PATRIoT Lab

Lecture 20 – Registers and Counters 1

Dr. Aftab M. Hussain,

Assistant Professor, PATRIoT Lab, CVEST

Chapter 6

Registers and counters

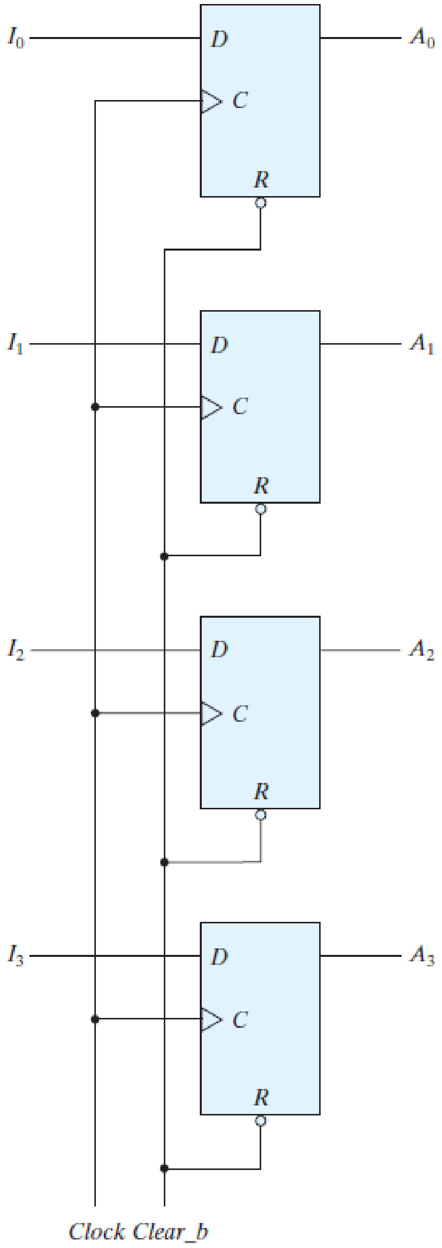
- A *register* is a group of flip-flops, each one of which shares a common clock and is capable of storing one bit of information
- An *n* -bit register consists of a group of *n* flip-flops capable of storing *n* bits of binary information
- A *counter* is essentially a register that goes through a predetermined sequence of binary states
- The counter circuit is designed in such a way as to produce the prescribed sequence of states
- Although counters are a special type of register, it is common to differentiate them by giving them a different name

Registers

- Consider a register constructed with four *D* -type flip-flops to form a four-bit data storage register

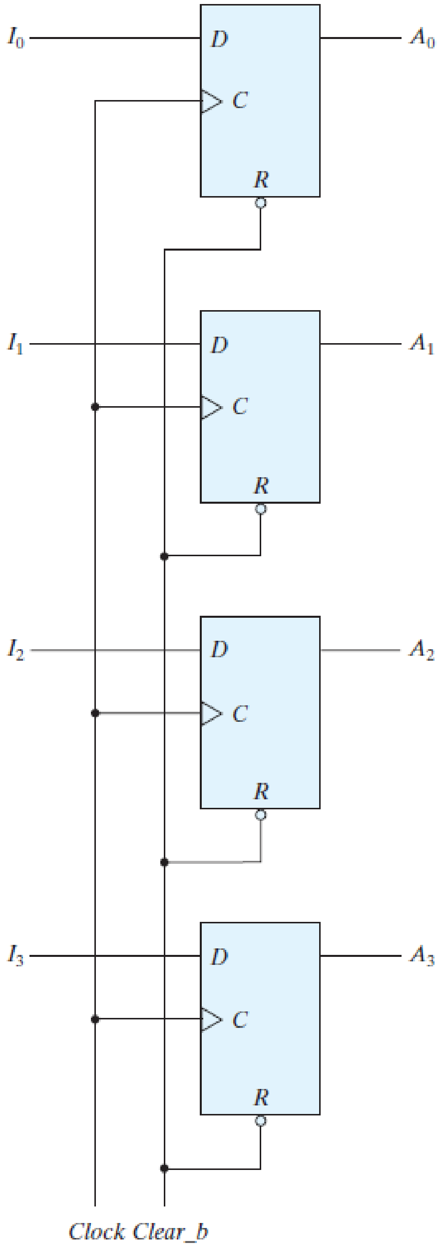
- The common clock input triggers all flip-flops on the positive edge of each pulse, and the binary data available at the four inputs are transferred into the register

- The value of (I_3 ,I_2 , I_1, I_0) immediately before the clock edge determines the value of (A_3, A_2, A_1, A_0) after the clock edge



Registers

- The four outputs can be sampled at any time to obtain the binary information stored in the register
- The input *Clear_b* goes to the active-low *R* (reset) input of all four flip-flops
- When this input goes to 0, all flip-flops are reset asynchronously
- The *Clear_b* input is useful for clearing the register to all 0's prior to its clocked operation
- The *R* inputs must be maintained at logic 1 (i.e., de-asserted) during normal clocked operation



Registers with load input

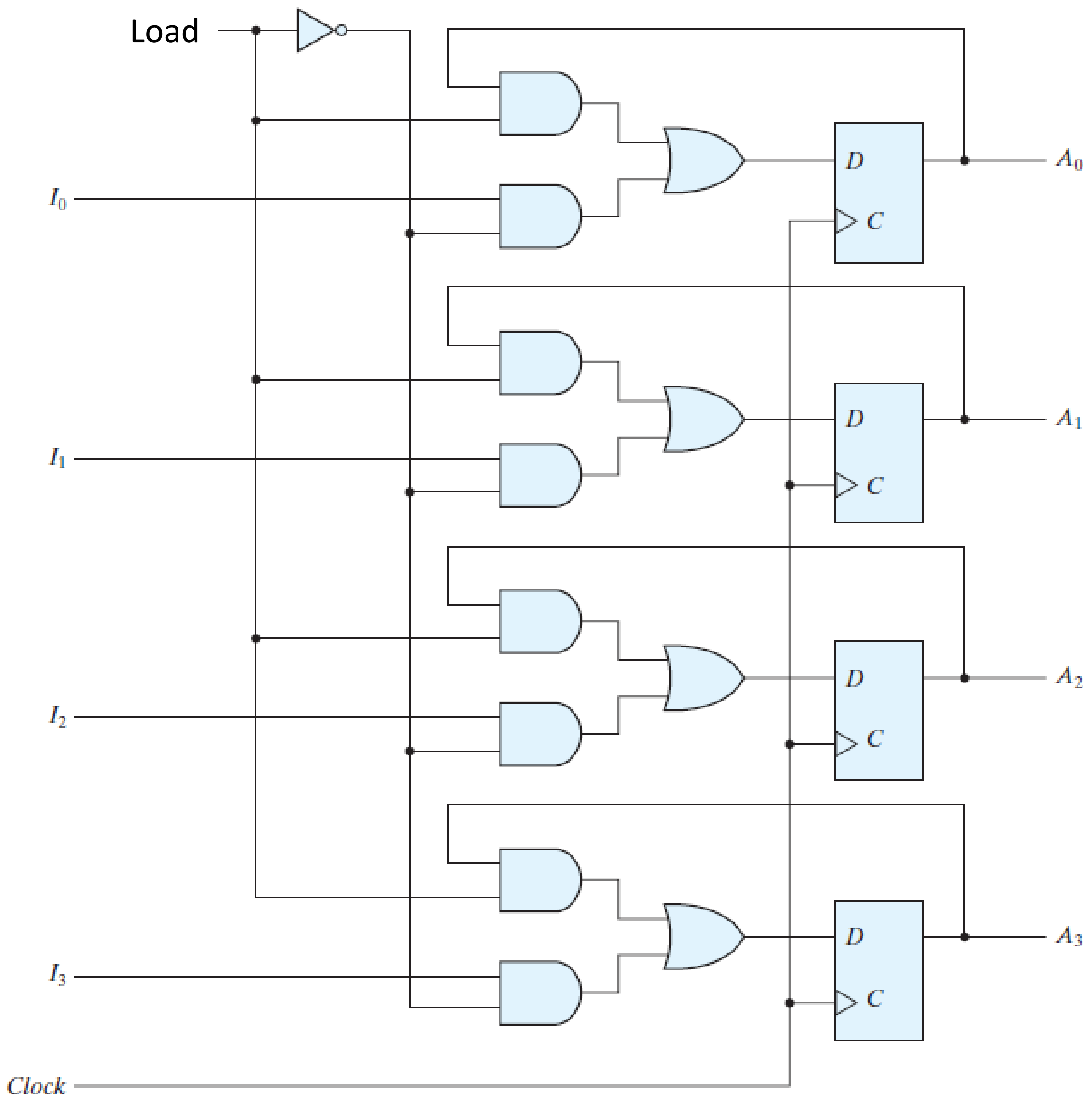
- Synchronous digital systems have a master clock generator that supplies a continuous train of clock pulses
- The pulses are applied to all flip-flops and registers in the system
- The master clock acts like a drum that supplies a constant beat to all parts of the system (like the heart-beat of the processor)
- However, we might not be interested in changing data in the register every time, in some cases, we may want to keep the data unchanged
- A separate control signal must be used to decide which register operation will execute at each clock pulse
- The transfer of new information into a register is referred to as *loading* or *updating* the register

Registers with load input

- In this configuration, if the contents of the register must be left unchanged, the inputs must be held constant, or the clock must be inhibited from the circuit
- However, inserting gates into the clock path is ill advised because it means that logic is performed with clock pulses
- The insertion of logic gates produces uneven propagation delays between the master clock and the inputs of flip-flops
- To fully synchronize the system, we must ensure that all clock pulses arrive at the same time anywhere in the system, so that all flip-flops trigger simultaneously
- For this reason, it is advisable to control the operation of the register with the *D* inputs, rather than controlling the clock in the *C* inputs of the flip-flops
- This creates the effect of a gated clock, but without affecting the clock path of the circuit

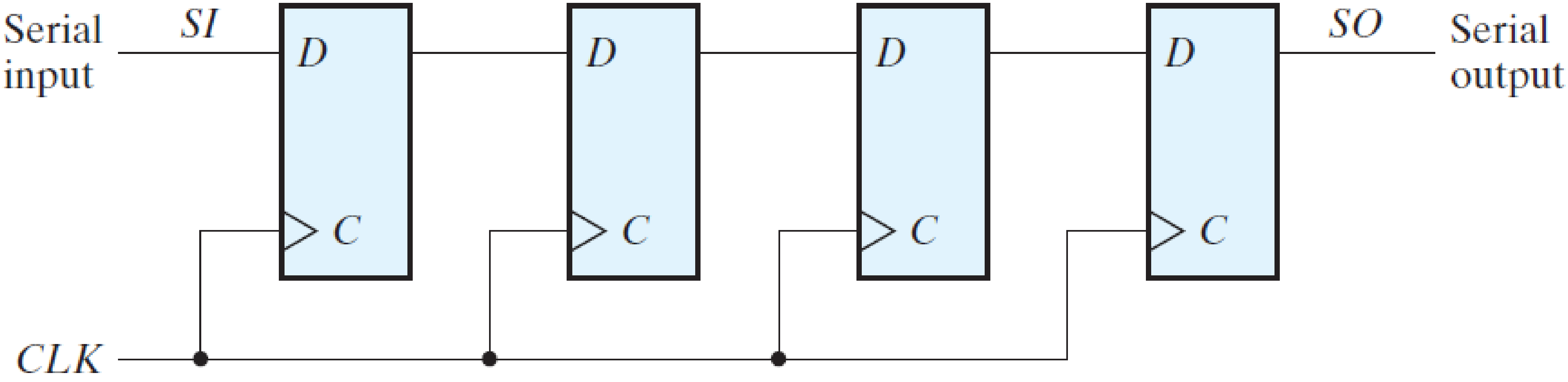
Registers with load input

- A four-bit data-storage register with a load control input is as shown
- The additional gates implement a two-channel mux whose output drives the input to the register with either the data bus or the output of the register
- When the load input is 0, the data at the four external inputs are transferred into the register with the next positive edge of the clock
- When the load input is 1, the outputs of the flip-flops are connected to their respective inputs
- The feedback connection from output to input is necessary because a *D* flip-flop does not have a "no change" condition



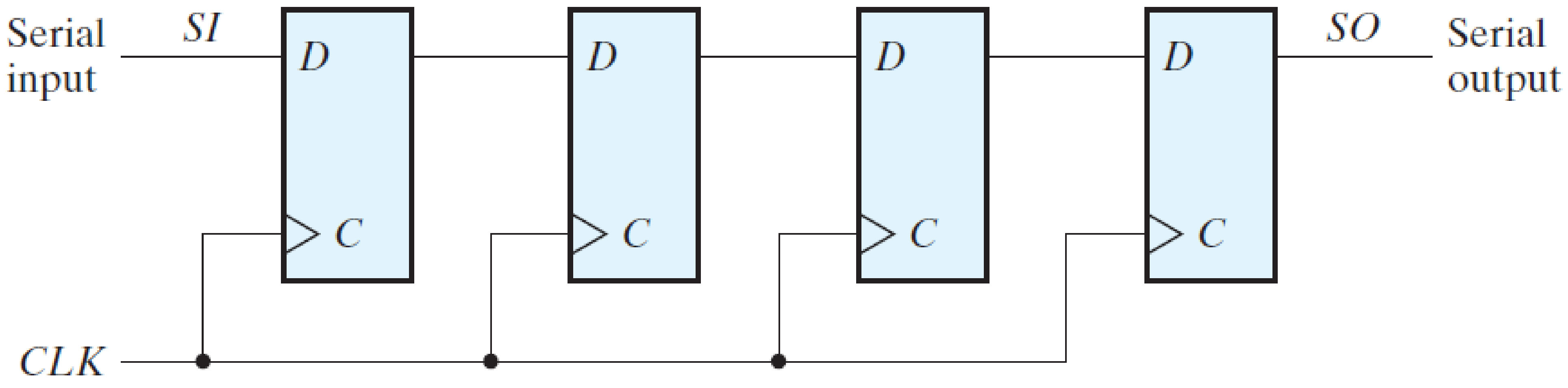
Shift register

- A register capable of shifting the binary information held in each cell to its neighboring cell, in a selected direction, is called a *shift register*
- The logical configuration of a shift register consists of a chain of flip-flops in cascade, with the output of one flip-flop connected to the input of the next flip-flop
- All flip-flops receive common clock pulses, which activate the shift of data from one stage to the next



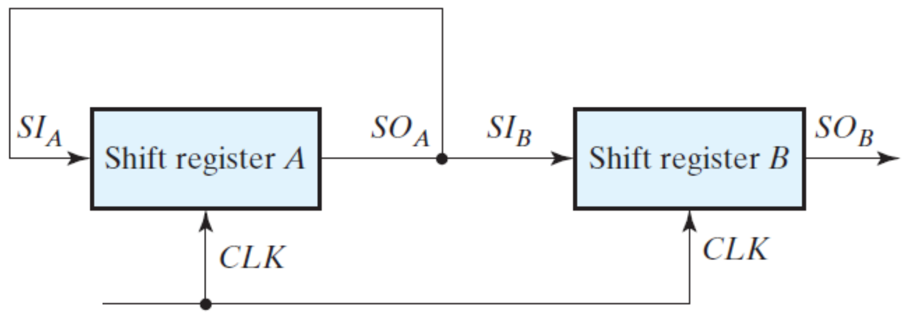
Shift register with load input

- Sometimes it is necessary to control the shift so that it occurs only with certain pulses, but not with others
- Recirculate the output of each cell back through a two-channel mux whose output is connected to the input of the cell
- When the clock action is not suppressed, the other channel of the mux provides a datapath to the cell

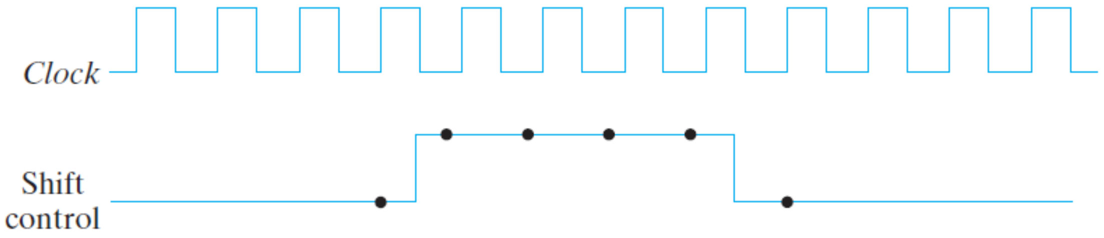


Serial transfer

- The datapath of a digital system is said to operate in serial mode when information is transferred and manipulated one bit at a time
- Information is transferred one bit at a time by shifting the bits out of the source register and into the destination register
- This type of transfer is in contrast to parallel transfer, whereby all the bits of the register are transferred at the same time
- The serial transfer of information from register *A* to register *B* is done with shift registers, as shown
- The serial output (*SO*) of register *A* is connected to the serial input (*SI*) of register *B*

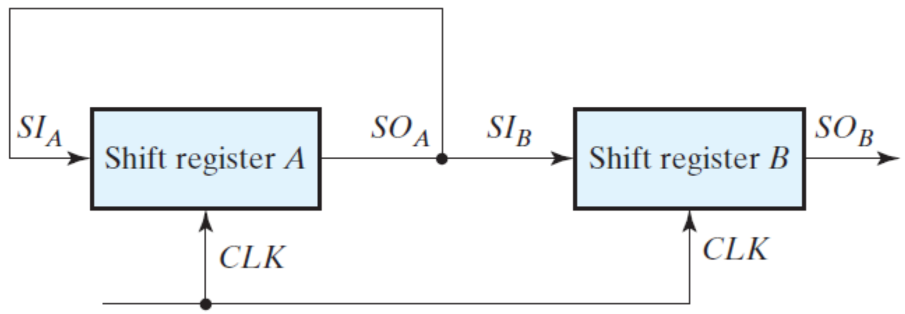


(a) Block diagram

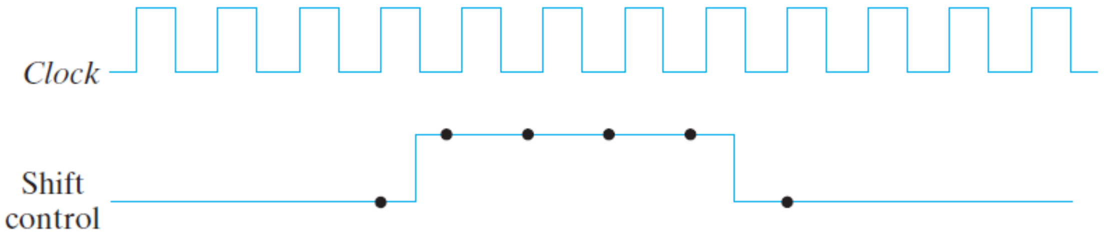


Serial transfer

- To prevent the loss of information stored in the source register, the information in register *A* is made to circulate by connecting the serial output to its serial input
- The initial content of register *B* is shifted out through its serial output and is lost unless it is transferred to a third shift register
- The shift control input determines when and how many times the registers are shifted
- For simplicity here, this is done with an AND gate that allows clock pulses to pass into the *CLK* terminals only when the shift control is active

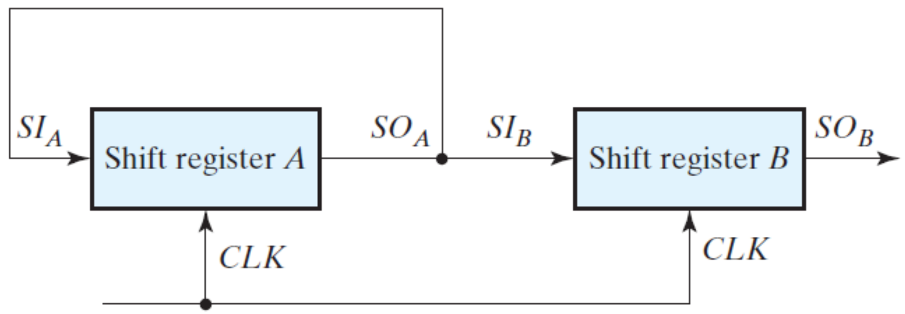


(a) Block diagram



Serial transfer

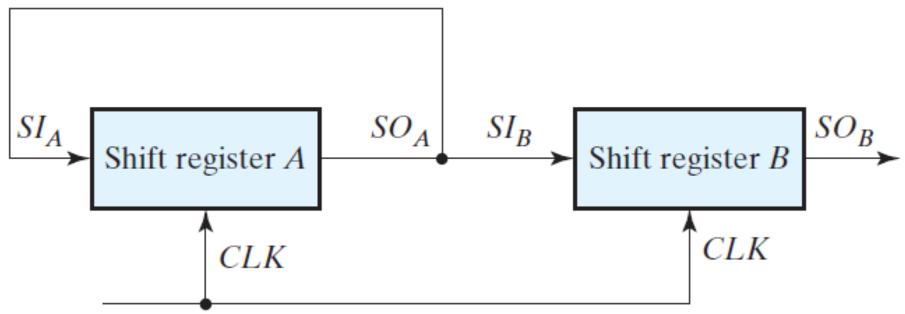
- Assume that the binary content of *A* before the shift is 1011 and that of *B* is 0010
- The serial transfer from *A* to *B* occurs in four steps
- With the first pulse, T_1, the rightmost bit of *A* is shifted into the leftmost bit of *B* and is also circulated into the leftmost position of *A*
- At the same time, all bits of *A* and *B* are shifted one position to the right



(a) Block diagram

Serial transfer

- The previous serial output from *B* in the rightmost position is lost, and its value changes from 0 to 1
- The next three pulses perform identical operations, shifting the bits of *A* into *B,* one at a time
- After the fourth shift, the shift control goes to 0, and registers *A* and *B* both have the value 1011
- Thus, the contents of *A* are copied into *B*, so that the contents of *A* remain unchanged i.e., the contents of *A* are restored to their original value

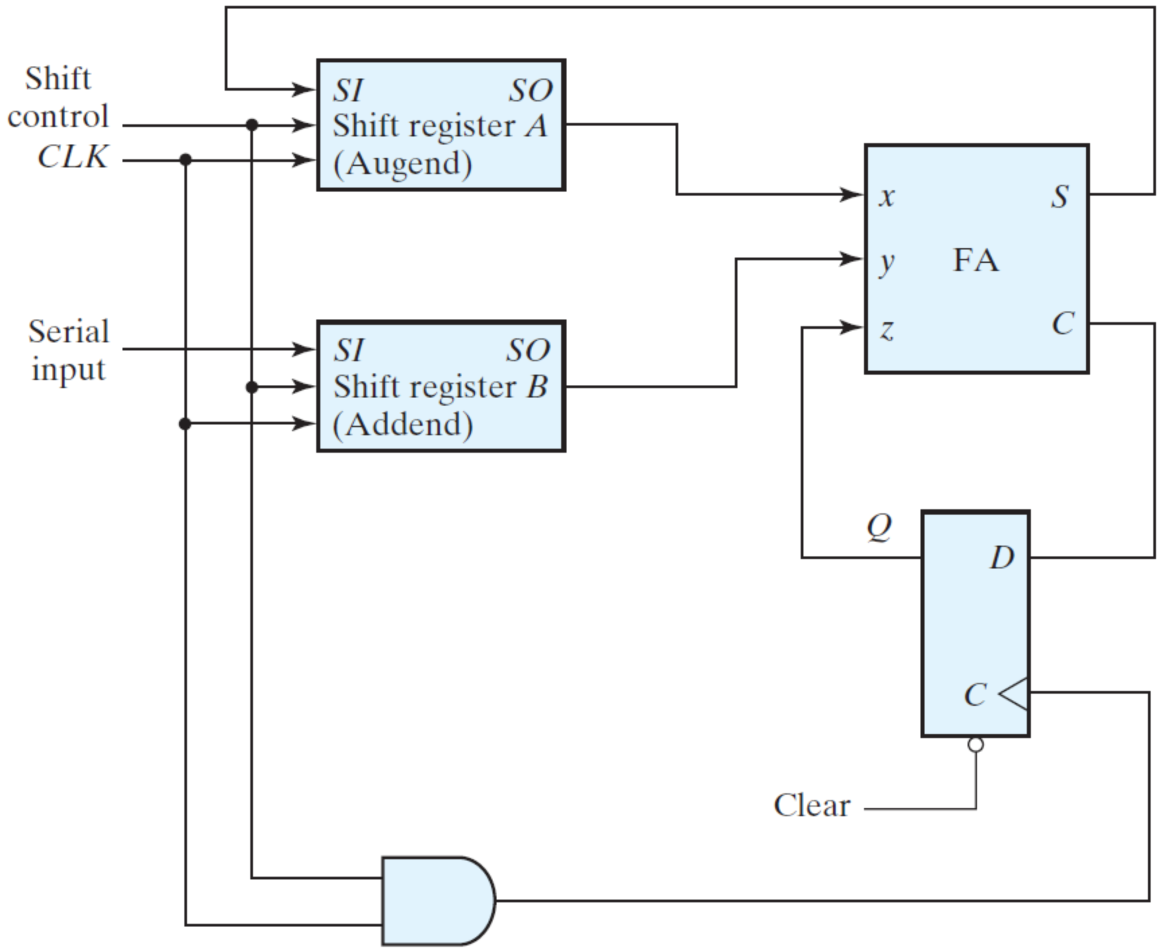


(a) Block diagram

Timing Pulse	Shift Register *A*				Shift Register *B*			
Initial value	1	0	1	1	0	0	1	0
After T_1	1	1	0	1	1	0	0	1
After T_2	1	1	1	0	1	1	0	0
After T_3	0	1	1	1	0	1	1	0
After T_4	1	0	1	1	1	0	1	1

Serial addition

- Can we design a serial addition circuit? The two binary numbers to be added serially are stored in two shift registers
- This is similar to the algorithm we use for adding manually
- Beginning with the least significant pair of bits, the circuit adds one pair at a time through a single full-adder (FA) circuit
- The carry out of the full adder is transferred to a *D* flip-flop, the output of which is then used as the carry input for the next pair of significant bits
- The sum bit from the *S* output of the full adder could be transferred into a third shift register
- By shifting the sum into *A* while the bits of *A* are shifted out, it is possible to use one register for storing both the augend and the sum bits
- The serial input of register *B* can be used to transfer a new binary number while the addend bits are shifted out during the addition



Serial addition

- Comparing the serial adder with the parallel adder (4-bit adder), we note several differences
- The parallel adder uses registers with a parallel load, whereas the serial adder uses shift registers
- The number of full-adder circuits in the parallel adder is equal to the number of bits in the binary numbers, whereas the serial adder requires only one full-adder circuit and a carry flip-flop
- Excluding the registers, the parallel adder is a combinational circuit, whereas the serial adder is a sequential circuit which consists of a full adder and a flip-flop that stores the output carry