REGISTERS

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.

In addition to the flip-flops, a register may have combinational gates that perform certain data-processing tasks.

In its broadest definition, a register consists of a group of flip-flops together with gates that affect their operation.

The flip-flops hold the binary information, and the gates determine how the information is transferred into the register.

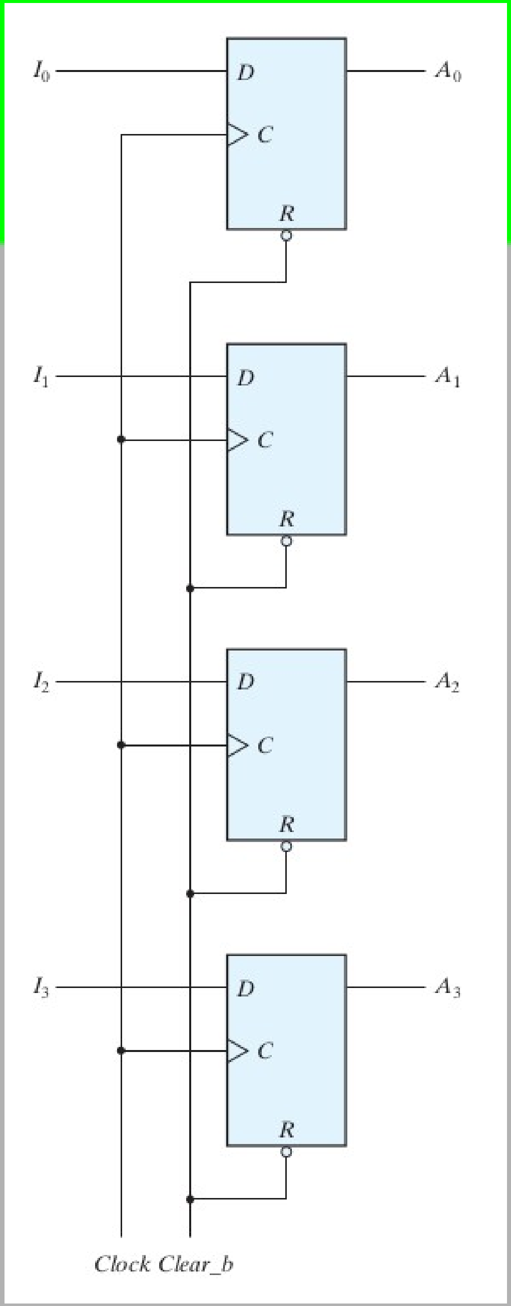
A counter is essentially a register that goes through a predetermined sequence of binary states.

The gates in the counter are connected 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.

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.

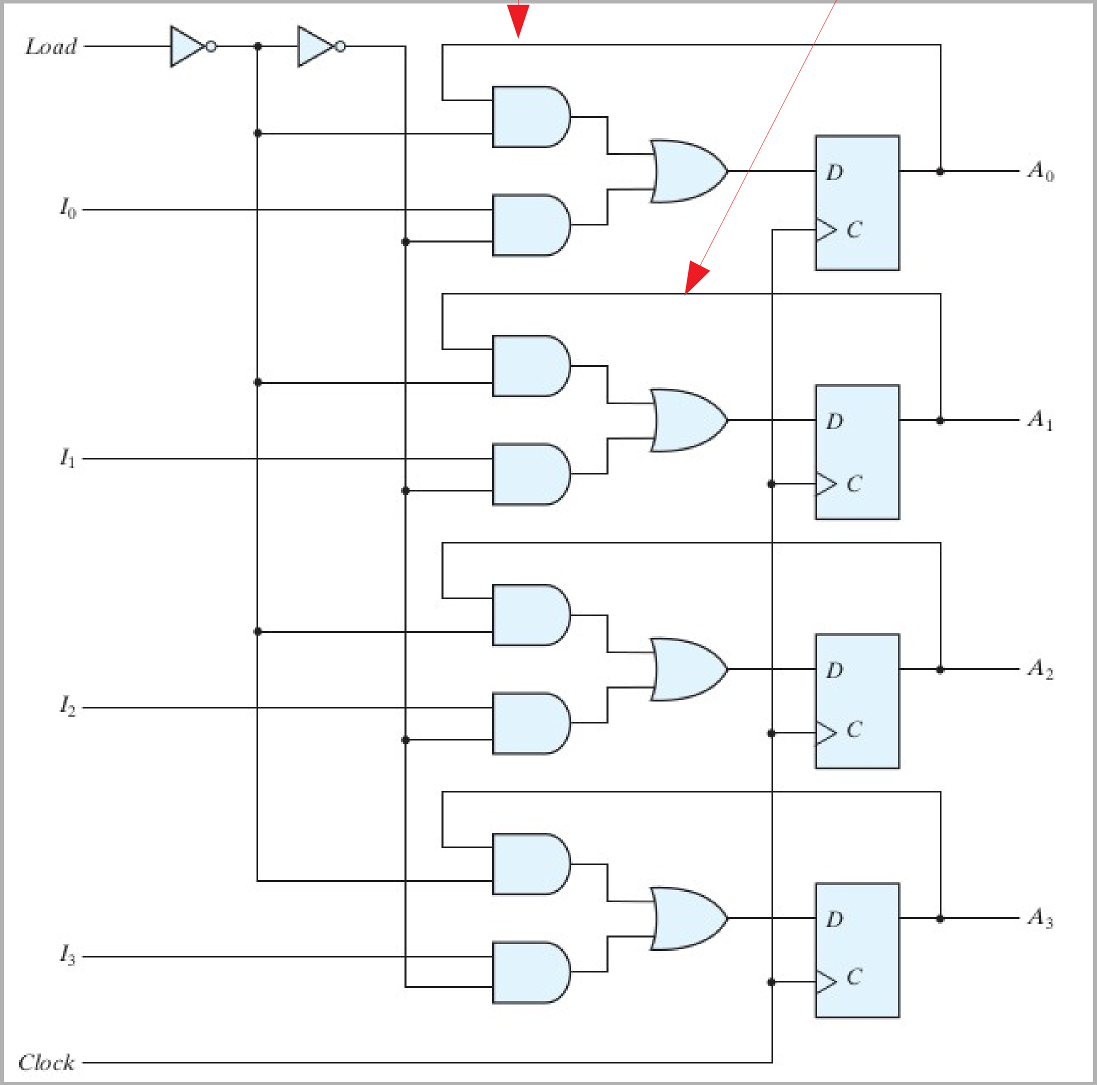
The value of (I3, I2, I1, I0) immediately
before the clock edge determines the value of
(A3, A2, A1, A 0) after the clock edge.



Four-bit register with parallel load

The feed-back connection from output to input is necessary because a D flip-flop does not have a "no change" condition.

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. The load input to the register determines the action to be taken with each clock pulse. When the load input is 1, 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 0, the outputs of the flip-flops are connected to their respective inputs.



The clock could be inhibited from reaching the register by controlling the clock input signal with an enabling gate.

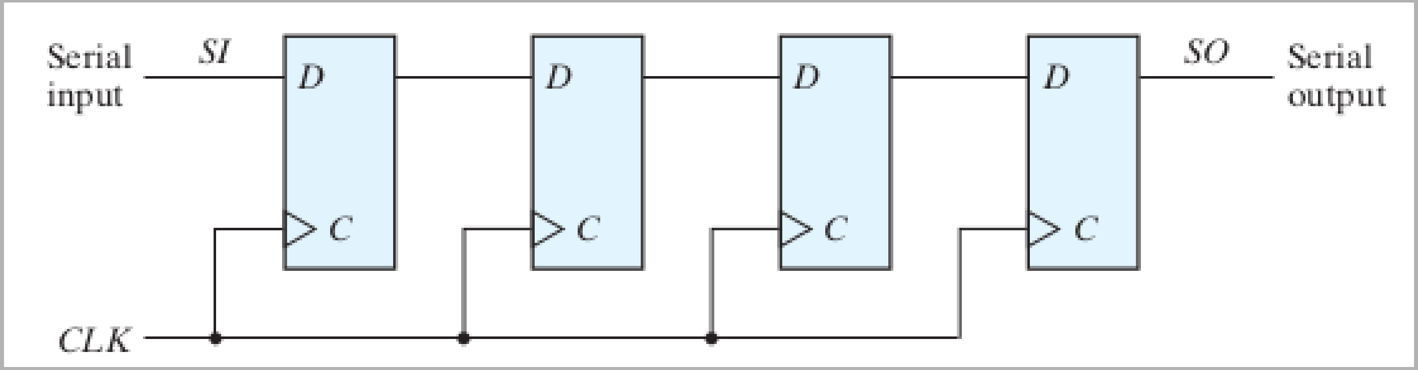
Inserting gates into the clock path is ill advised because it means that logic is performed with clock pulses. The insertion of logic gates produces uneven propagation delays between the master clock and the inputs of flip-flops.

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.

Performing logic with clock pulses inserts variable delays and may cause the system to go out of synchronism.

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.

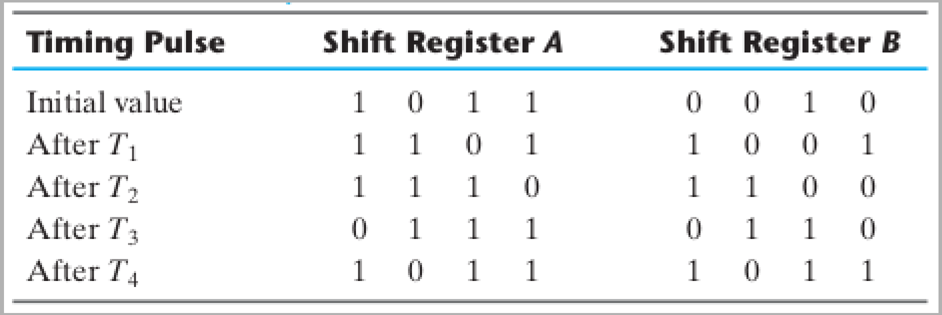
Shift register



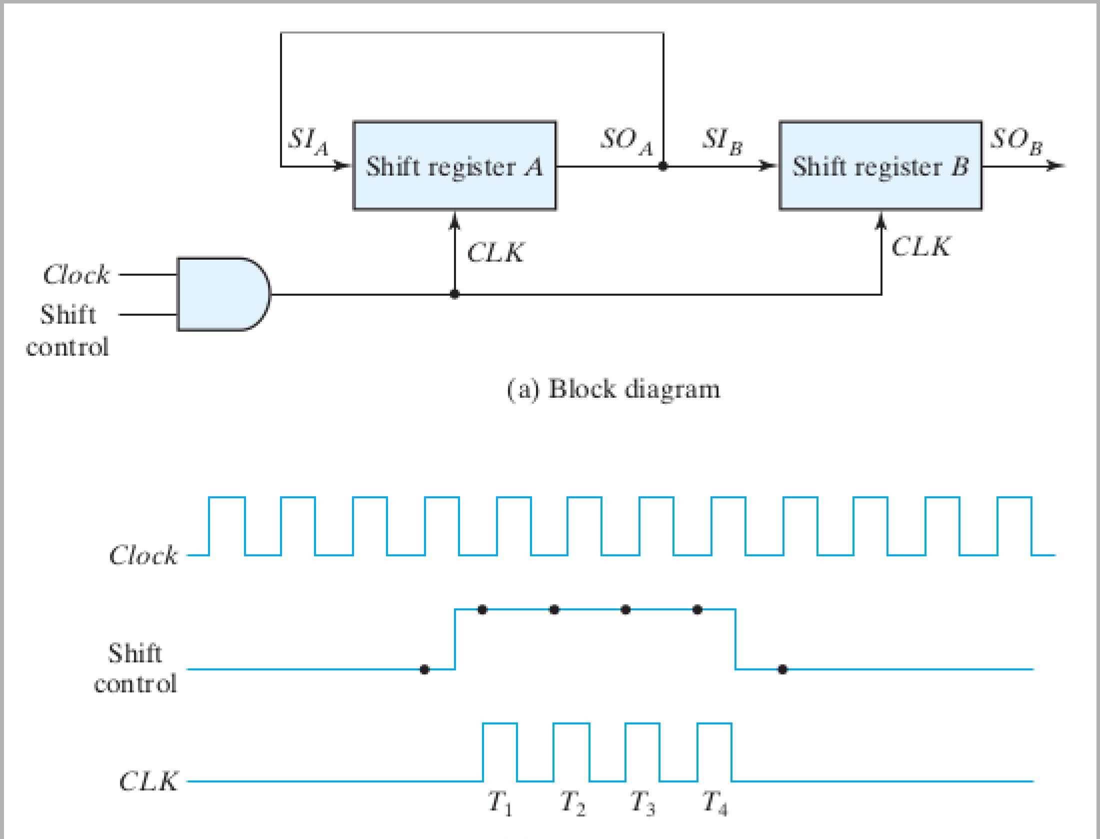
Sometimes it is necessary to control the shift so that it occurs only with certain pulses, but not with others.

This is best done by recirculating the output of each register cell back through a two-channel mux whose output is connected to the input of the cell.

Serial Transfer



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



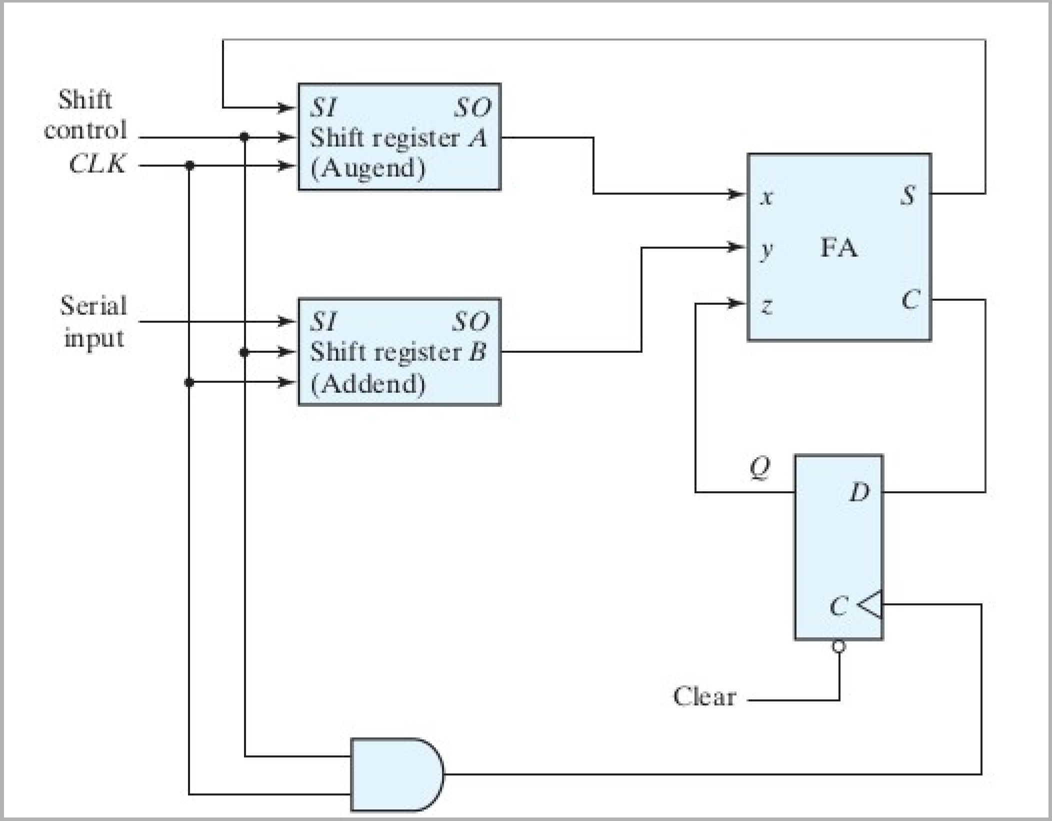
(a) Block diagram

(b) Timing diagram

The control unit that supervises the transfer of data must be designed in such a way that it enables the shift registers, through the shift control signal, for a fixed time of four clock pulses in order to pass an entire word.

Serial adder

The two binary numbers to be added serially are stored in two shift registers. 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.

Comparing the serial adder with a parallel 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.

This design is typical in serial operations because the result of a bit-time operation may depend not only on the present inputs, but also on previous inputs that must be stored in flip-flops.

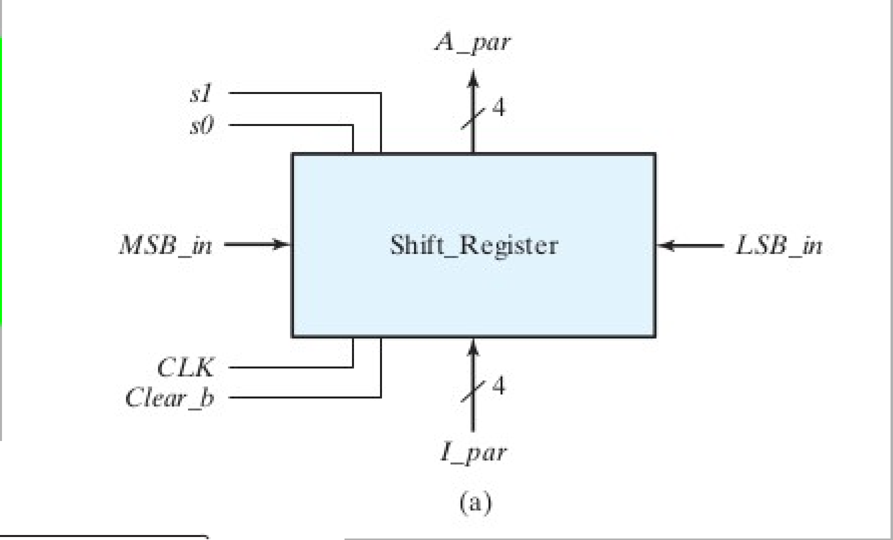
Universal Shift Register

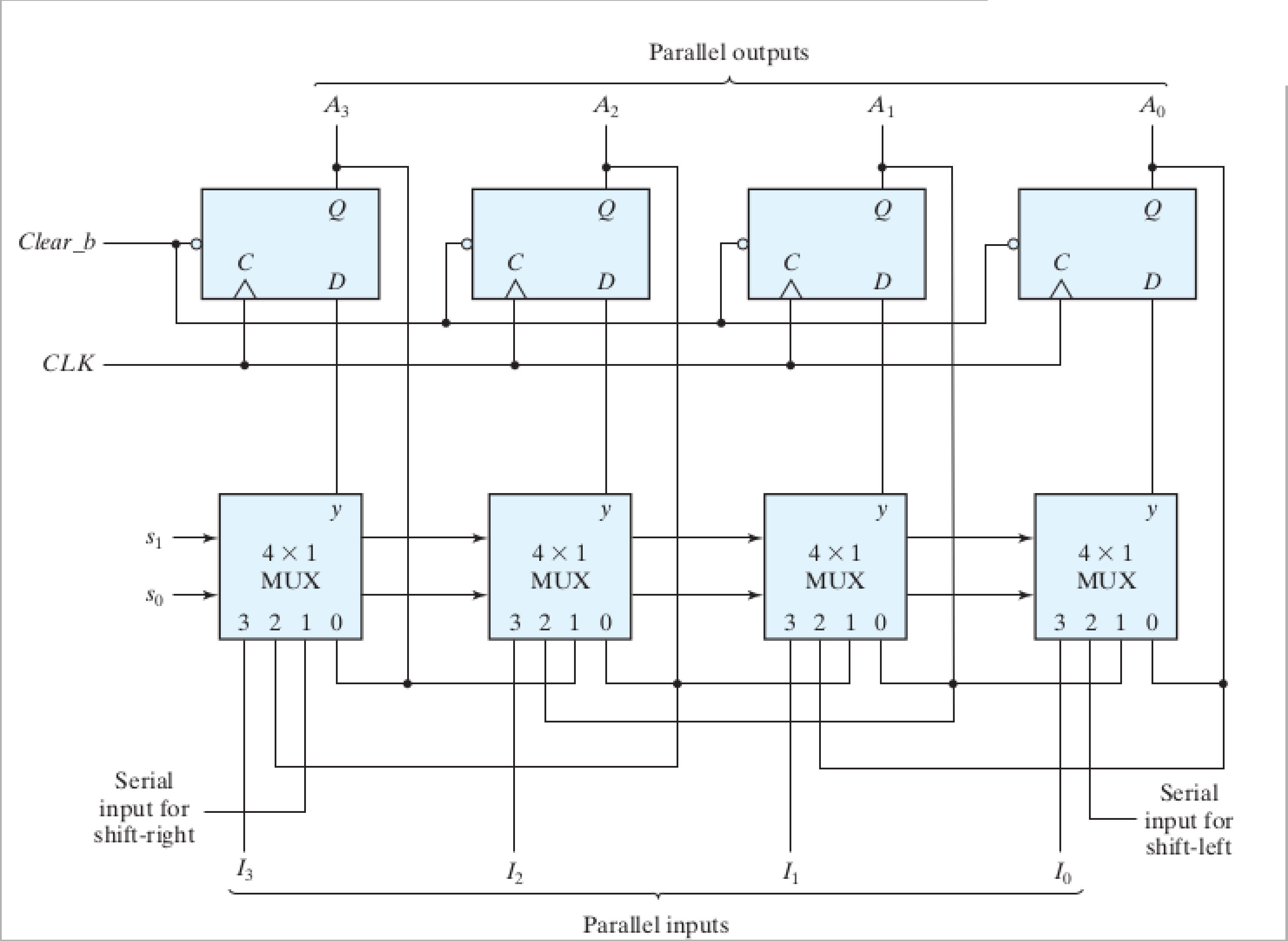
The most general shift register has the following capabilities:

1. A clear control to clear the register to 0.
2. A clock input to synchronize the operations.
3. A shift-right control to enable the shift-right operation and the serial input and output lines associated with the shift right.
4. A shift-left control to enable the shift-left operation and the serial input and output lines associated with the shift left.
5. A parallel-load control to enable a parallel transfer and the n input lines associated with the parallel transfer.
6. n parallel output lines.
7. A control state that leaves the information in the register unchanged in response to the clock.

Other shift registers may have only some of the preceding functions, with at least one shift operation.

Mode Control		
s_1	s_0	Register Operation
0	0	No change
0	1	Shift right
1	0	Shift left
1	1	Parallel load





The four multiplexers have two common selection inputs
s1 and s0 .

Input 0 in each multiplexer is selected when s1 s0 = 00, input 1 is selected when
s1 s0 = 01, and similarly for the other two inputs.

When s1 s0 = 00,
the present value of the register is applied to the D inputs of the flip-flops.

This condition forms a path from the output of each flip-flop into the input of the same flip-flop, so that
the output recirculates to the input in this mode of operation.

The next clock edge transfers into each flip-flop the binary value it held previously,
and no change of state occurs.

When s1 s0 = 01, terminal 1 of the multiplexer inputs has a path to the D inputs of the flip-flops. This causes a shift-right operation, with the serial input transferred into flip-flop A 3

When s1 s0 = 10, a shift-left operation results, with the other serial input going into flip-flop A 0 .

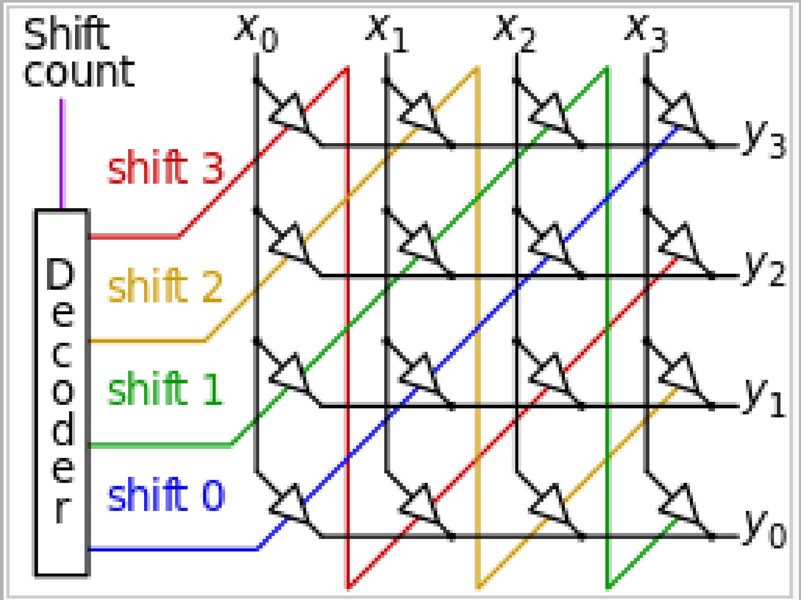
Finally, when s1 s0 = 11, the binary information on the parallel input lines is transferred into the register simultaneously during the next clock edge.

Note that data enters MSB_in for a shift-right operation and enters LSB_in for a shift-left operation.

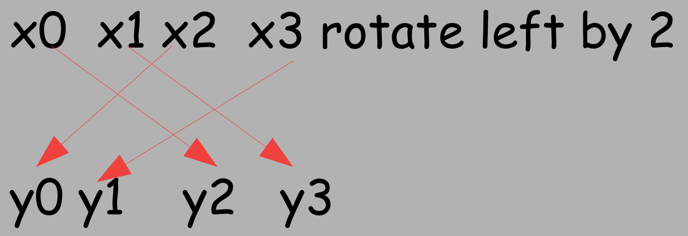
Clear_b is an active-low signal that clears all of the flip-flops.

A barrel shifter is a digital circuit that can rotate or shift a data word by a specified number of bits in one clock cycle.

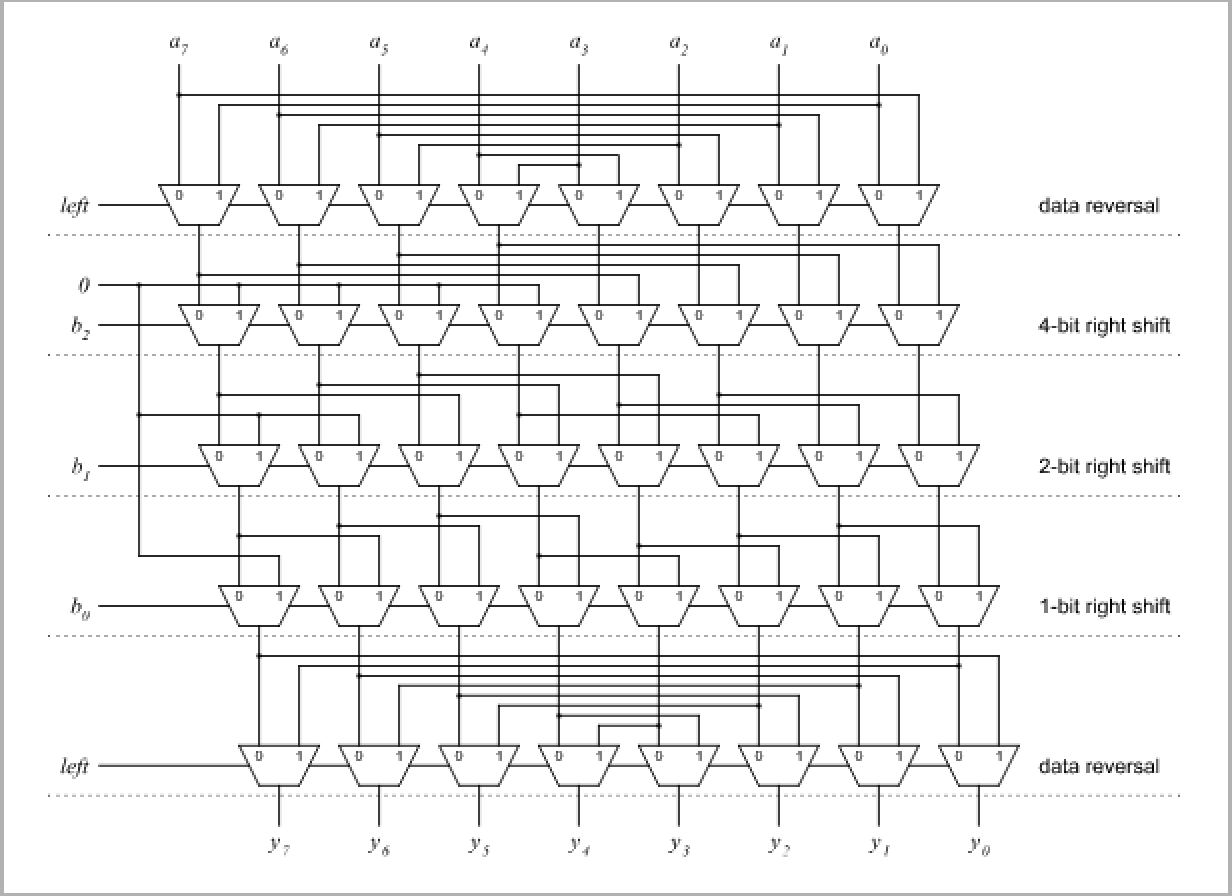
It can be implemented as a sequence of multiplexers



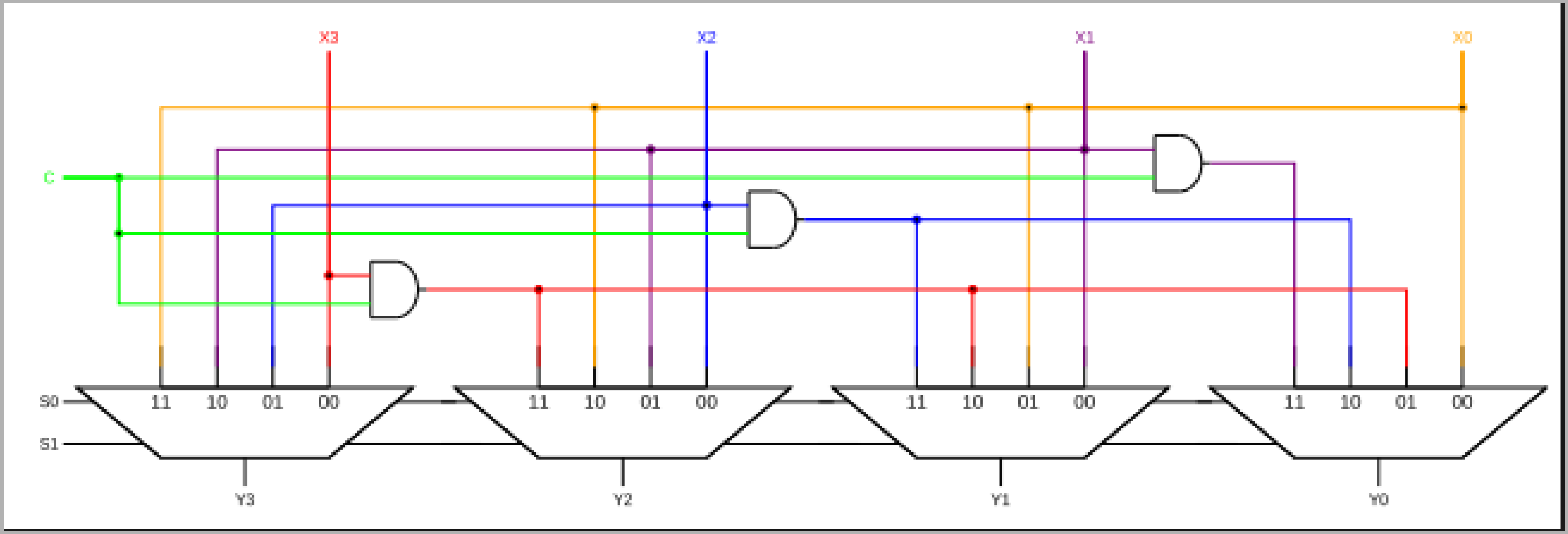
Schematic of a 4-bit crossbar barrel shifter. *x* denotes input bits and *y* denotes output bits.



A right shifter can be extended to also perform left shift operations by adding a row of n multiplexors both before and after the right shifter When a left shift operation is performed, these multiplexors reverse the data into and out of the right shifter. When a right shift operation is performed, the data into and out of the shifter is not changed.

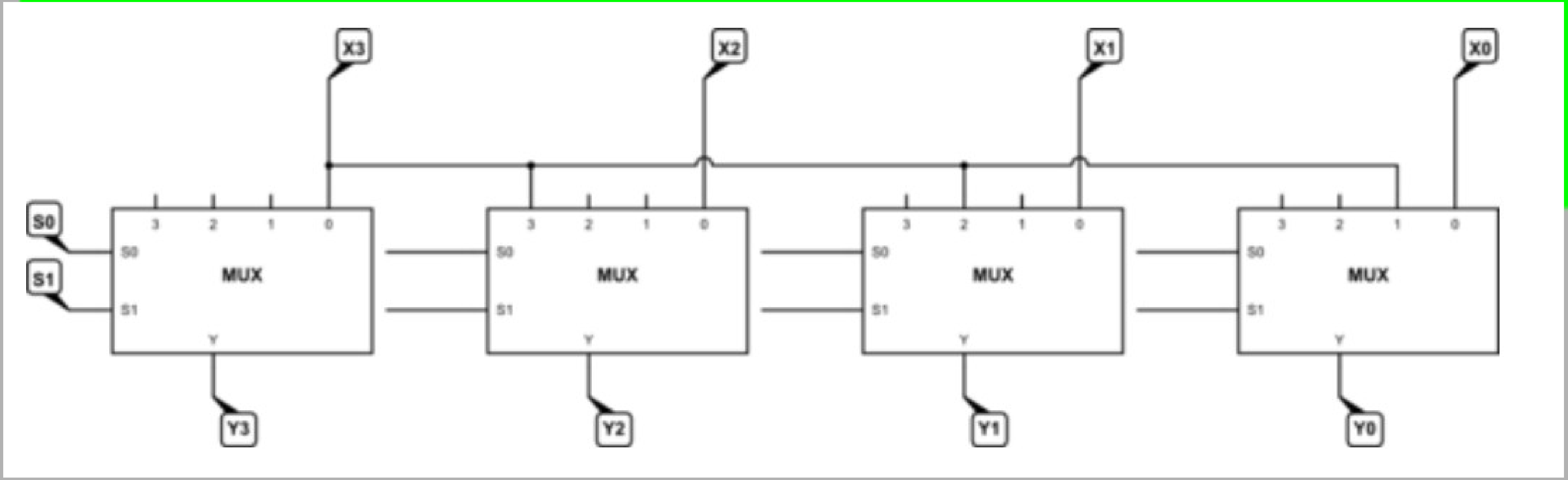


Shift vs Rotate



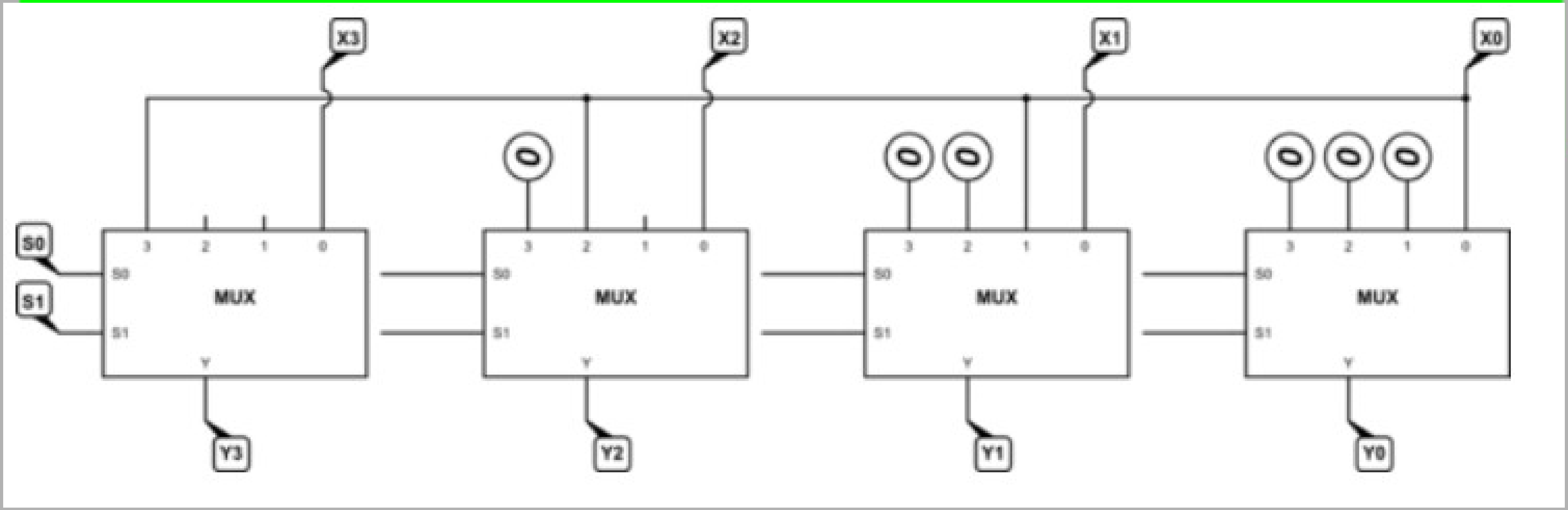
If c=0 then it's a shift, c=1 a rotate

S0	S1	
0	0	no change
0	1	shift left 1 (or rotate)
1	0	shift left 2 (or rotate)
1	1	shift left 3 (or rotate)

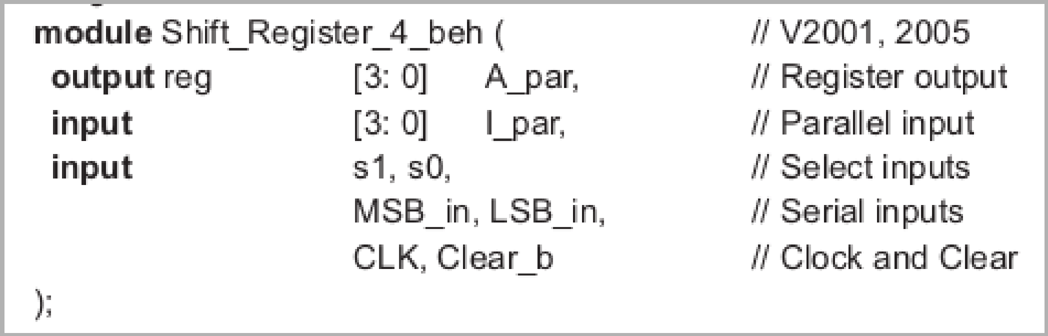


S=00 that each of the input bits are unshifted and unrotated. Each is simply propagated to their directly associated output wires without complexity.

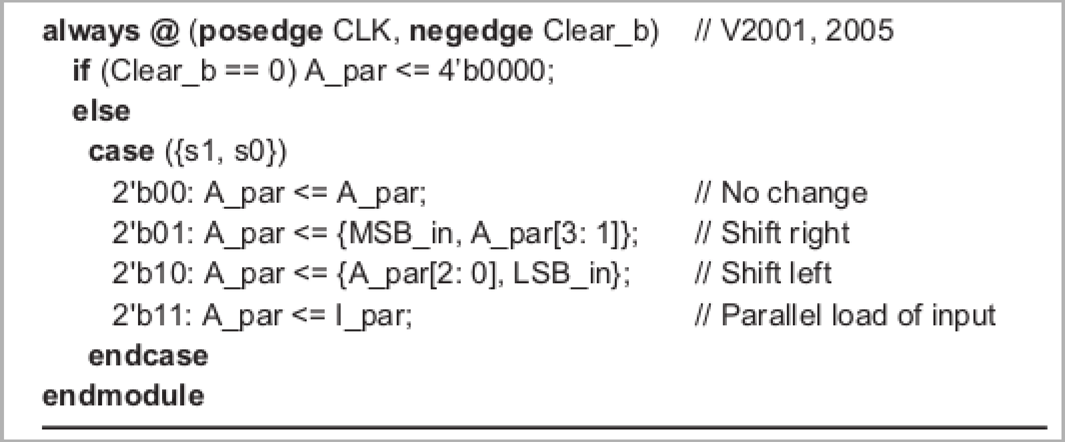
If S=1 then X3 appears at Y0. This would be correct for a left-rotate-by-1 case.
If S=2 then X3 appears at Y1. This would be correct for a left-rotate-by-2 case.
If S=3 then X3 appears at Y2. This would be correct for a left-rotate-by-3 case.



A barrel shifter is designed to perform a left rotate by S. Setting C=0 doesn't change the rotate behaviour. All it does is cause the barrel shifter to mask off (to 0) any bits shifted off of the left end before they are rotated.



```
module Shift_Register_4_beh (                         // V2001, 2005
  output reg           [3: 0]     A_par,              // Register output
  input                [3: 0]     I_par,              // Parallel input
  input                s1, s0,                        // Select inputs
                       MSB_in, LSB_in,                // Serial inputs
                       CLK, Clear_b                   // Clock and Clear
);
```



```
always @ (posedge CLK, negedge Clear_b)    // V2001, 2005
   if (Clear_b == 0) A_par <= 4'b0000;
   else
     case ({s1, s0})
       2'b00: A_par <= A_par;                  // No change
       2'b01: A_par <= {MSB_in, A_par[3: 1]};  // Shift right
       2'b10: A_par <= {A_par[2: 0], LSB_in};  // Shift left
       2'b11: A_par <= I_par;                  // Parallel load of input
     endcase
endmodule
```