Chapter 11

CLOCKED SEQUENTIAL CIRCUITS

State Machines

- A synchronous sequential circuit, consisting of a sequential logic and a combinational logic section, whose outputs and internal flip-flops progress through a predictable sequence of states in response to a clock and other input signals.

Example: Single-pulse generator

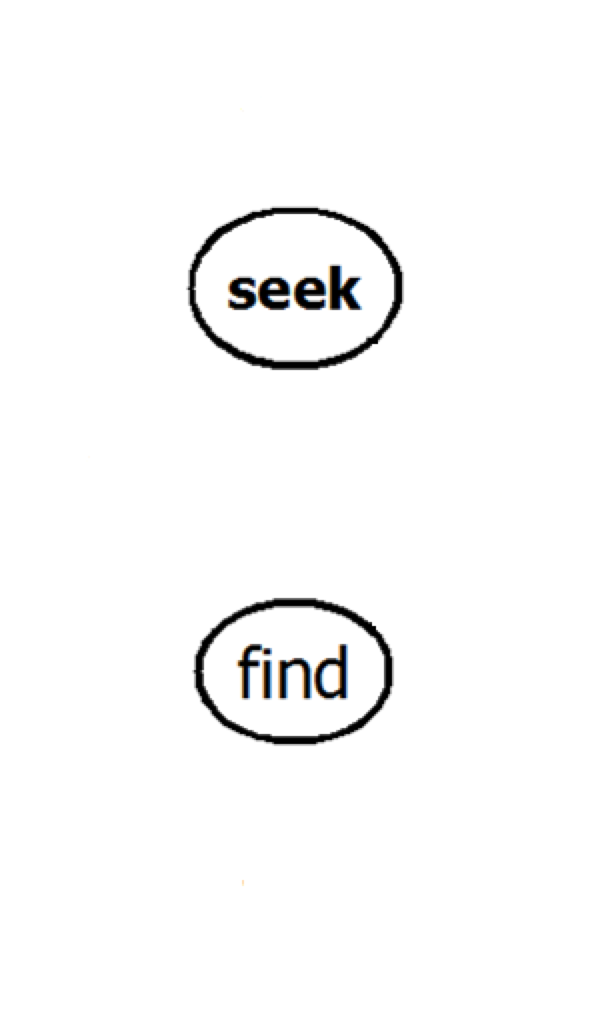
- Design a single-pulse generator based on the given conditions. Use D flip-flops for the state logic.

Example: Single-pulse generator

- The state machine operates as follows:
    - The circuit has two states: *seek* and *find*, an input called *sync* and an output called *pulse*.

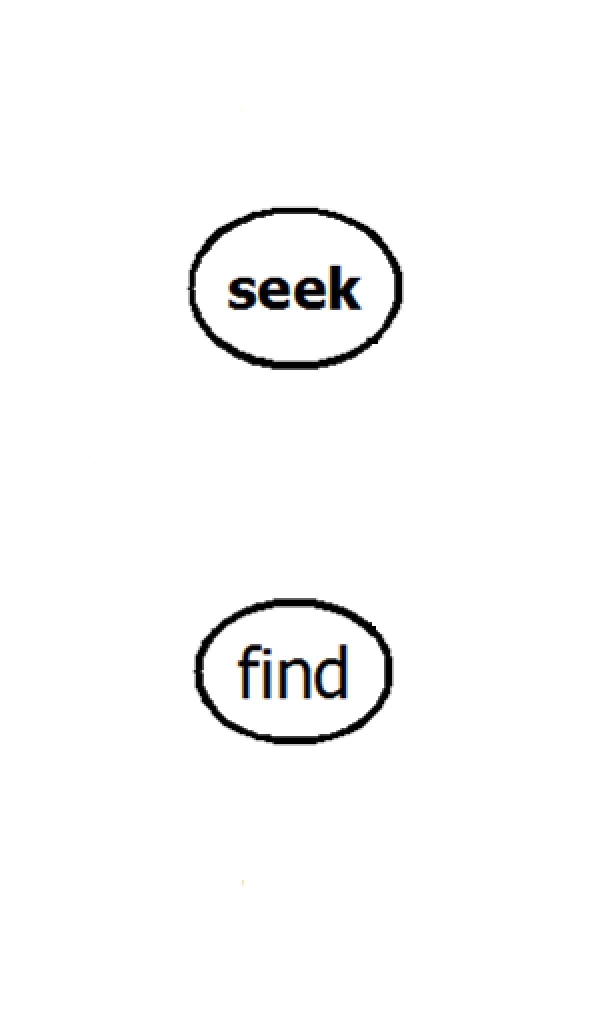
Example: Single-pulse generator

- The state machine operates as follows:
  - The circuit has two states: *seek* and *find*, an input called *sync* and an output called *pulse*.



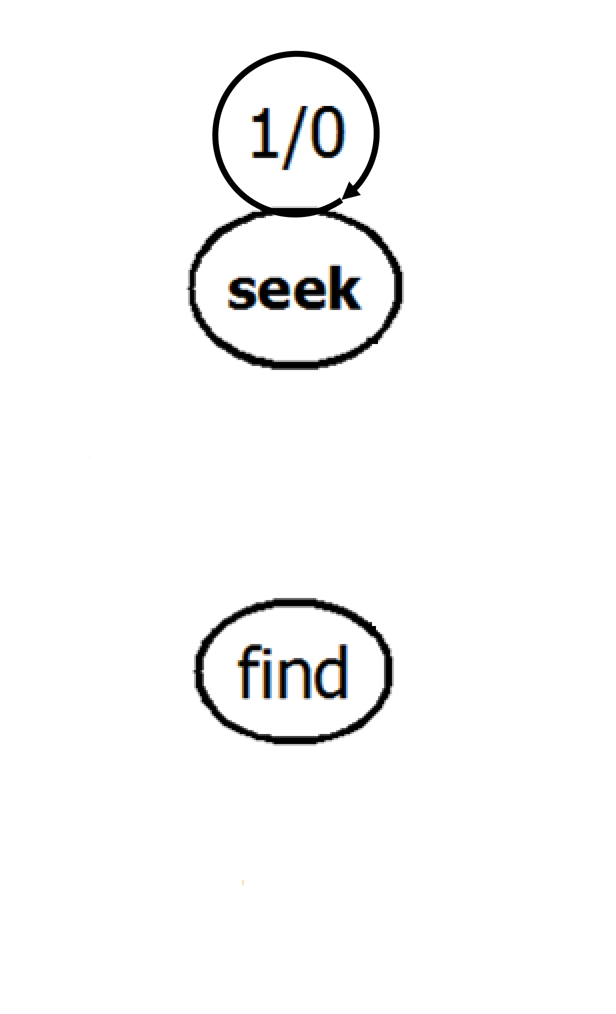
Example: Single-pulse generator

- The state machine operates as follows:
  - The state machine resets to the state *seek*. If *sync* = 1, the machine remains in *seek* and the output, *pulse*, remains LOW.



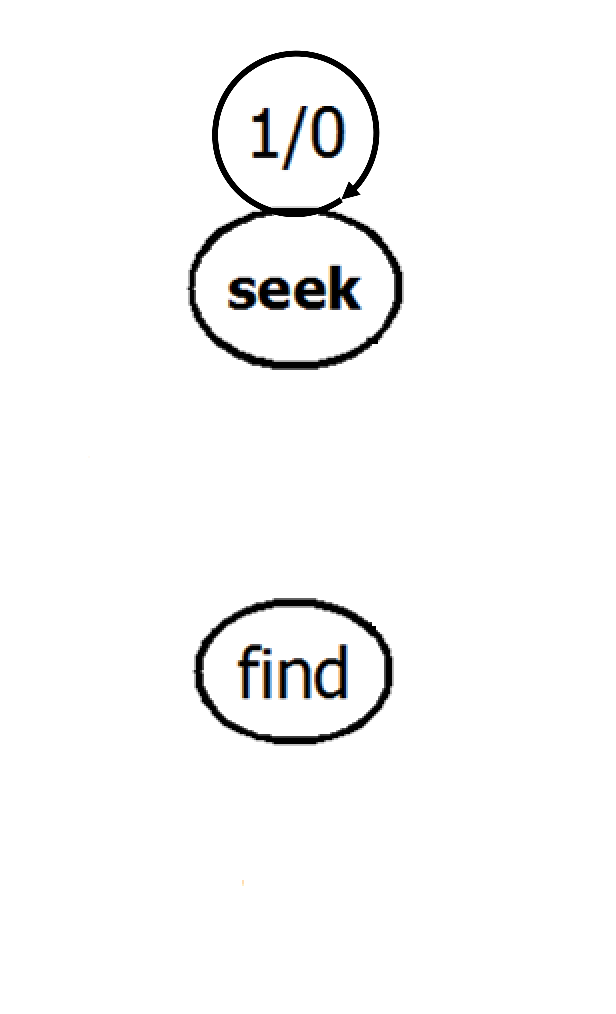
Example: Single-pulse generator

- The state machine operates as follows:
  - The state machine resets to the state *seek*. If *sync* = 1, the machine remains in *seek* and the output, *pulse*, remains LOW.



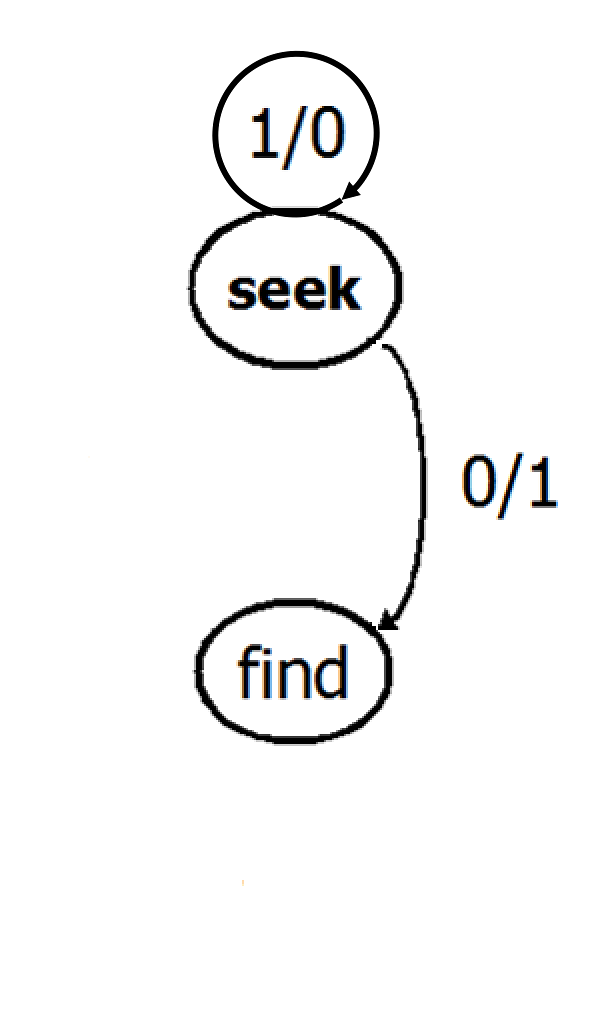
Example: Single-pulse generator

- The state machine operates as follows:
  - When *sync* = 0, the machine makes a transition to *find*. In this transition, *pulse* goes HIGH.



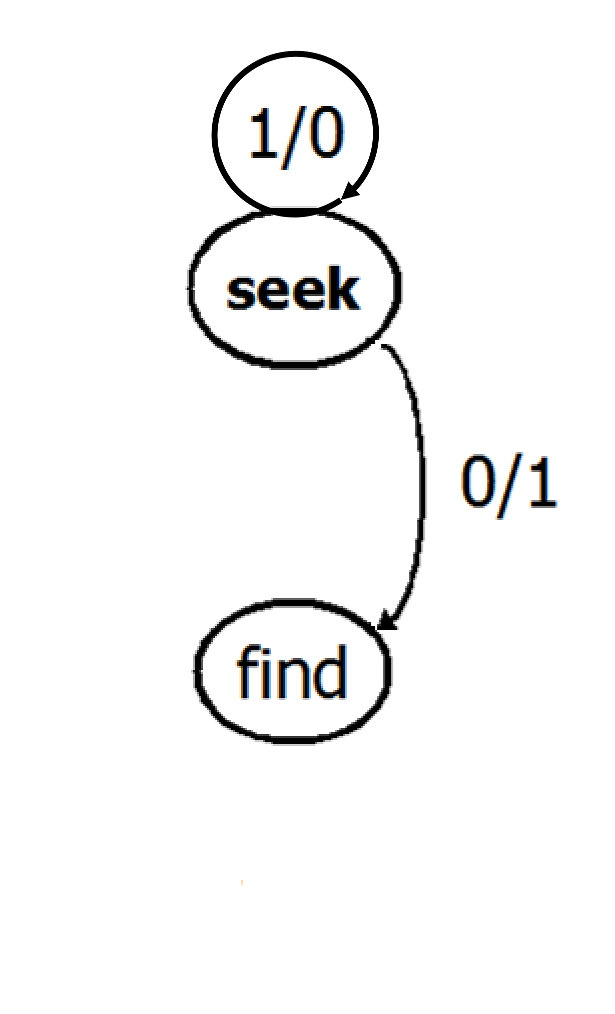
Example: Single-pulse generator

- The state machine operates as follows:
  - When *sync* = 0, the machine makes a transition to *find*. In this transition, *pulse* goes HIGH.



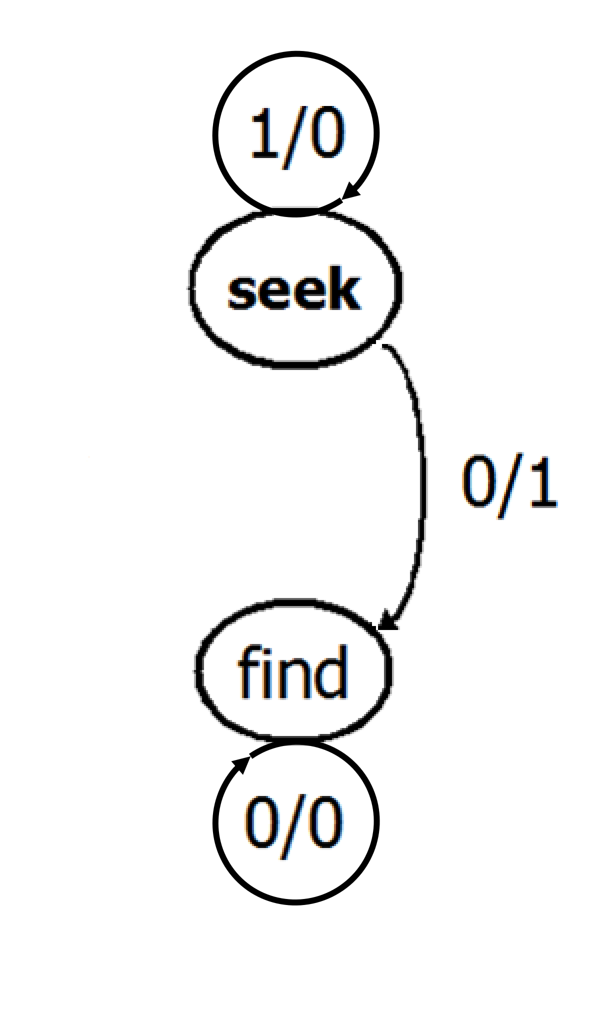
Example: Single-pulse generator

- The state machine operates as follows:
  - When the machine is in state *find* and *sync* = 0, the machine remains in *find* and *pulse* goes LOW.



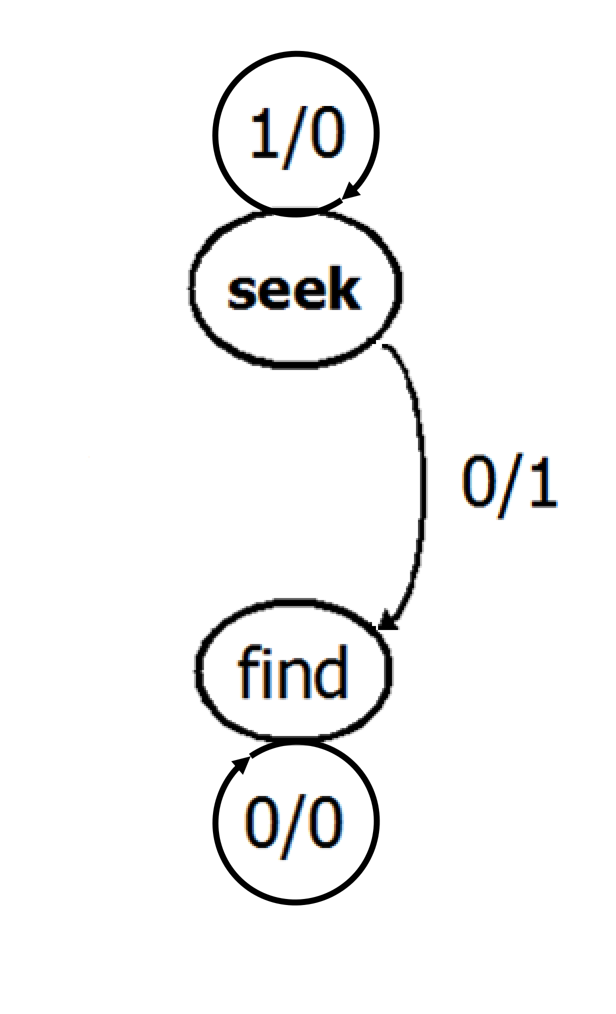
Example: Single-pulse generator

- The state machine operates as follows:
  - When the machine is in state *find* and *sync* = 0, the machine remains in *find* and *pulse* goes LOW.



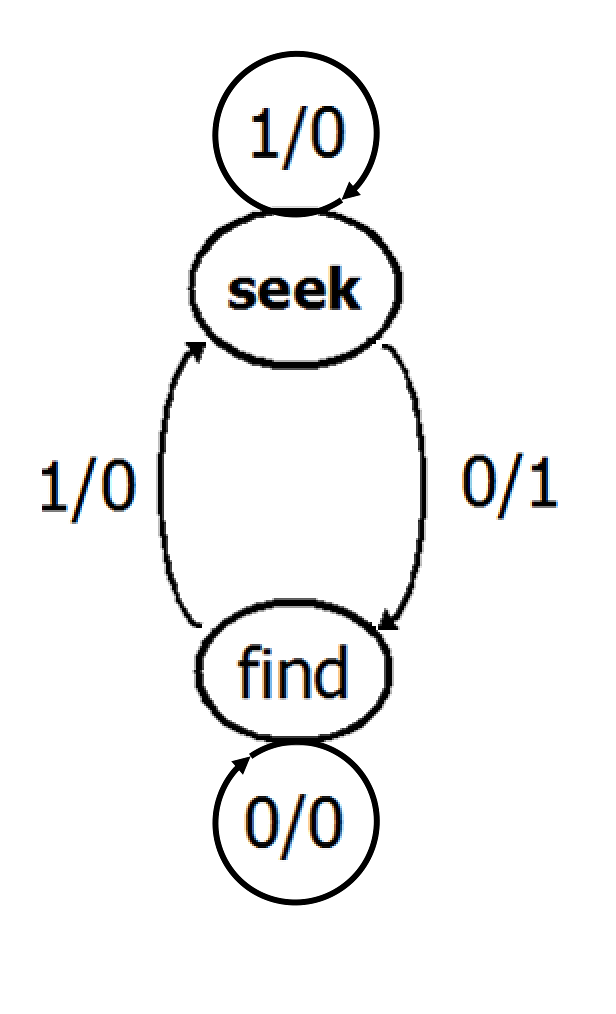
Example: Single-pulse generator

- The state machine operates as follows:
  - When the machine is in *find* and *sync* = 1, the machine goes back to *seek* and *pulse* remains LOW.



Example: Single-pulse generator

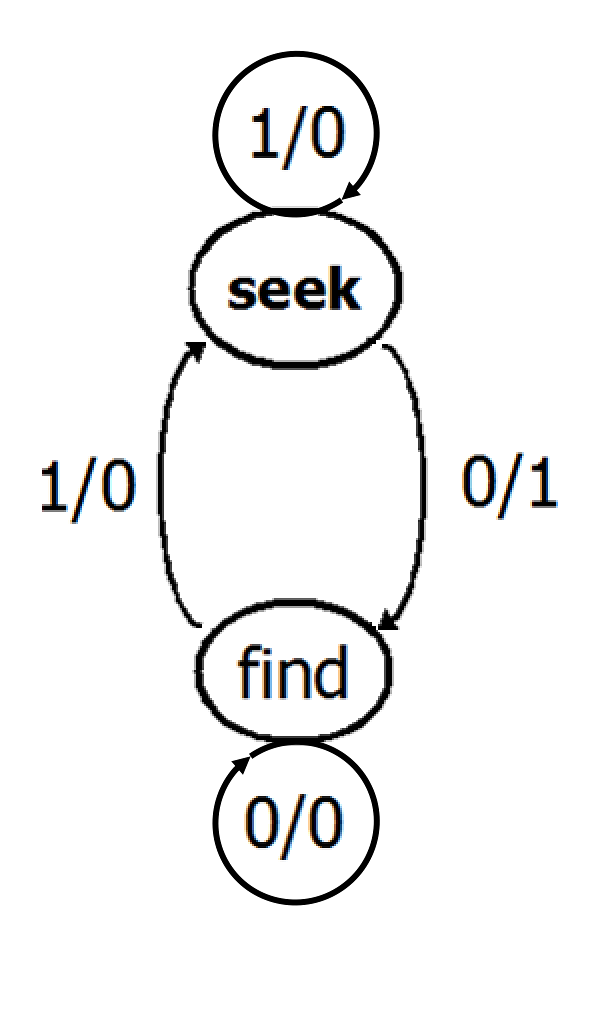
- The state machine operates as follows:
  - When the machine is in *find* and *sync* = 1, the machine goes back to *seek* and *pulse* remains LOW.



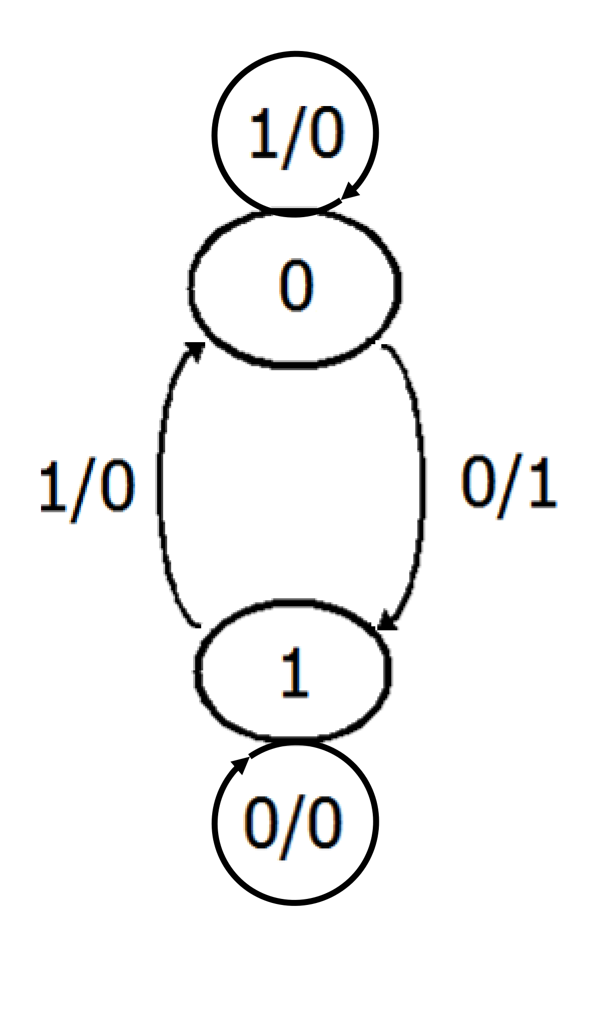
Example: Single-pulse generator

- Design a single-pulse generator based on the given conditions. Use D flip-flops for the state logic.

- Seek = 0
- Find = 1



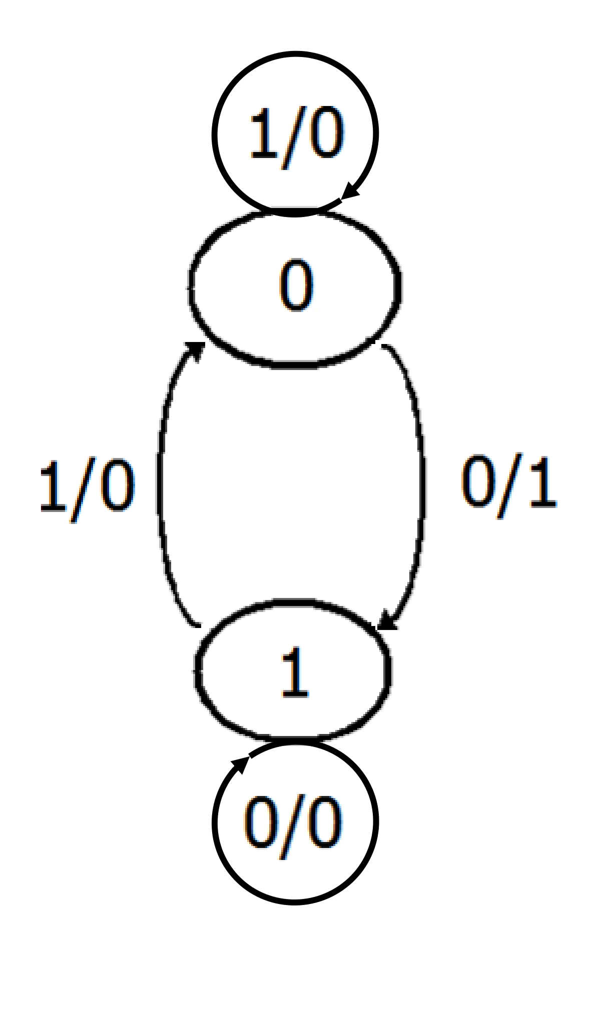
Example: Single-pulse generator



State Table

PS	Input	NS	Output
A	sync	A	pulse
0	0		
0	1		
1	0		
1	1		

Example: Single-pulse generator



State Table

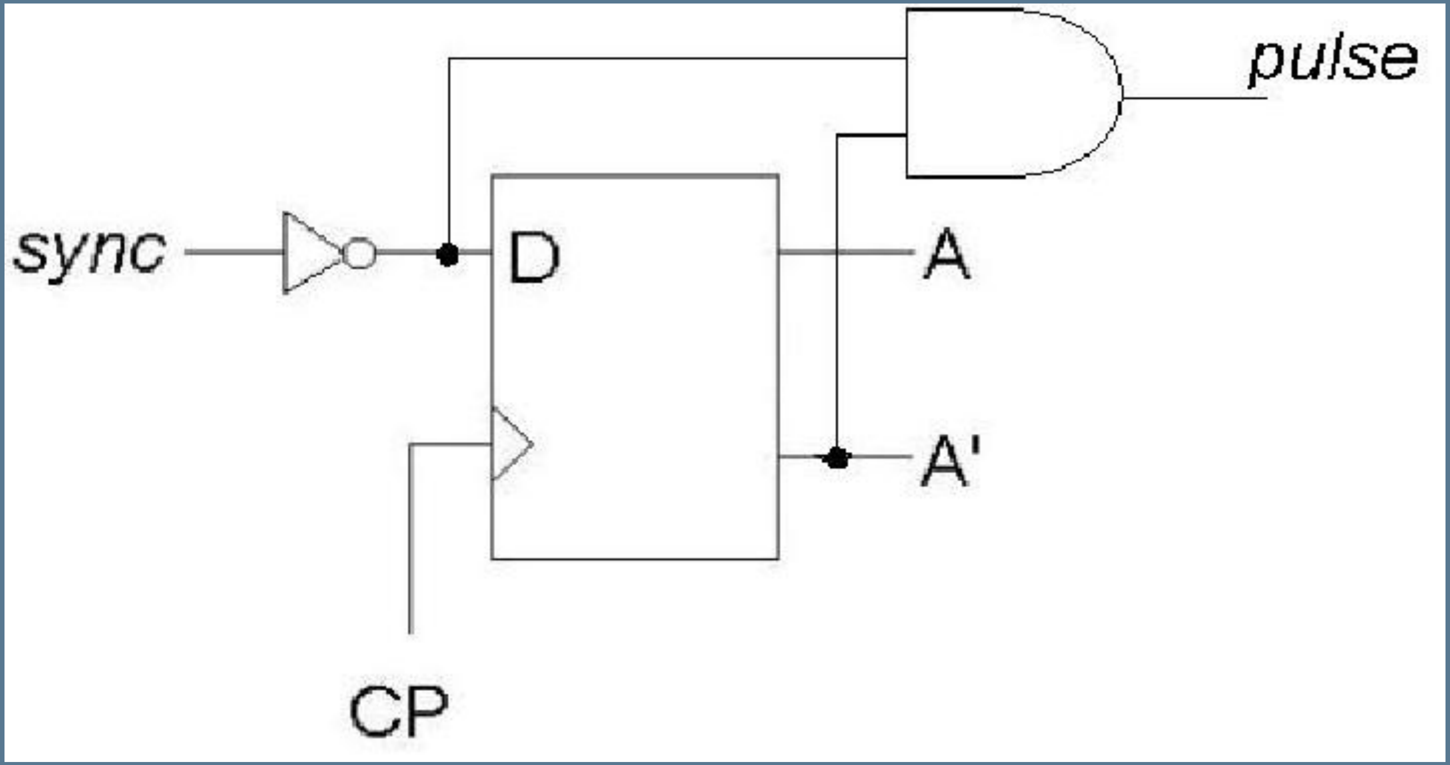
PS	Input	NS	Output
A	sync	A	pulse
0	0	1	1
0	1	0	0
1	0	1	0
1	1	0	0

Example: Single-pulse generator

- DA = sync’
- pulse = A’ sync′

PS	Input	NS	Output
A	sync	A	pulse
0	0	1	1
0	1	0	0
1	0	1	0
1	1	0	0

Example: Single-pulse generator



Single-pulse generator

Registers

- Registers consist of an arrangement of flip-flops and are important in applications involving the storage and data transfer in a digital system.

- The basic difference between a register and a counter is that a register has no specified sequence of states, except in certain very specialized applications.

Basic Register Functions

- Data movement
  - The shifting capability of a register permits the movement of data from stage to stage within the register or into or out of the register.

- Data storage
  - The storage capacity of a register is the number of bits (0s and 1s) of digital data it can retain.

Types of Registers

- Serial-Parallel Registers
- Shift Registers
- Rotate Registers

Serial-Parallel Registers

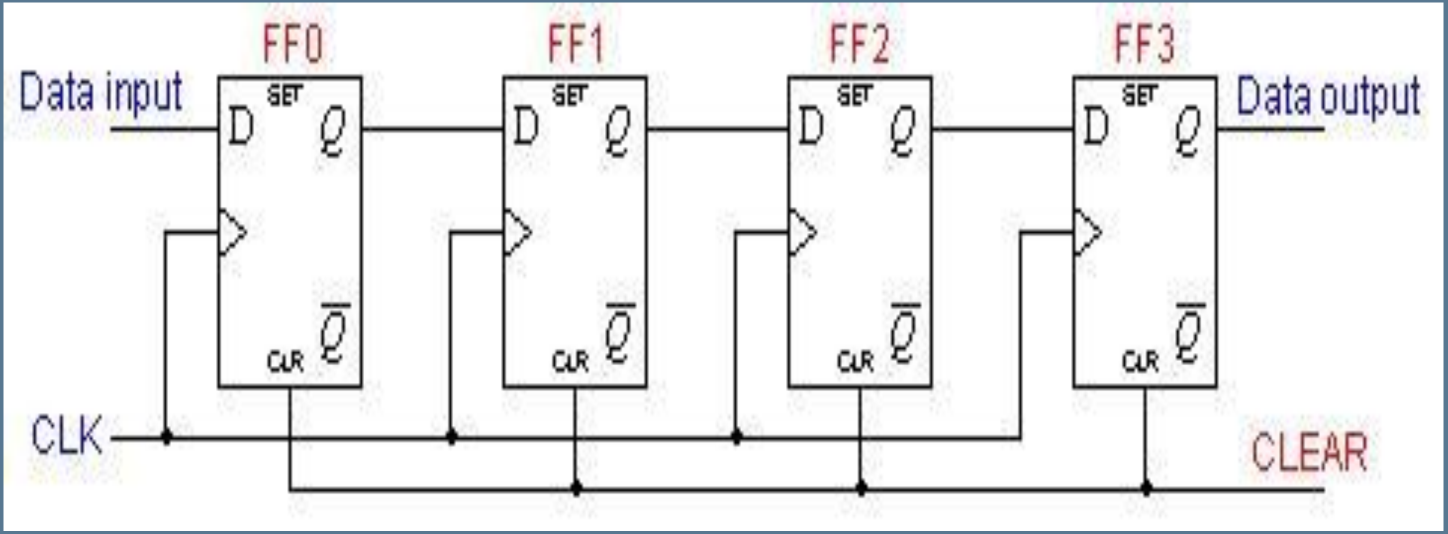
- Serial shifting
  - Movement of data from one end of a shift register to the other at a rate of one bit per clock pulse.
- Parallel transfer
  - Movement of data into all flip-flops of a shift register at the same time.

Serial-Parallel Registers

- Types
  - Serial in-serial out
  - Serial in-parallel out
  - Parallel in-serial out
  - Parallel in-parallel out

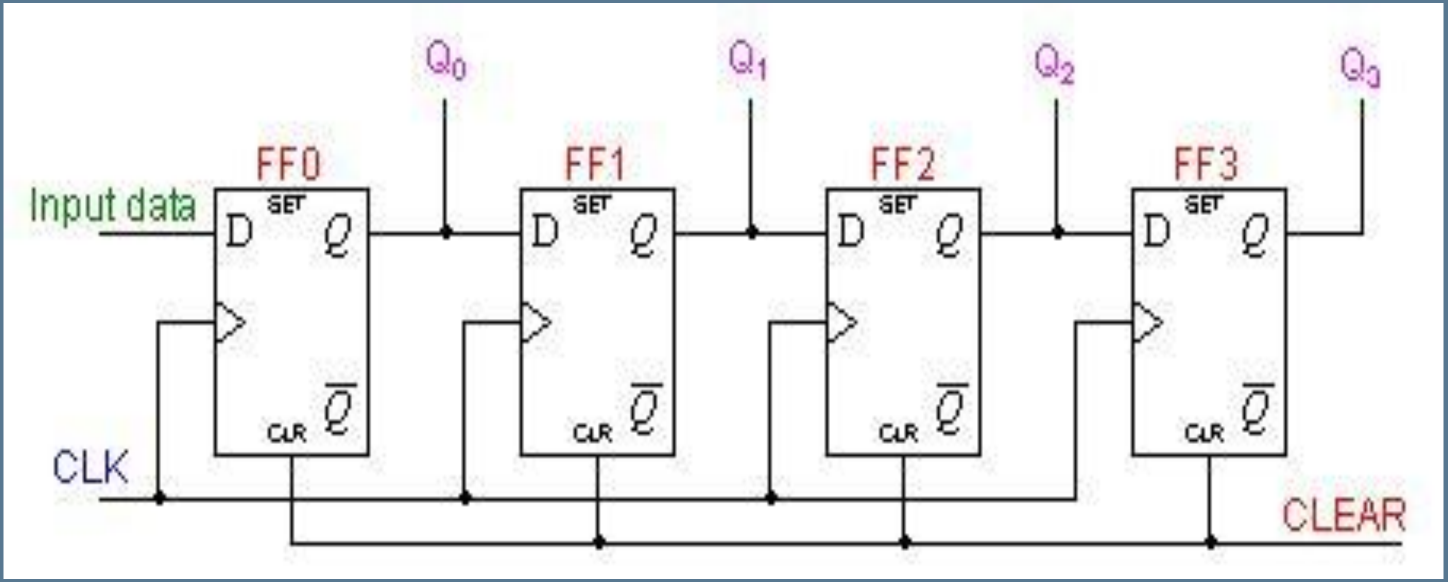
Serial In-Serial Out Register

- The register accepts one bit at a time on a single line. It produces the stored information on its output also in serial form.



Serial In-Parallel Out Register

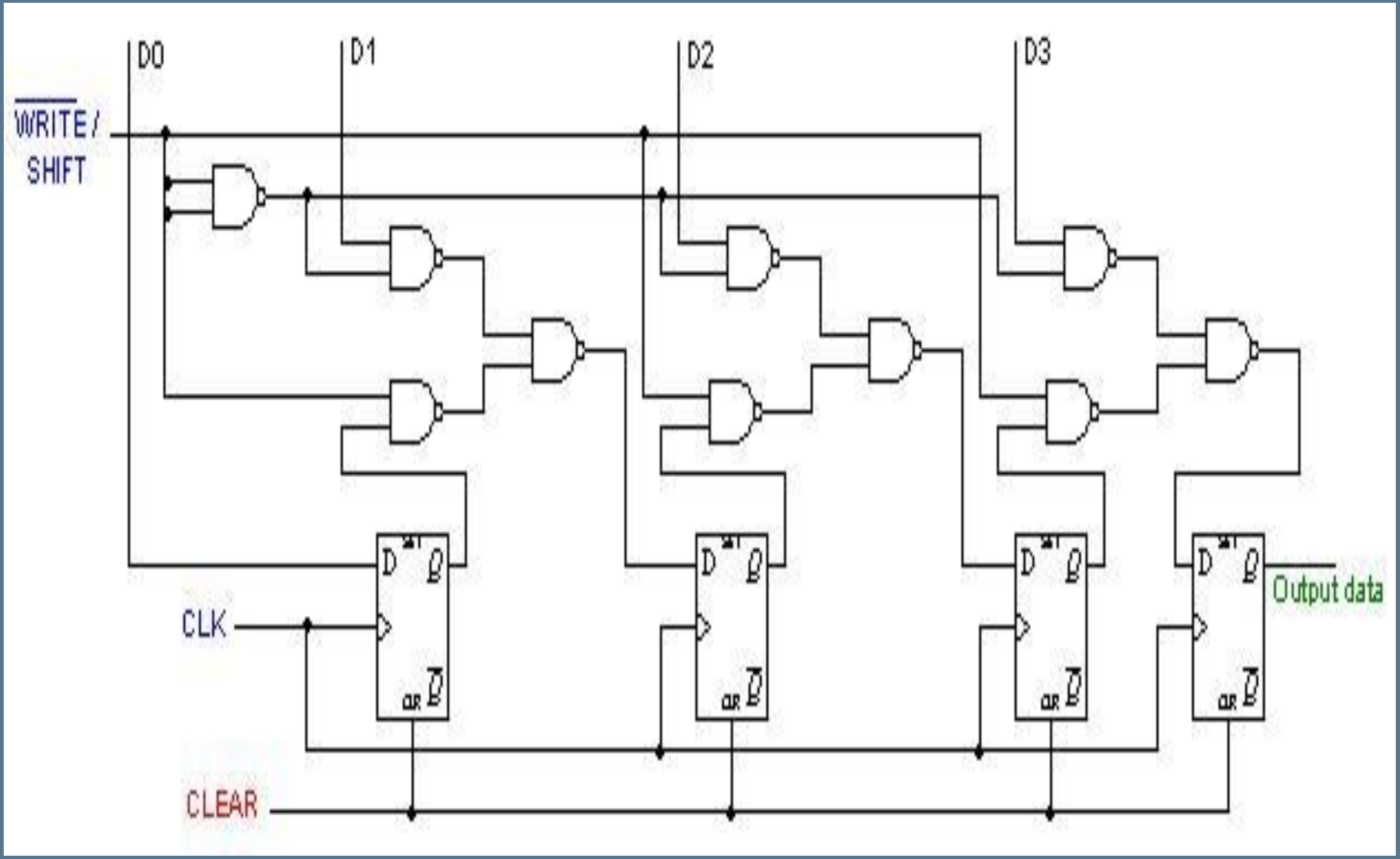
- Once the data are stored (serially), each bit appears on its respective output line and all bits are available simultaneously.



Parallel in-serial out Register

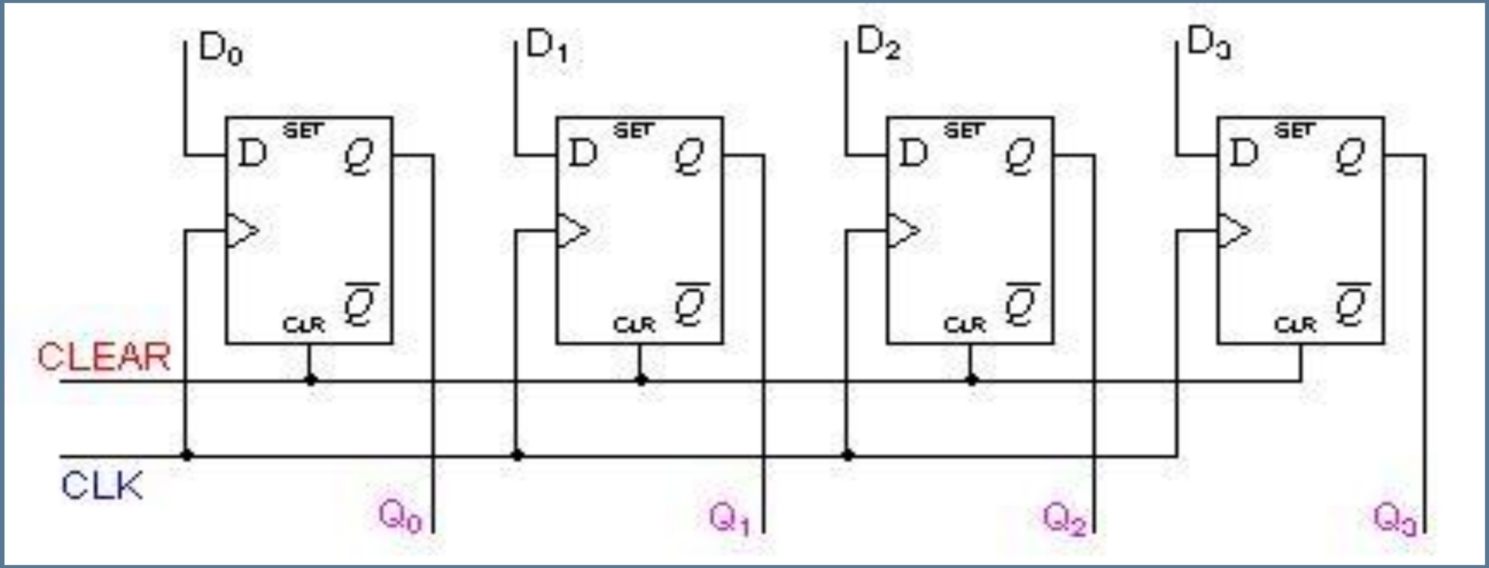
- Data bits are entered simultaneously into their respective stages on parallel lines. It produces the output in serial form.

Parallel In-Serial Out Register



Parallel In-Parallel Out Register

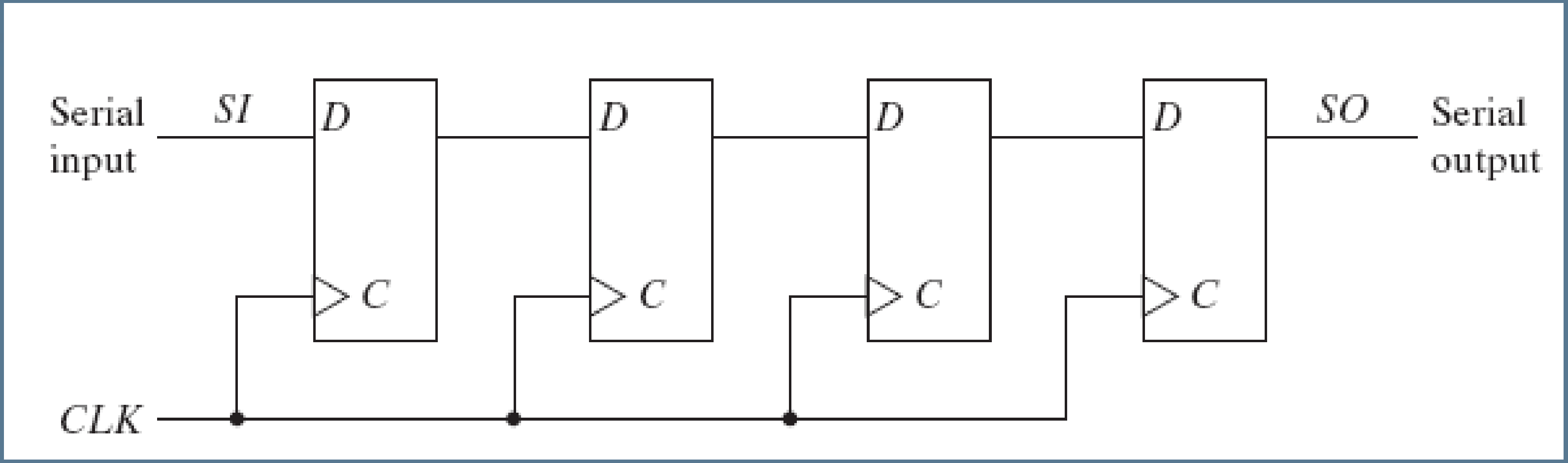
- Immediately following the simultaneously entry of all data bits, the bits appear on the parallel outputs.



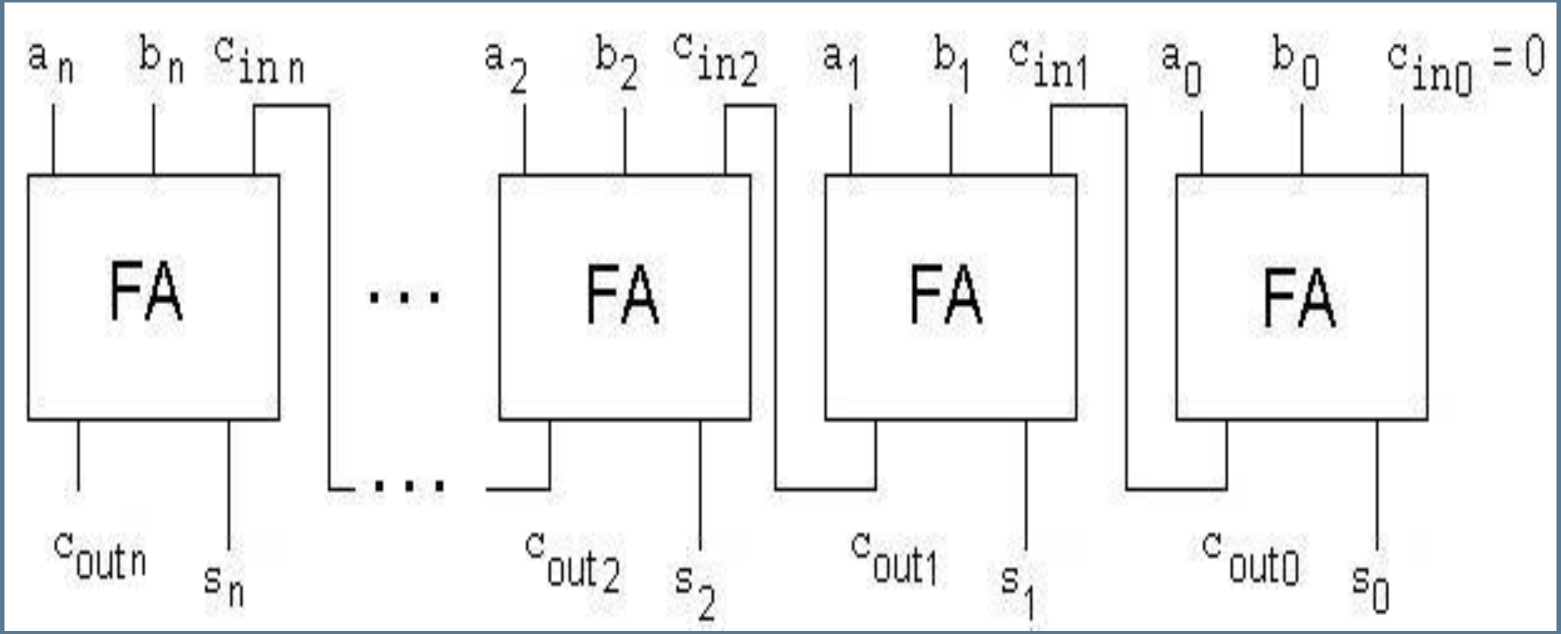
Shift Registers

- Synchronous sequential circuits used to store or move *n*-bit data. It consists of *n* flip-flops, connected so that data are transferred in and out of the flip-flops in a standard pattern.
- Types
  - Unidirectional
  - Bidirectional
  - Shift Register with parallel load

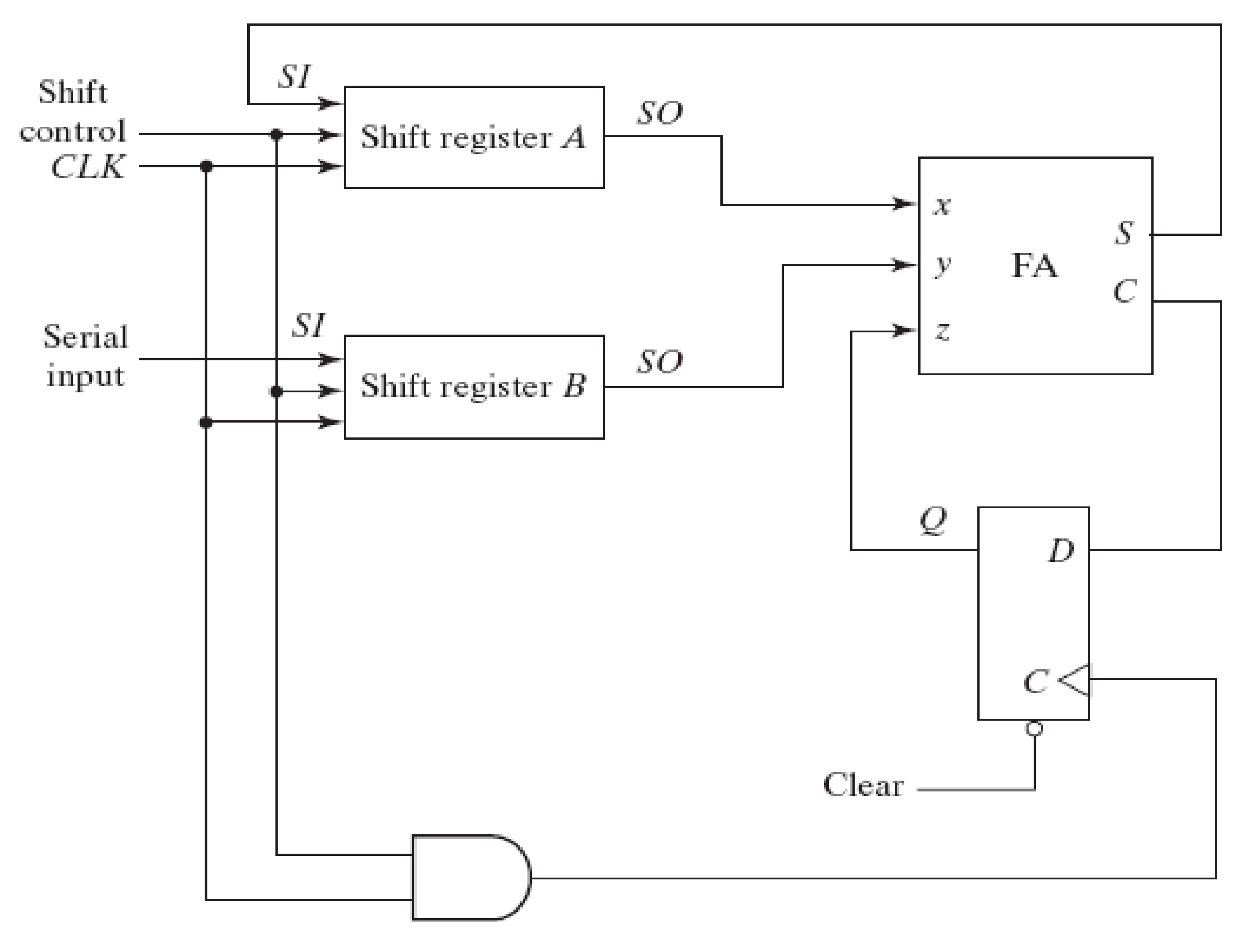
Unidirectional



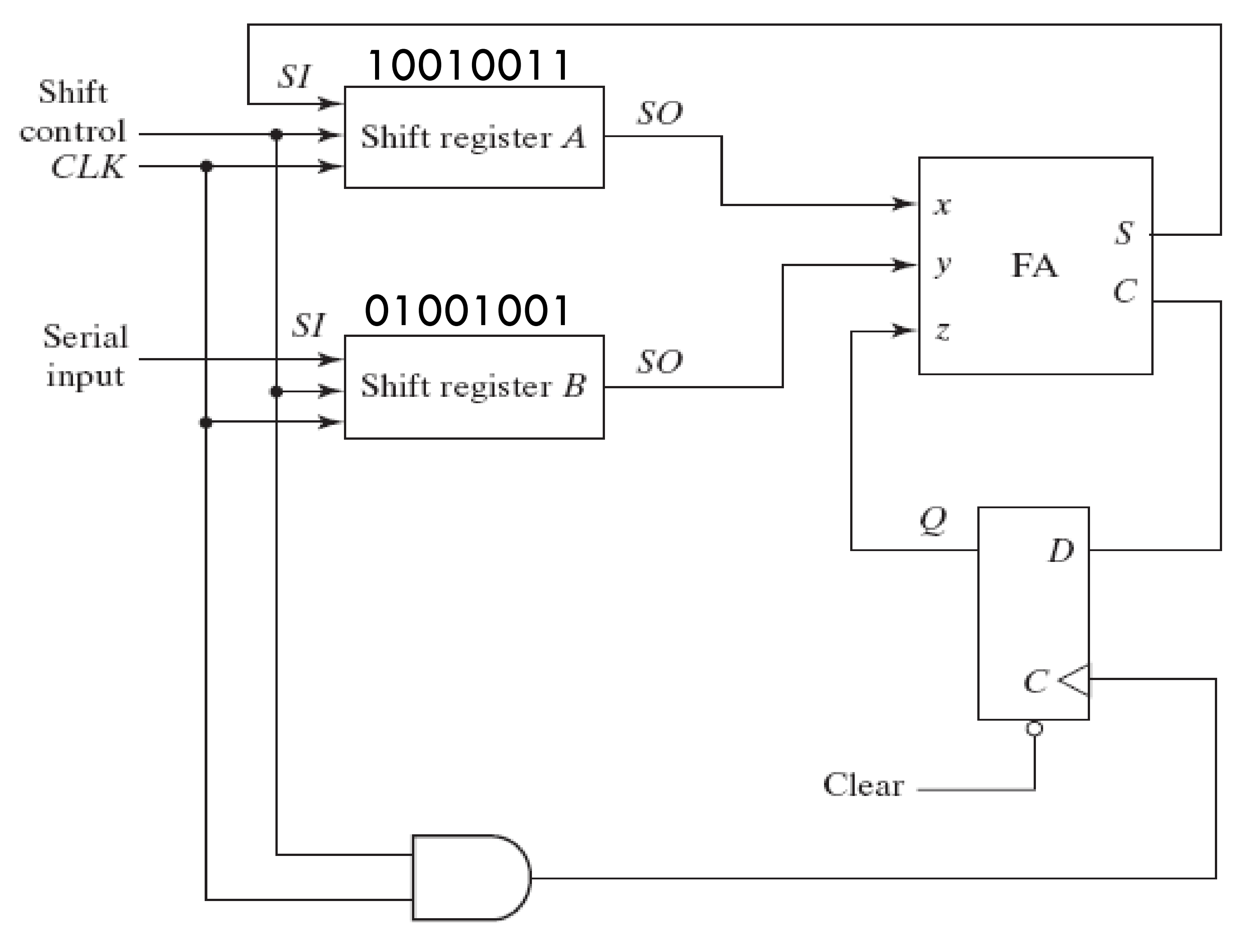
Recall: Parallel Adder



Serial Adder

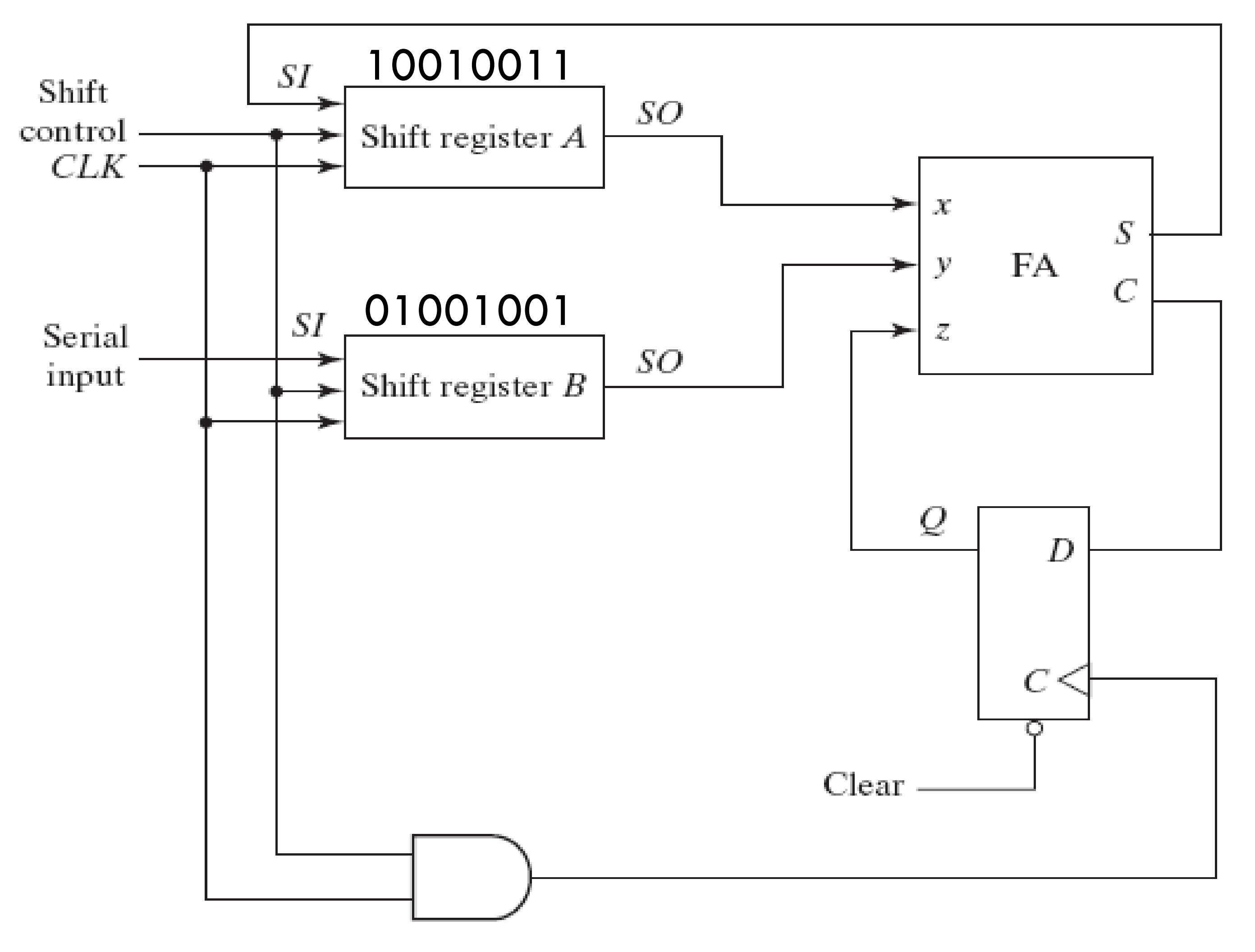


Serial Adder



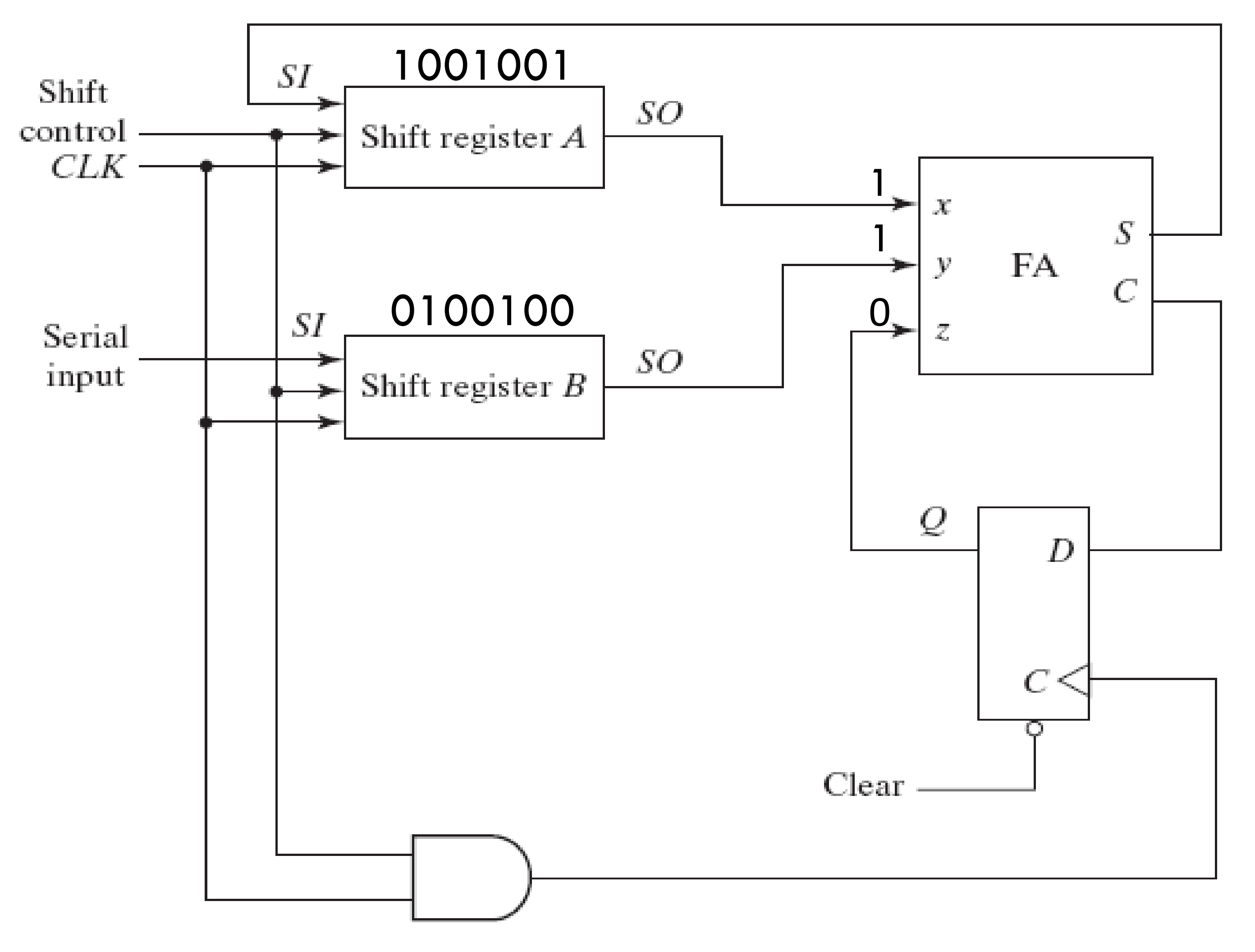
Serial Adder

```
  10010011
+ 01001001
----------
  11011100
```



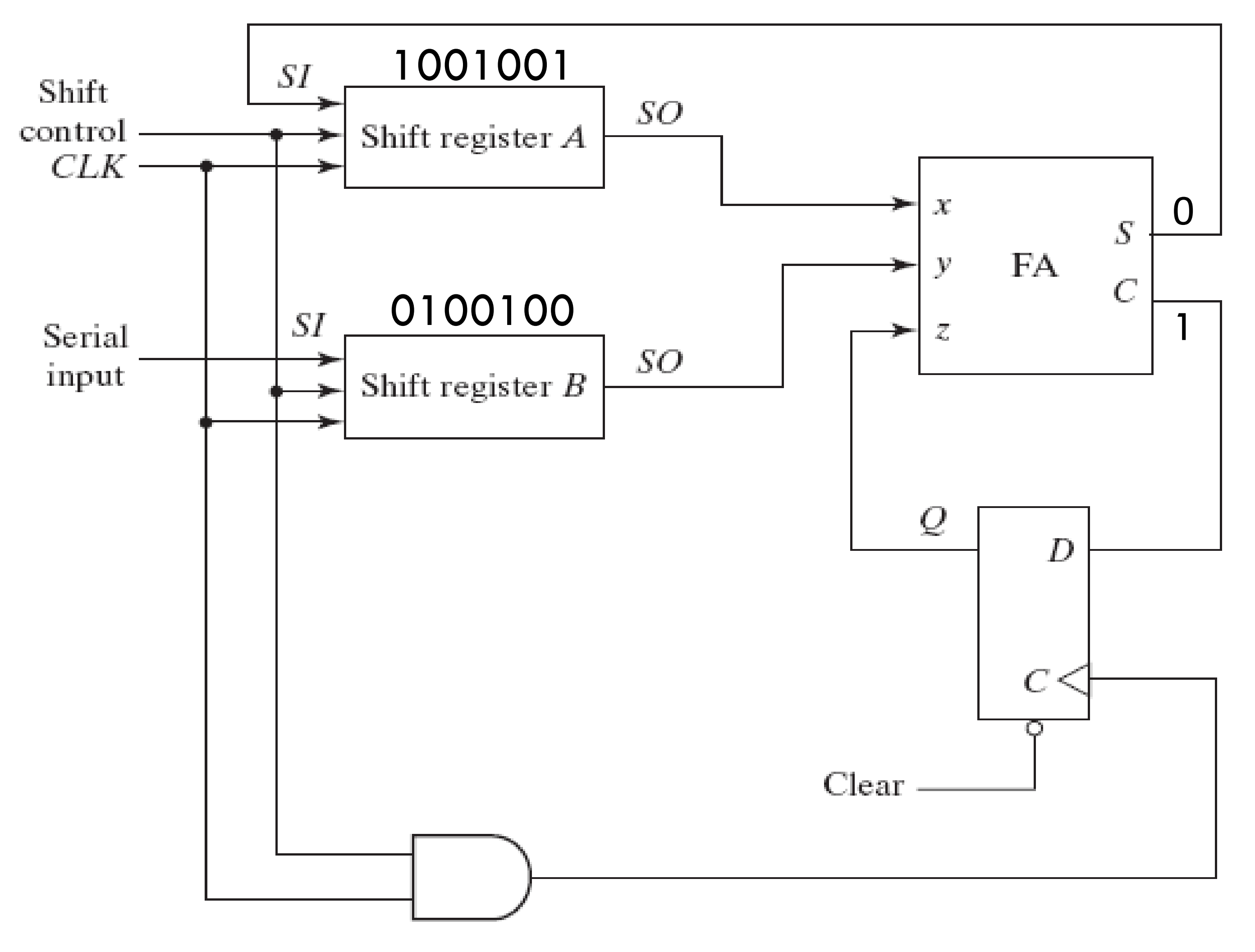
Serial Adder

```
  10010011
+ 01001001
----------
  11011100
```



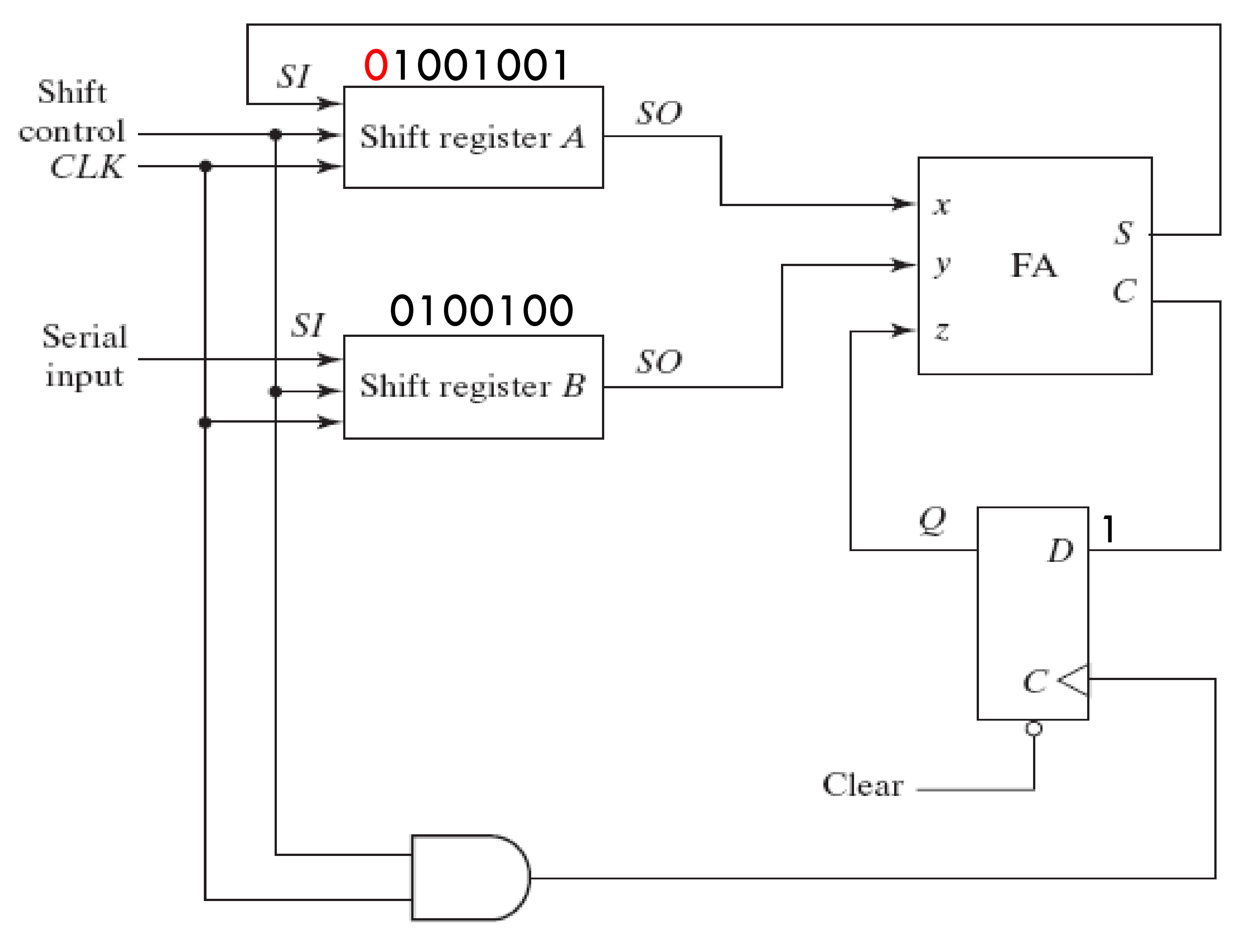
Serial Adder

```
   10010011
 + 01001001
 ----------
   11011100
```



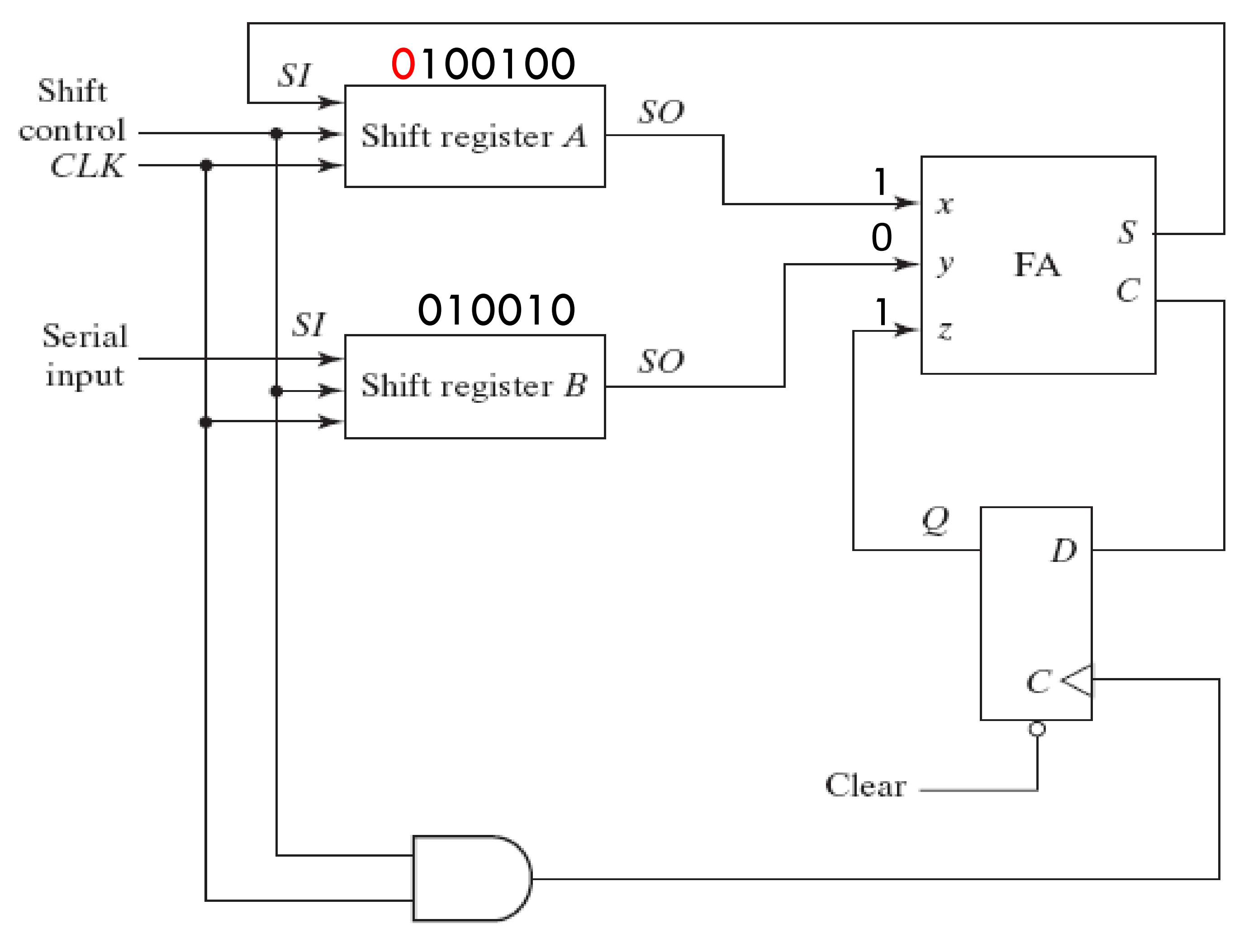
Serial Adder

```
  10010011
+ 01001001
----------
  11011100
```



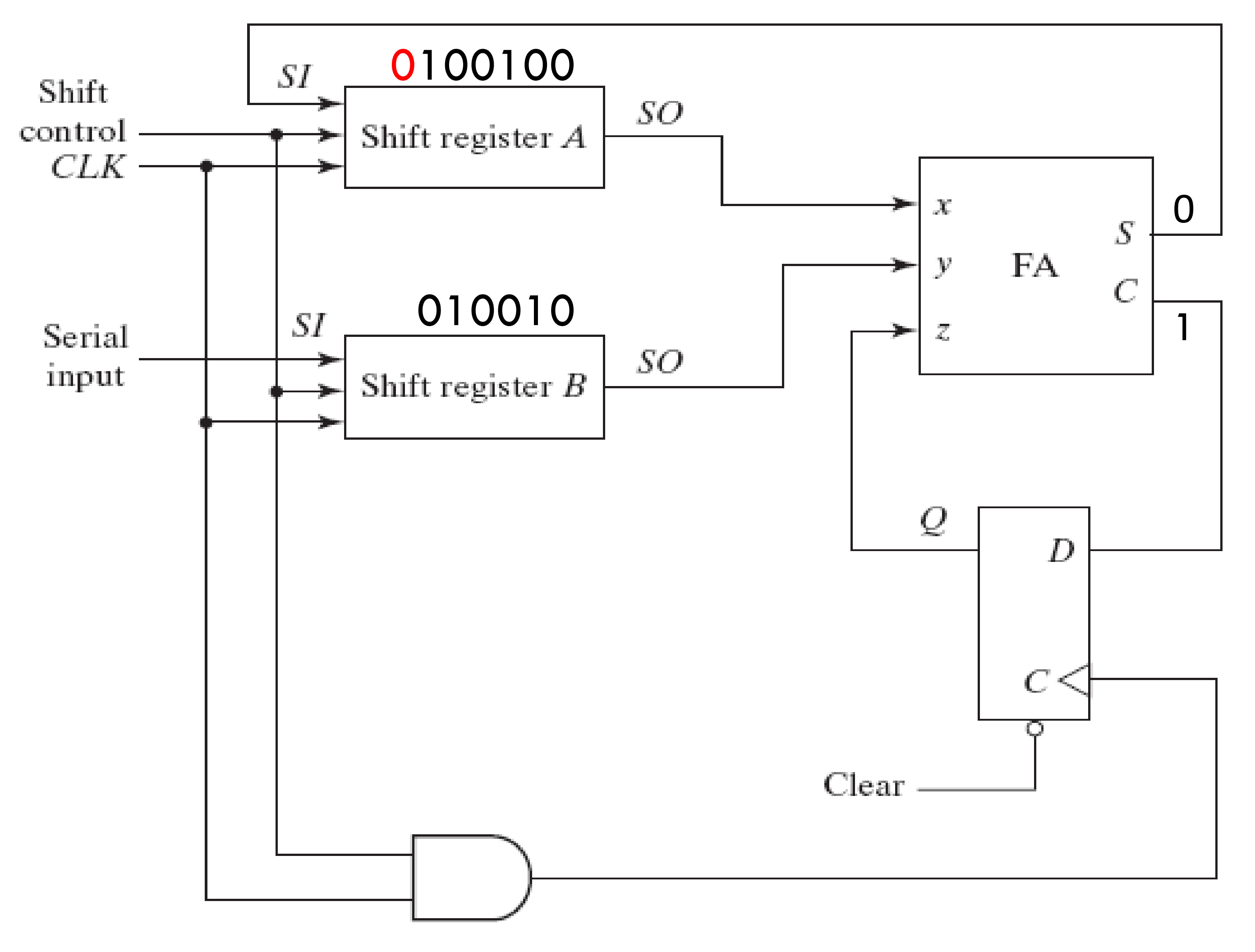
Serial Adder

```
  10010011
+ 01001001
----------
  11011100
```



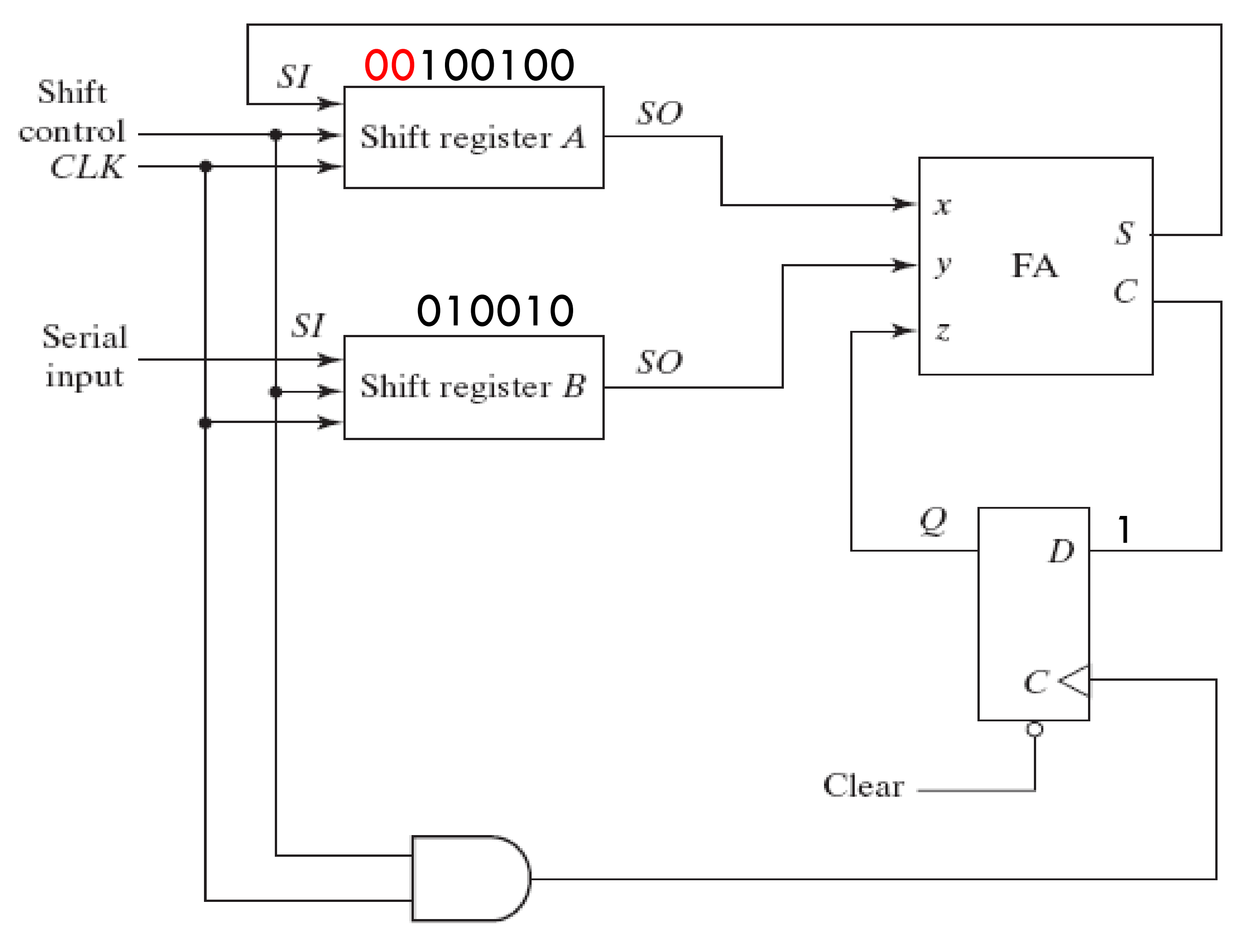
Serial Adder

```
  10010011
+ 01001001
----------
  11011100
```



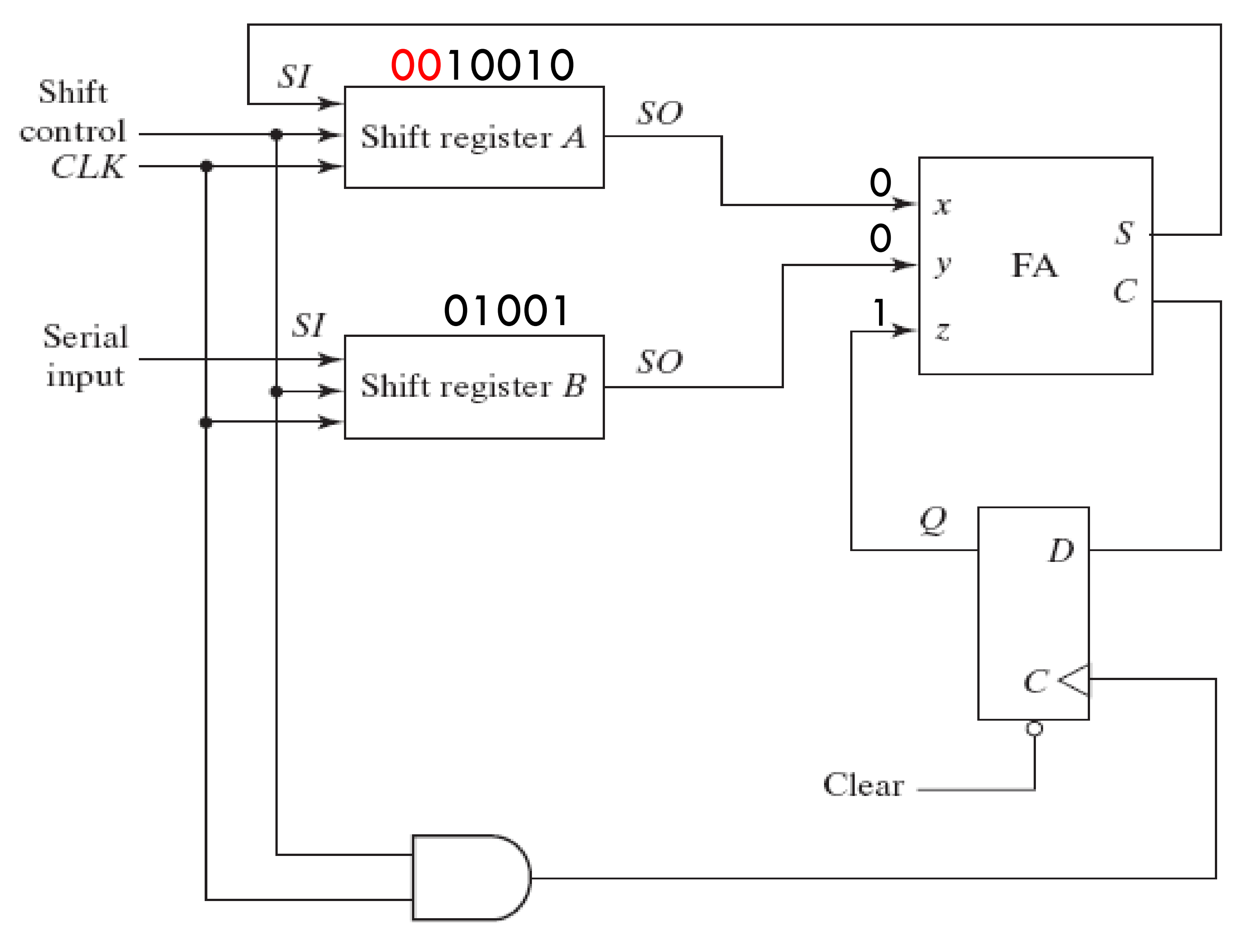
Serial Adder

```
  10010011
+ 01001001
----------
  11011100
```



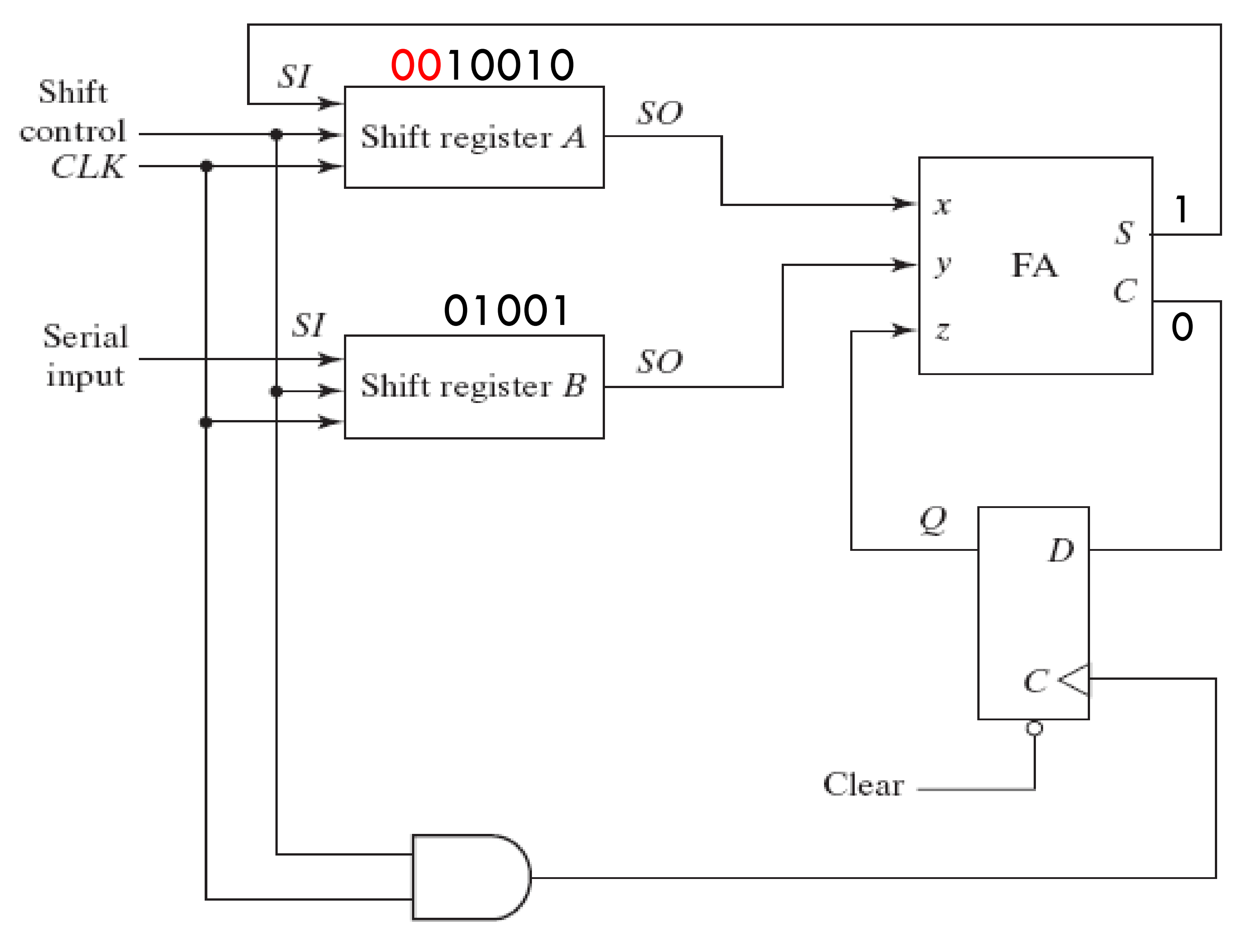
Serial Adder

```
  10010011
+ 01001001
----------
  11011100
```



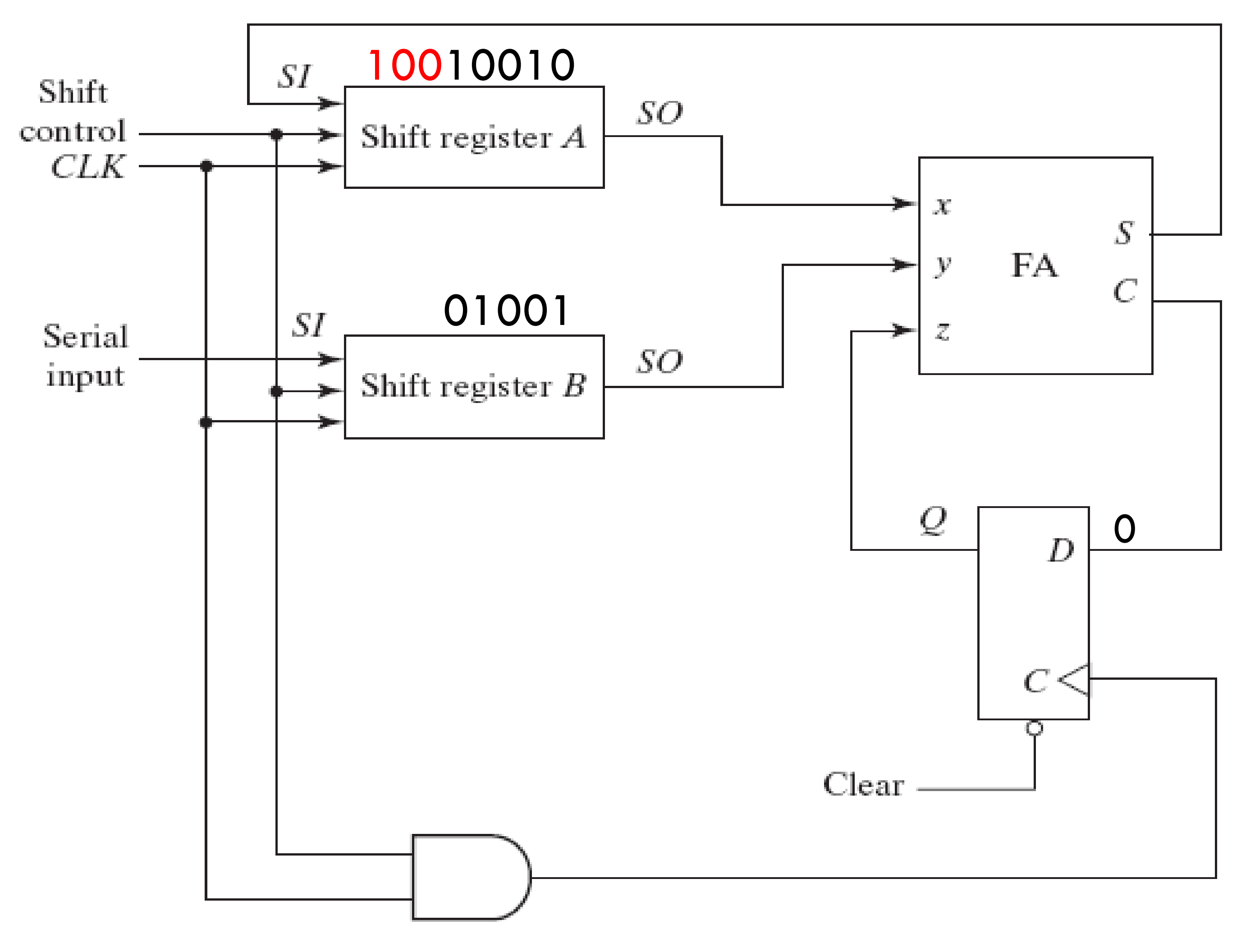
Serial Adder

```
  10010011
+ 01001001
----------
  11011100
```



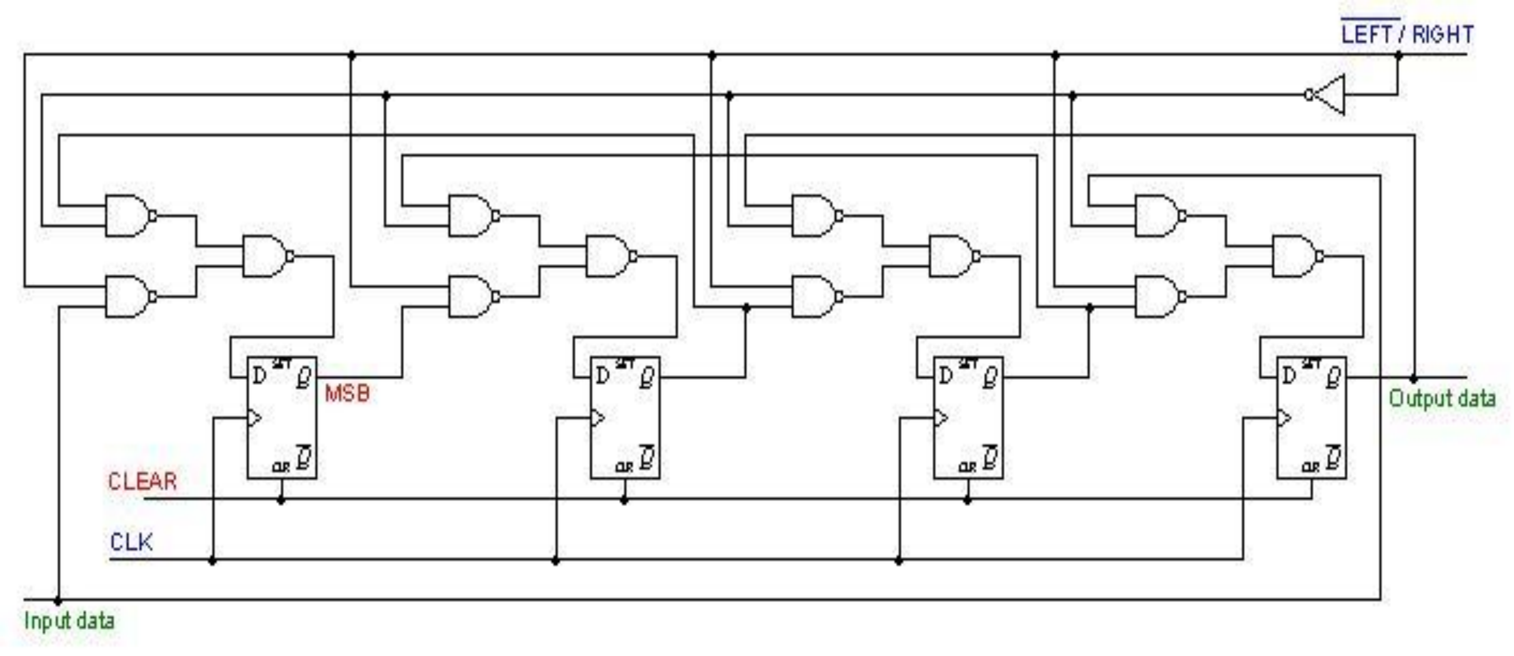
Serial Adder

```
  10010011
+ 01001001
----------
  11011100
```



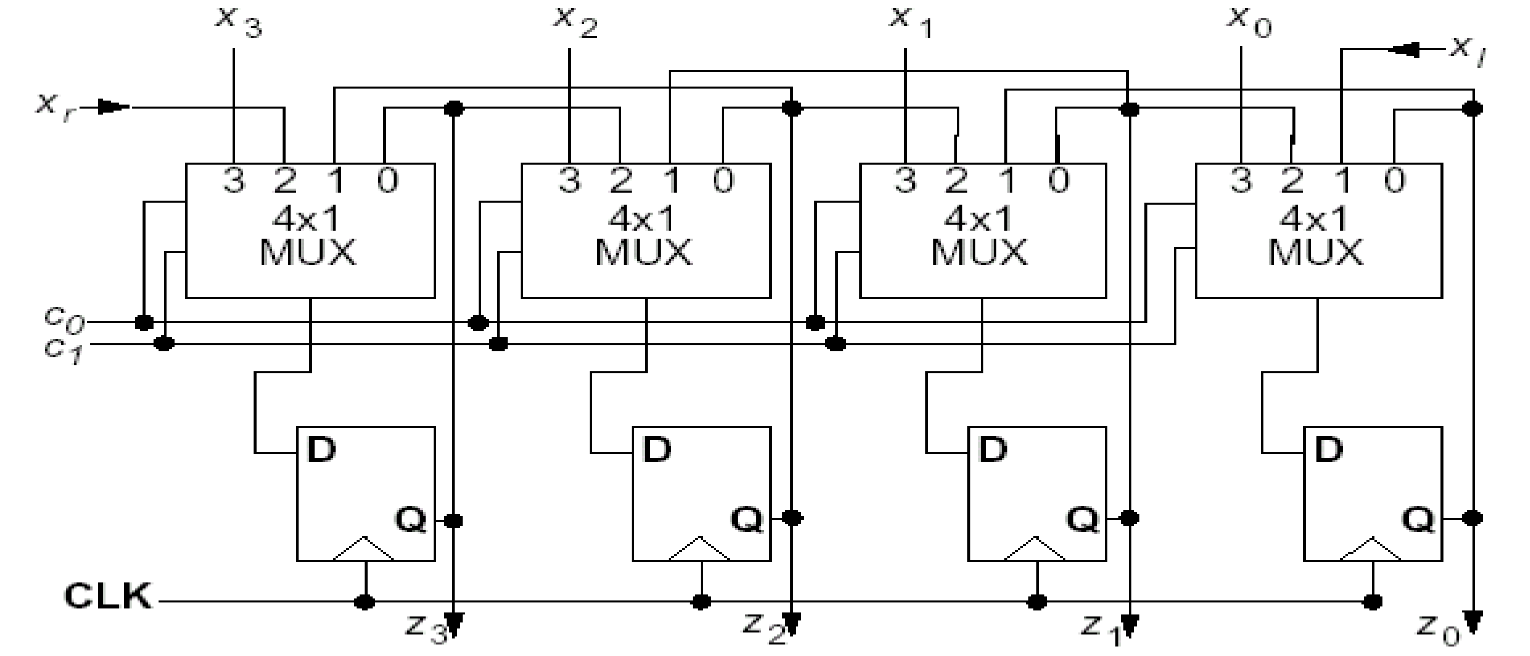
Bidirectional

- A *bidirectional*, or *reversible*, shift register is one in which the data can be shifted either left or right.



Bidirectional Shift Register with Parallel Load

- A general-purpose register capable of performing three operations: shift left, shift right, and parallel load



Rotate Registers

- Shifting of data with the output of the last flip-flop connected to the synchronous input of the first flop-flop. The result is continuous circulation of the same data.

Applications of registers

- Time delay (serial in-serial out register)
- Memory addresses
  - In assembly language, register is used as a fast memory
- Serial-to-Parallel Data converter
- Arithmetic Logic Unit (ALU)