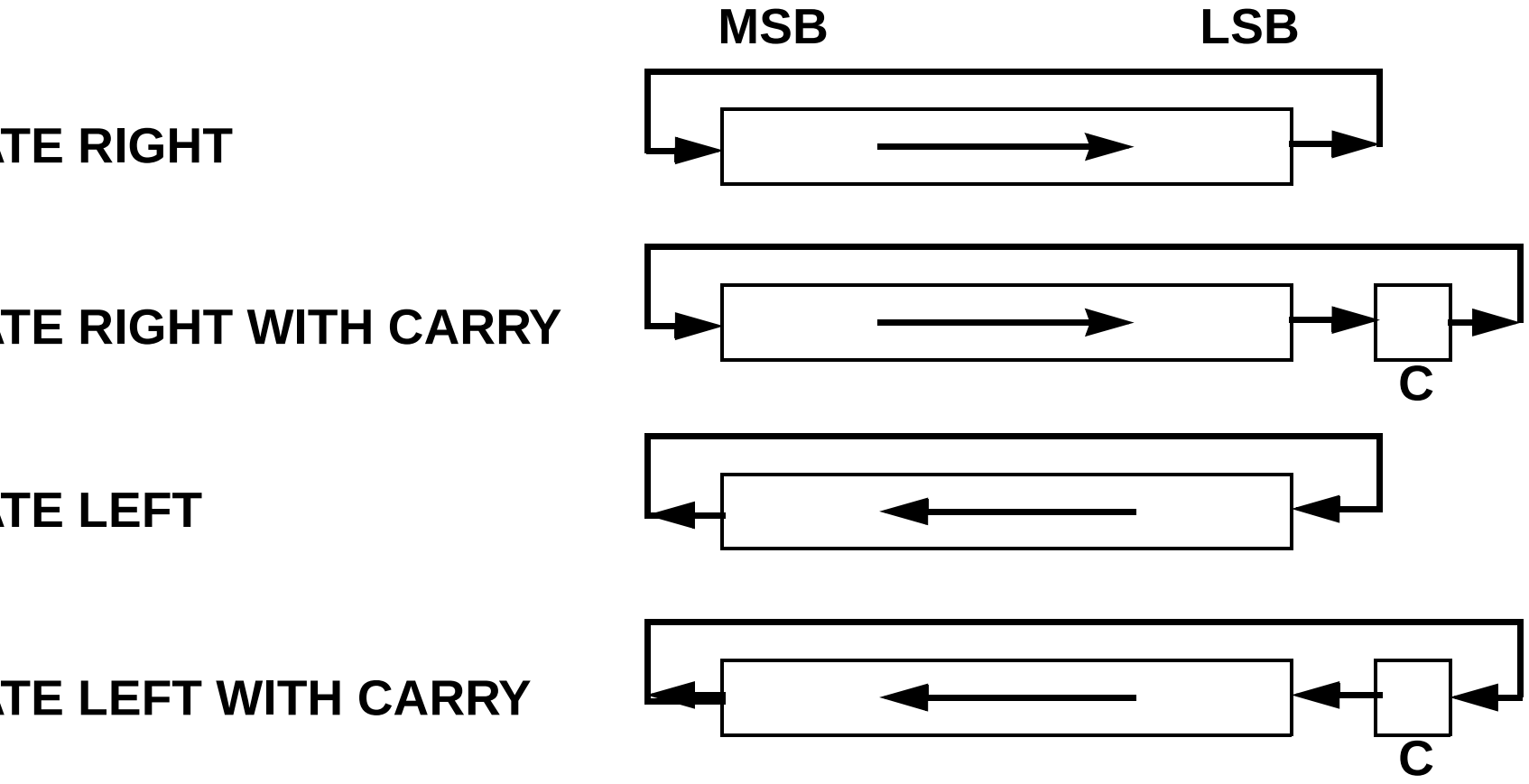
- For example, an 8-bit bidirectional shift register with parallel load can be formed as follows.



ROTATE REGISTERS

INTRODUCTION

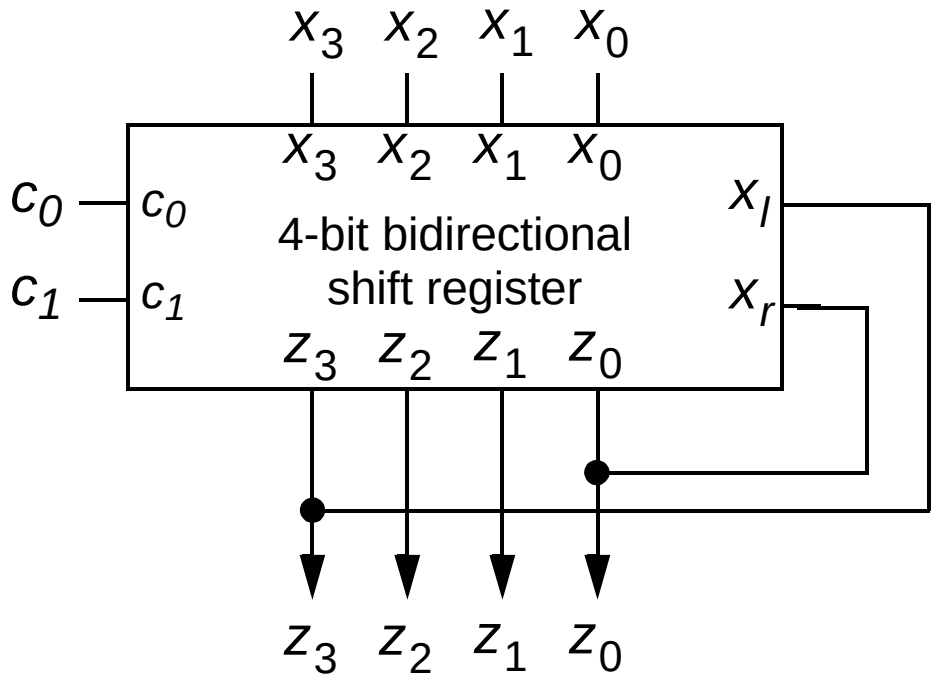
- A rotate register is the same as a logical shift register except that the discarded bit is fed back into the empty space from the shift.



ROTATE REGISTERS

USING SHIFT REGISTERS

- Rotate registers can actually be implemented using shift registers that have serial data inputs (such as the 4-bit bidirectional shift register discussed).
- For example, a 4-bit rotate register can be formed as follows.



COUNTERS

INTRODUCTION

- A counter is a register that on each clock pulse counts up or down, usually in binary.
- Types of counters
  - ripple counters
  - synchronous counters
  - binary counters
  - BCD counters
  - Gray-code counters
  - Ring counters (a 1 moves in a ring from one flip-flop to the next)
  - up/down counters (ability to increment or decrement)
  - counters with a parallel load (load in starting value with parallel input)

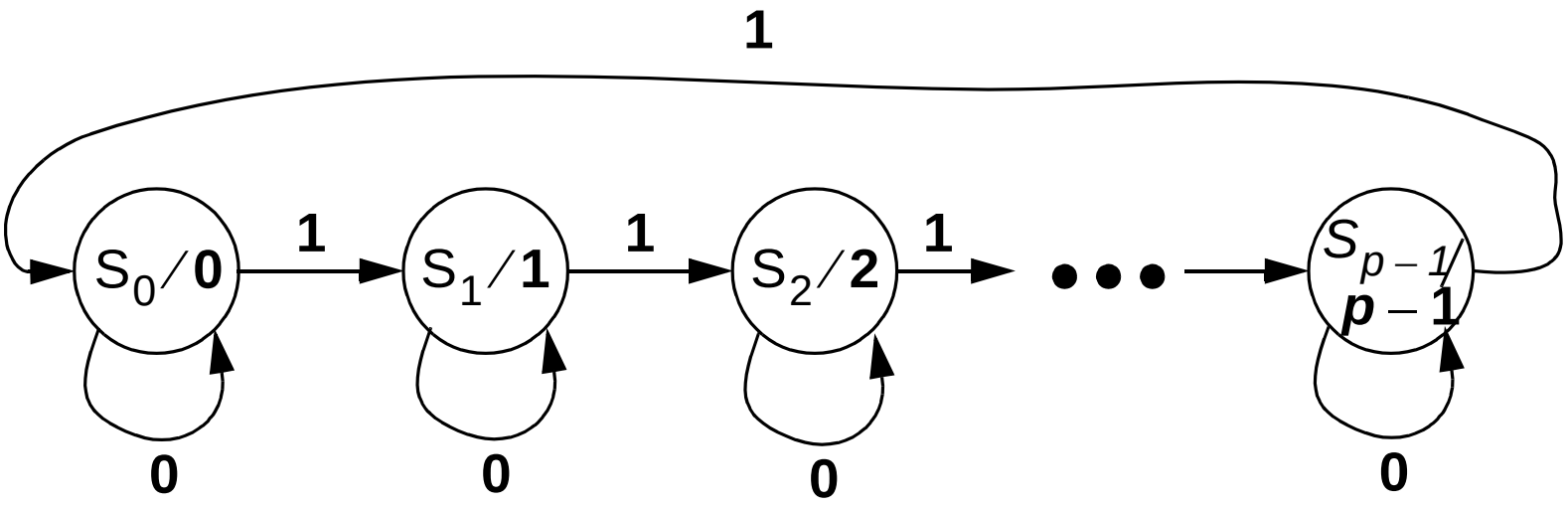
COUNTERS

MODULO-*P* COUNTERS

- A modulo-*p* counter is defined by the following equation.

$$S(t+1) = (S(t) + x) \ mod \ p$$

- The state diagram for the modulo-*p* counter is as follows.



COUNTERS

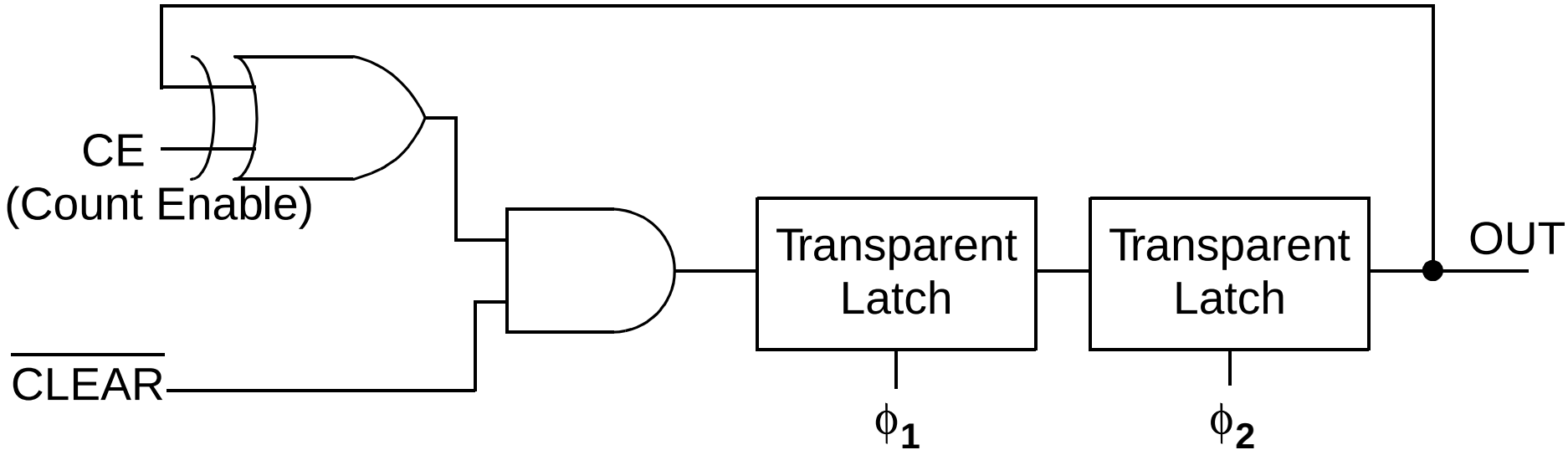
RIPPLE AND SYNCHRONOUS

- An *n*-bit binary counter consists of *n* flip-flops and can count in binary from 0 through $2^n - 1$.
  - This can be formed with a modulo-*p* counter where $p = 2^n$.
- Two main categories exist for counters:
  - Ripple counters
    - One flip-flop transition serves to trigger other flip-flops.
    - The clock pulse is usually only sent to the first flip-flop.
    - This requires a memory cell that can complement its value.
    - The JK flip-flip would be one approach (we have not studied this!)
  - Synchronous counters
    - Change of state is determined from the present state.
    - Clock pulse sent to all flip-flops.

COUNTERS

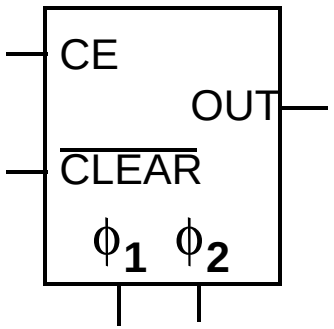
TOGGLE CELL

- A toggle cell will be useful for implementing counters.



Present Latch Value	CE	$\overline{\text{CLEAR}}$	Next Latch Value	OUT
X	**X**	**0**	**0**	**?**
0	**0**	**1**	**0**	**0**
1	**0**	**1**	**1**	**1**
0	**1**	**1**	**1**	**0**
1	**1**	**1**	**0**	**1**

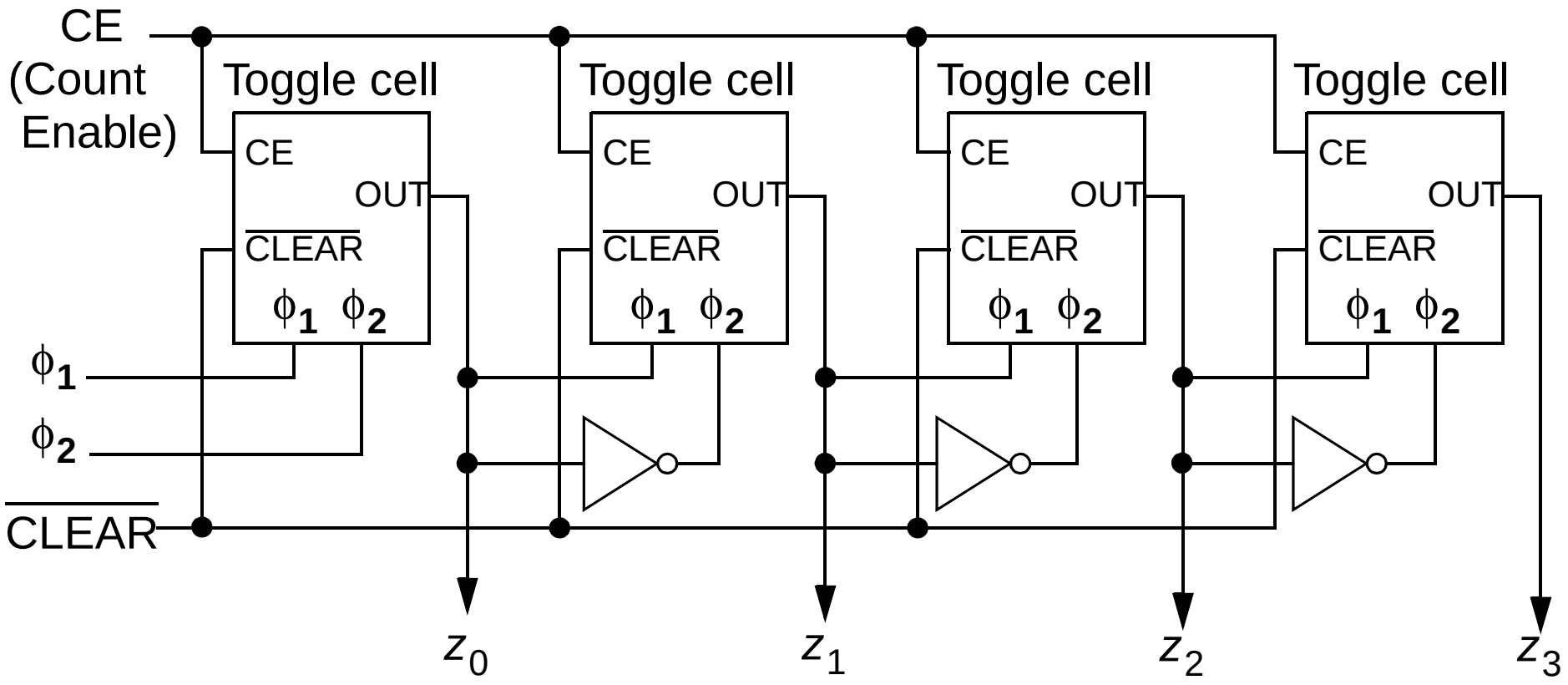
Toggle cell



COUNTERS

RIPPLE COUNTER

- The toggle cell can be used as follows to form a ripple counter.

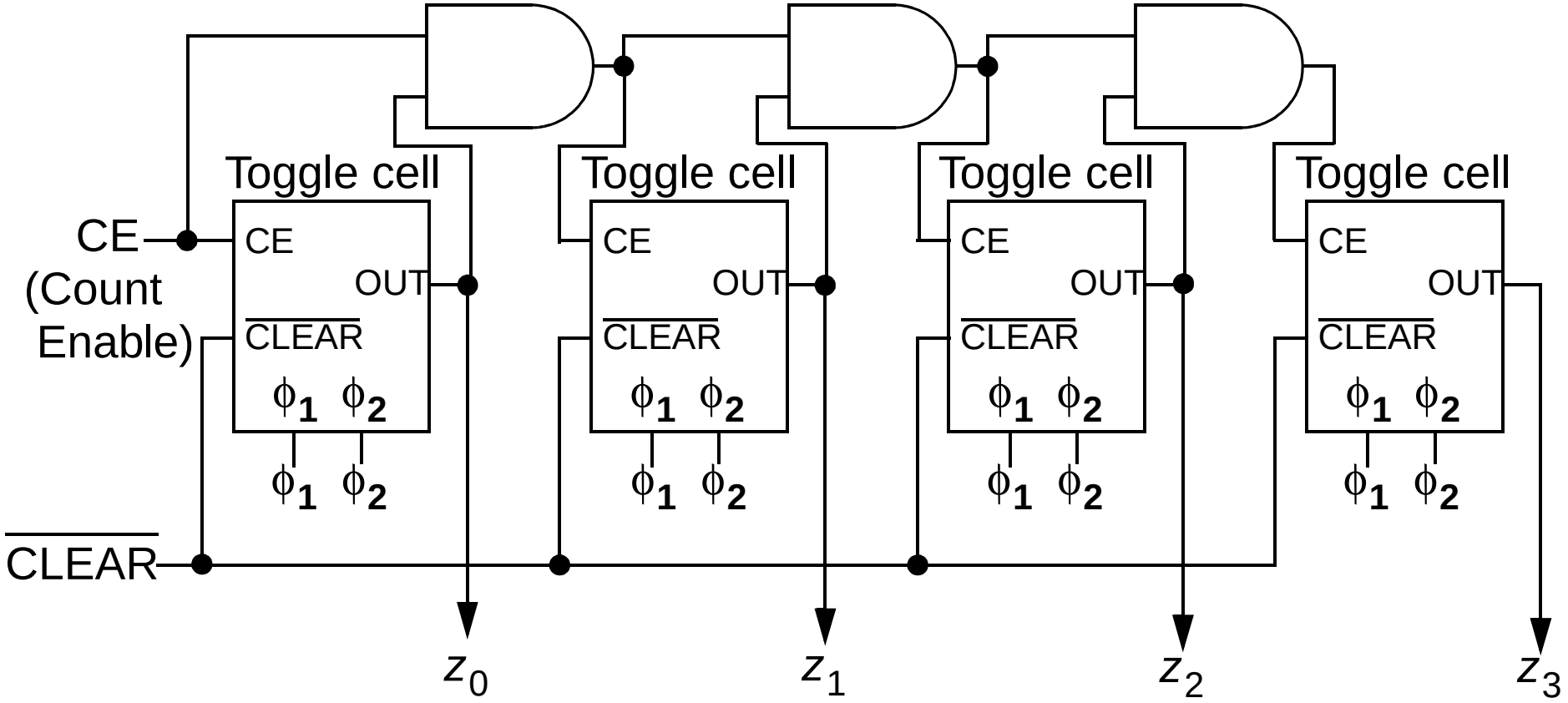


- Notice that the previous toggle cell is connected to the clock input of the next cell. This causes the bits to ripple through the counter.

COUNTERS

SYNCHRONOUS COUNTER

- Below is an example 4-bit synchronous counter using toggle cells.

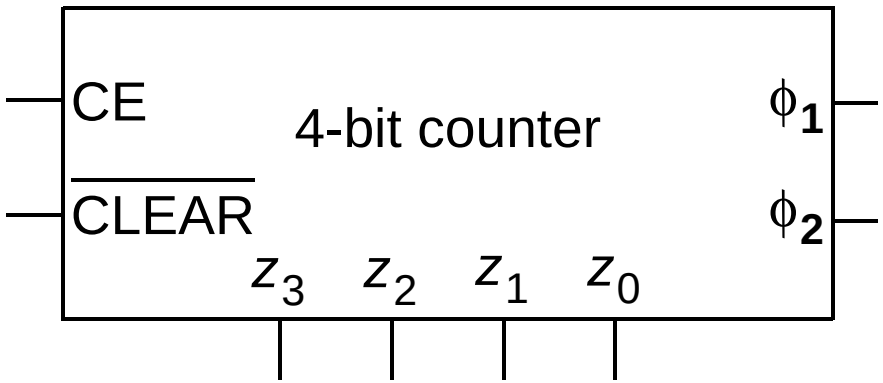


- Notice that clock is sent to all toggle cells.
- A simplified form is in Figure 5-11, pp. 269 of Mano & Kime.

COUNTERS

MORE ON MODULO-*P*

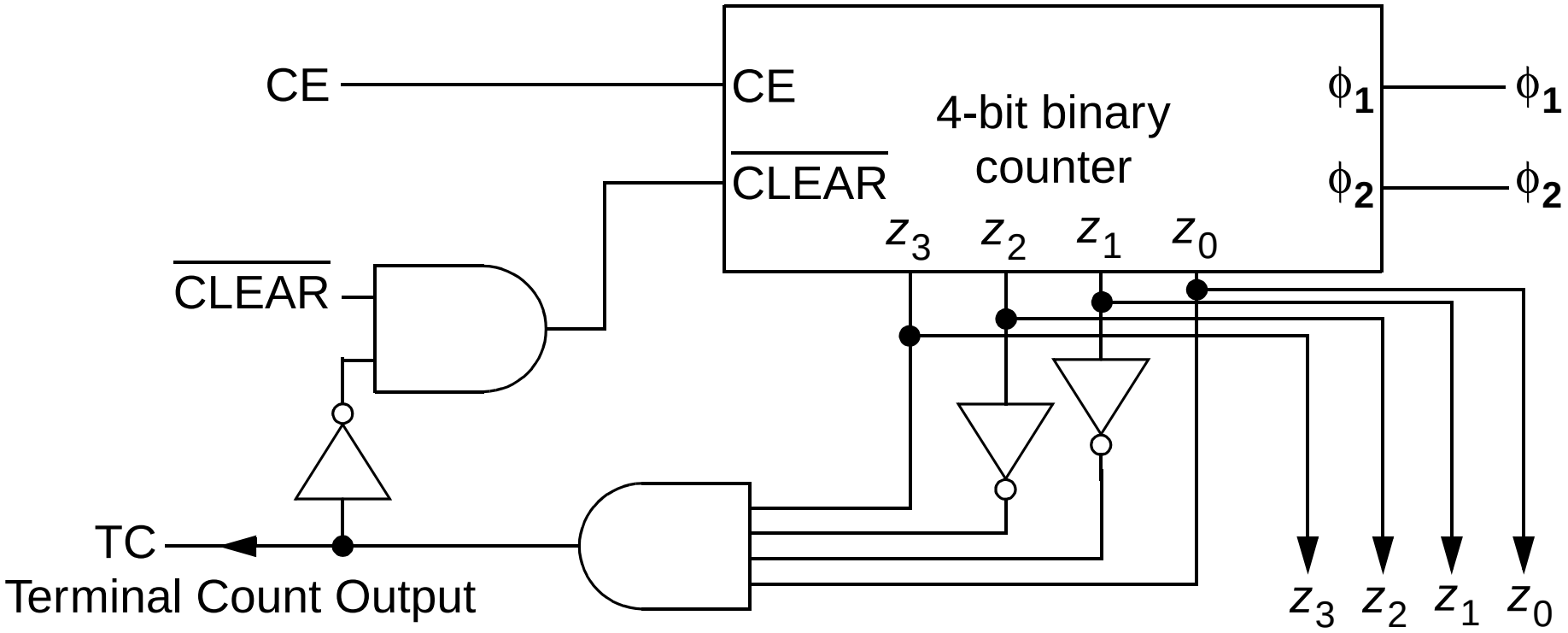
- Notice that the counters developed so far can count from 0 to $2^n - 1$ for n toggle cells.
  - Therefore, for module-p counting, the p is currently limited to 2^n.
- How about if we wish p to be a non-power of 2?
  - Need to build what can be referred to as a divide by counter.
  - Given the following counter block, a general modulo-p counter can be constructed by clearing the counter after the desired maximum value.



COUNTERS

BCD COUNTER (MODULO-10)

- To illustrate general modulo-*p* counters, consider the following implementation of a single digit decimal counter using BCD.

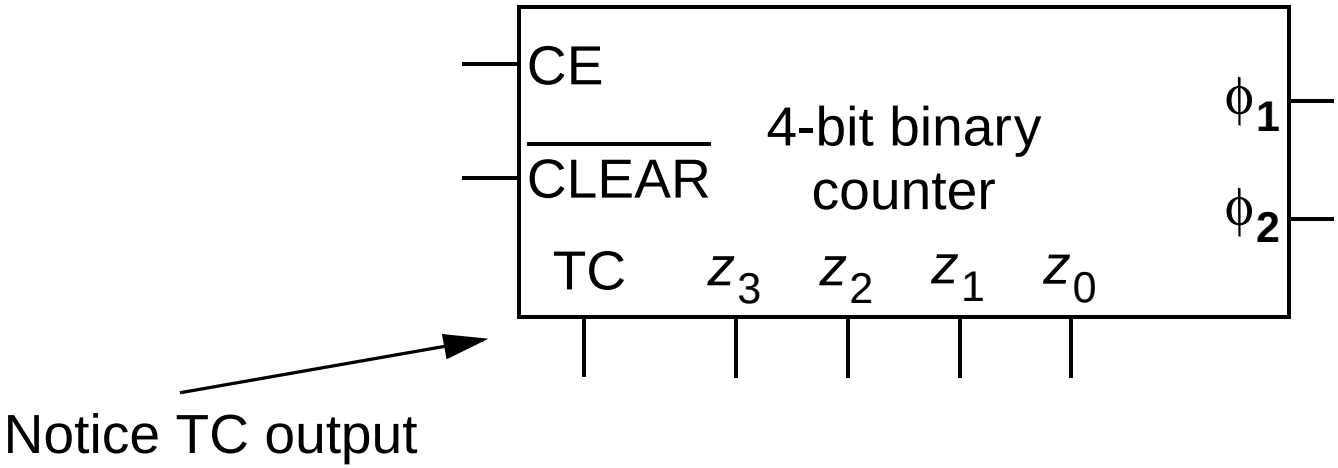


- Notice that the counter is cleared after a value of 9 (**1001**).

COUNTERS

TERMINAL COUNT (TC)

- The previous BCD counter was built by deriving a terminal count (TC) output signal.
- A terminal count output signal for any counter can be useful, so, we will be included in general block diagram for a binary counter.



- In this 4-bit binary counter example, TC=**1** only when the output is **1111**.

COUNTERS

CASCADING COUNTERS

- With a terminal count output (TC), counters can be easily cascaded together to form larger counters.
  - For instance, an 8-bit binary counter can be formed as follows.