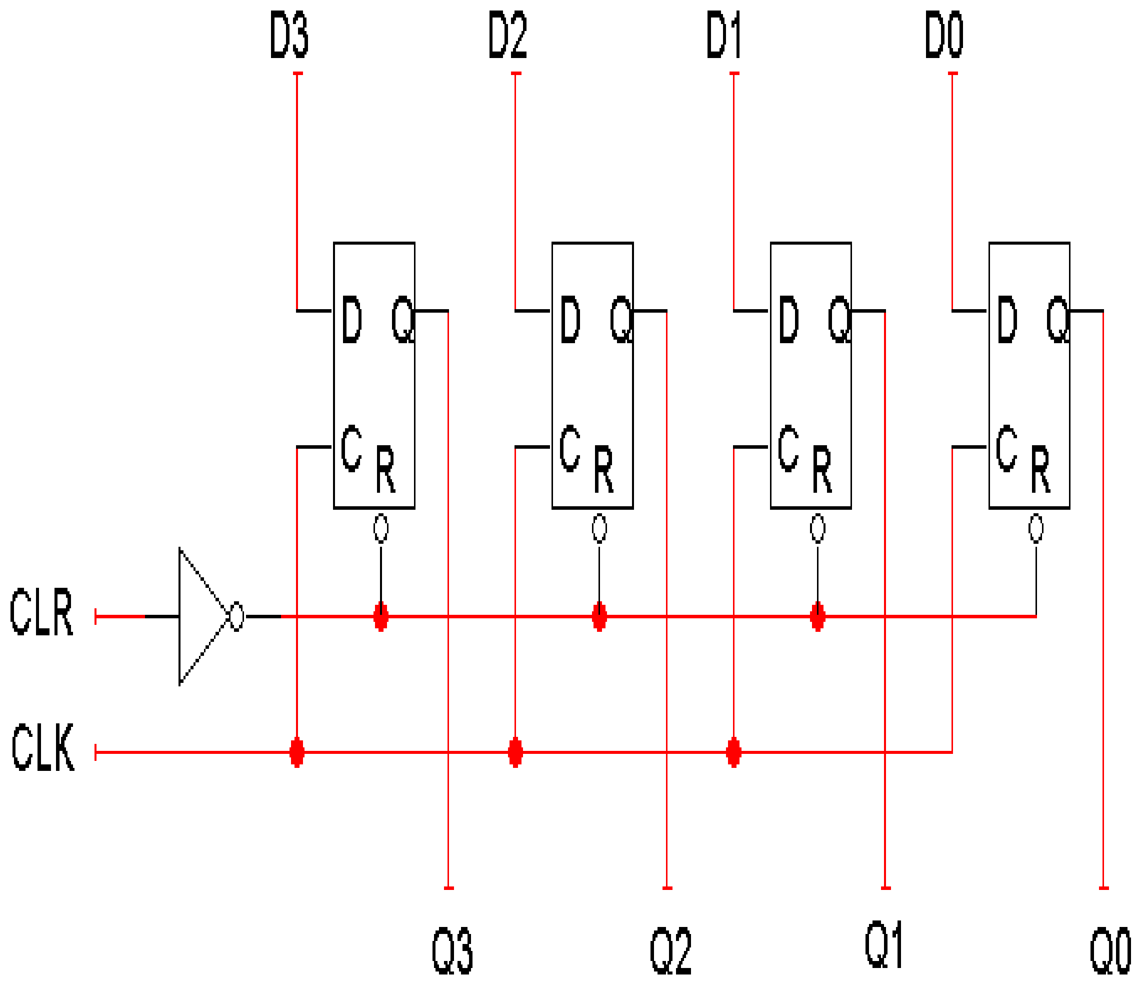
UNIT -V

Registers and Counters

A basic register

- Basic registers are easy to build. We can store multiple bits just by putting a bunch of flip-flops together!
- A 4-bit register is given below
  - This register uses D flip-flops, so it's easy to store data without worrying about flip-flop input equations
  - All the flip-flops share a common CLK and CLR signal



Shift Registers

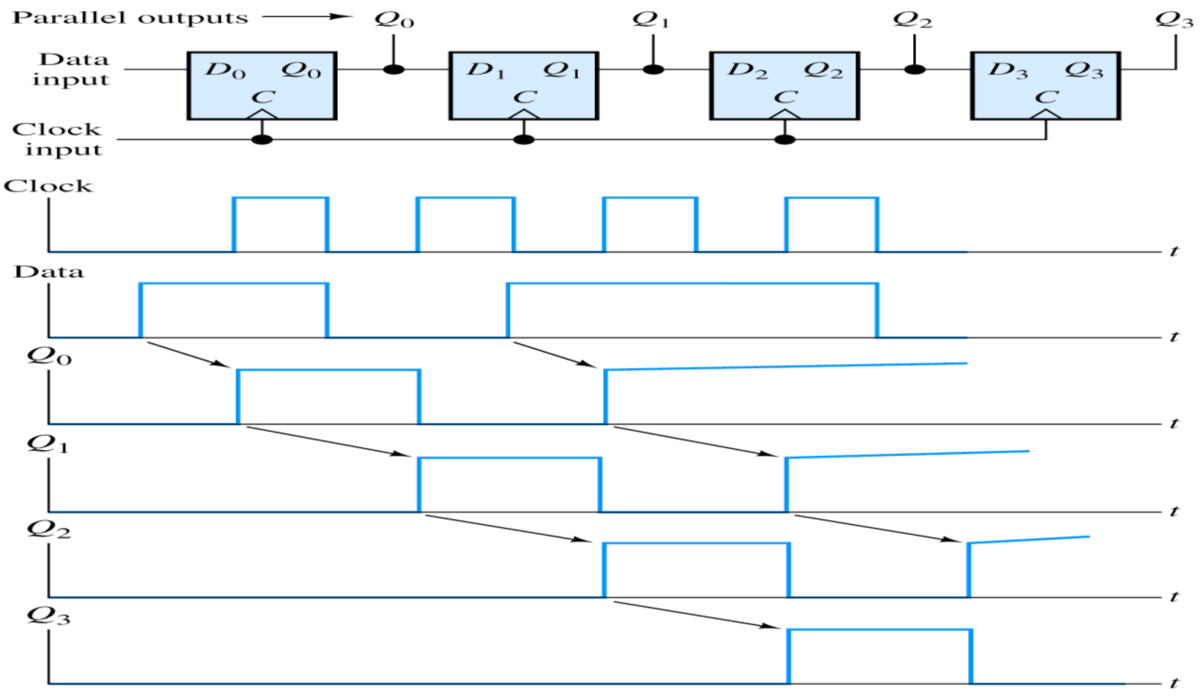


Figure 7.48 Serial-input parallel-output shift register.

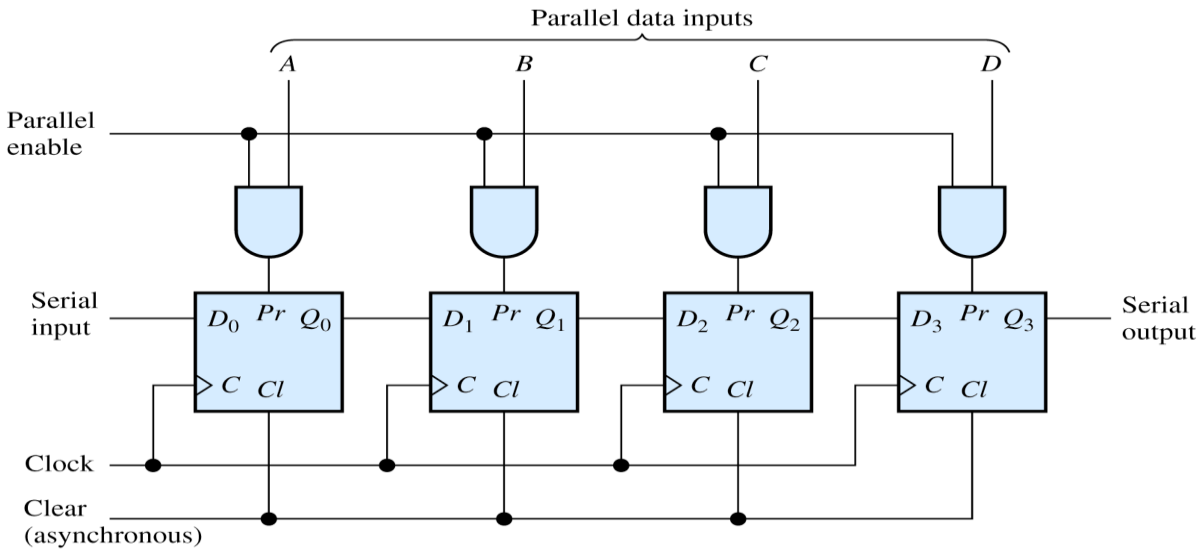

Parallel data inputs
A
B
C
D
Parallel enable
Serial input
D_0 Pr Q_0 C Cl
D_1 Pr Q_1 C Cl
D_2 Pr Q_2 C Cl
D_3 Pr Q_3 C Cl
Serial output
Clock
Clear (asynchronous)

Figure 7.49 Parallel-input serial-output shift register.

Counters:

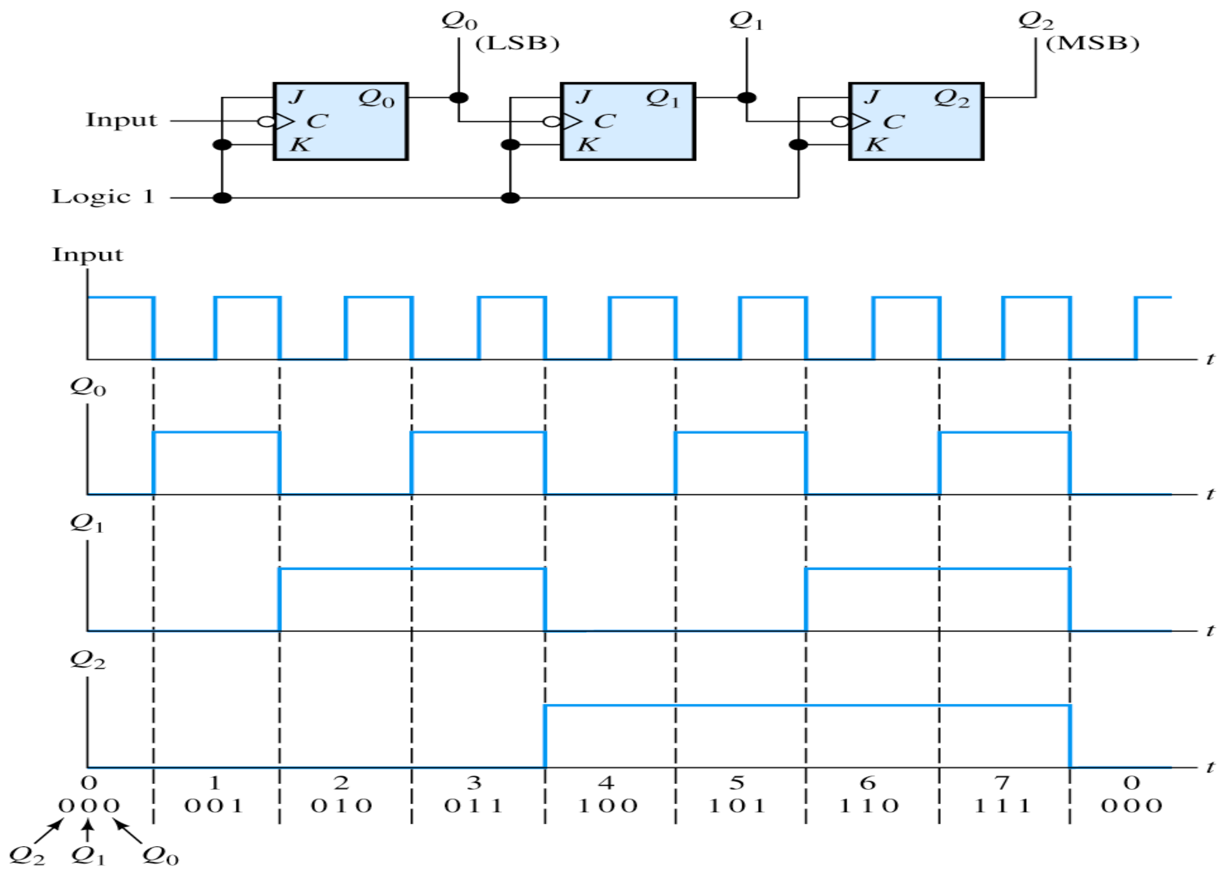


Figure 7.50 Ripple counter.

Introducing counters

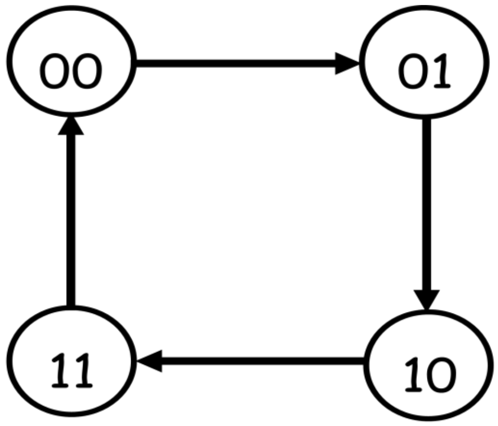
- Counters are a specific type of sequential circuit
- The state serves as the "output" (Moore)
- A counter that follows the binary number sequence is called a binary counter
  - n-bit binary counter: n flip-flops, count in binary from 0 to 2^n-1
- Counters are available in two types:
  - Synchronous Counters
  - Ripple Counters
- Synchronous Counters:

A common clock signal is connected to the C input of each flip-flop

Synchronous Binary Up Counter

- The output value increases by one on each clock cycle
- After the largest value, the output “wraps around” back to 0

Using two bits, we’d get something like this:



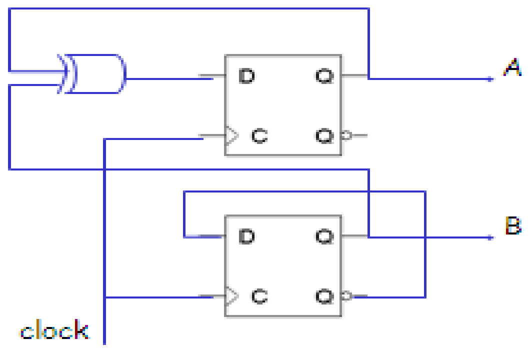
Present State		Next State	
A	B	A	B
0	0	0	1
0	1	1	0
1	0	1	1
1	1	0	0

Synchronous Binary Up Counter

Present State		Next State	
A	B	A	B
0	0	0	1
0	1	1	0
1	0	1	1
1	1	0	0

D1= A’B + AB’

D0= B’



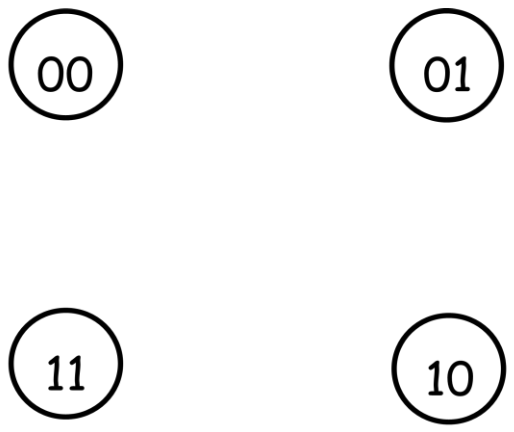
Synch Binary Up/Down Counter

- 2-bit Up/Down counter
  - Counter outputs will be 00, 01, 10 and 11
  - There is a single input, X.

> X= 0, the counter counts up

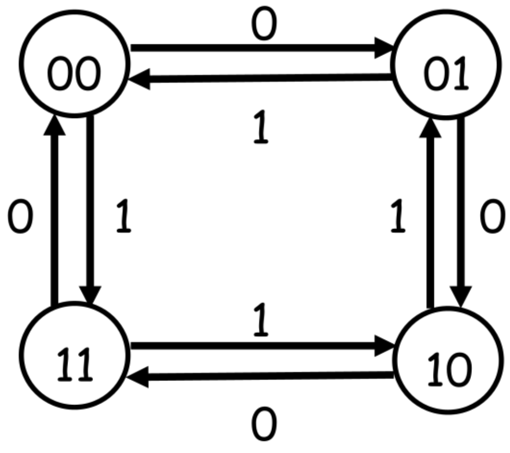
> X= 1, the counter counts down

- We'll need two flip-flops again. Here are the four possible states:



The complete state diagram and table

- Here's the complete state diagram and state table for this circuit



Present State		Inputs	Next State	
Q_1	Q_0	X	Q_1	Q_0
0	0	0	0	1
0	0	1	1	1
0	1	0	1	0
0	1	1	0	0
1	0	0	1	1
1	0	1	0	1
1	1	0	0	0
1	1	1	1	0

D flip-flop inputs

- If we use D flip-flops, then the D inputs will just be the same as the desired next states
- Equations for the D flip-flop inputs are shown at the right

Why does $D_0 = Q_0'$ make sense?

Present State		Inputs	Next State	
Q_1	Q_0	X	Q_1	Q_0
0	0	0	0	1
0	0	1	1	1
0	1	0	1	0
0	1	1	0	0
1	0	0	1	1
1	0	1	0	1
1	1	0	0	0
1	1	1	1	0

⊕ X

			Q_0	
	0	1	0	1
Q_1	1	0	1	0
		X	X	

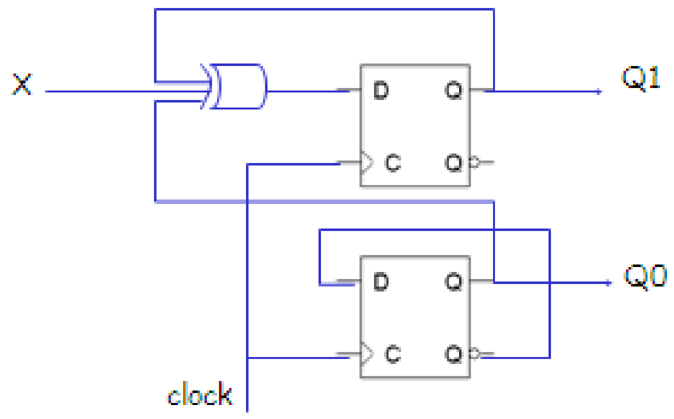
$D_1 = Q_1 \oplus Q_0$

			Q_0	
	1	1	0	0
Q_1	1	1	0	0
		X	X	

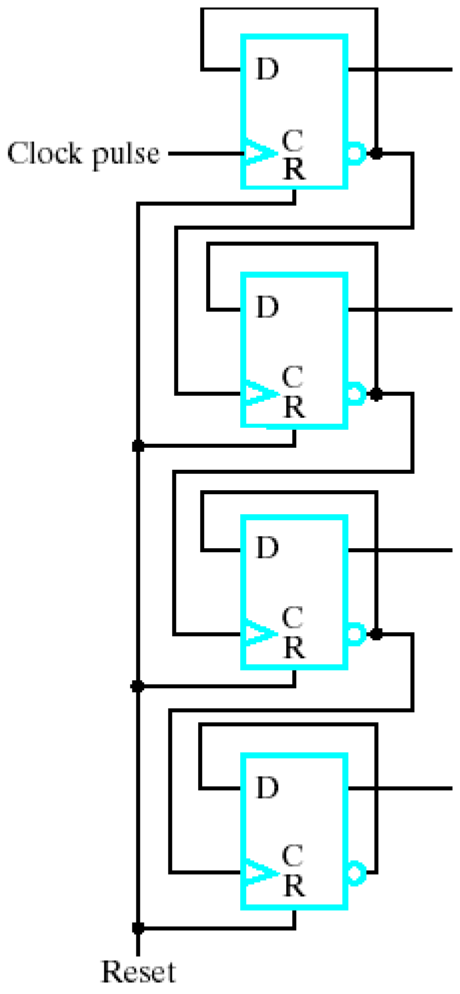
$D_0 = Q_0^1$

Synchronous Binary Up/Down Counter

Synchronous Binary Up/Down Counter



Ripple Counter



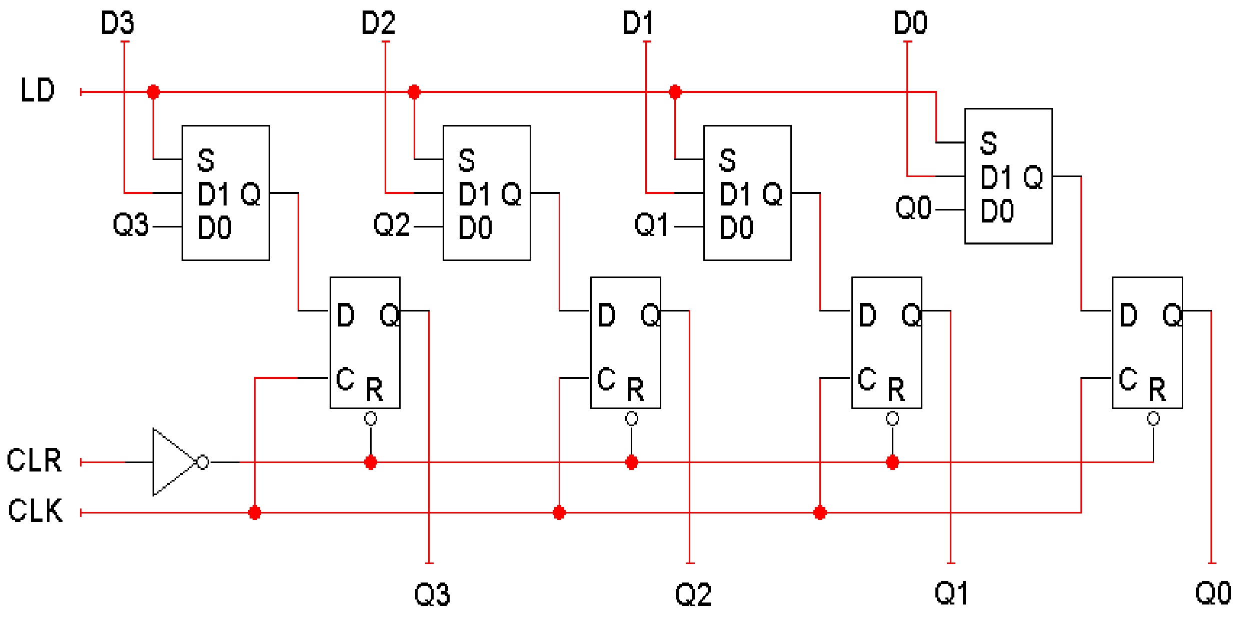
Upward Counting Sequence

Q_3	Q_2	Q_1	Q_0
0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1
1	0	0	0
1	0	0	1
1	0	1	0
1	0	1	1
1	1	0	0
1	1	0	1
1	1	1	0
1	1	1	1

Simple, yet asynchronous circuits ....

Register with parallel load

- When LD = 0, the flip-flop inputs are Q_3-Q_0, so each flip-flop just keeps its current value
- When LD = 1, the flip-flop inputs are D_3-D_0, and this new value is "loaded" into the register.



Universal shift register:

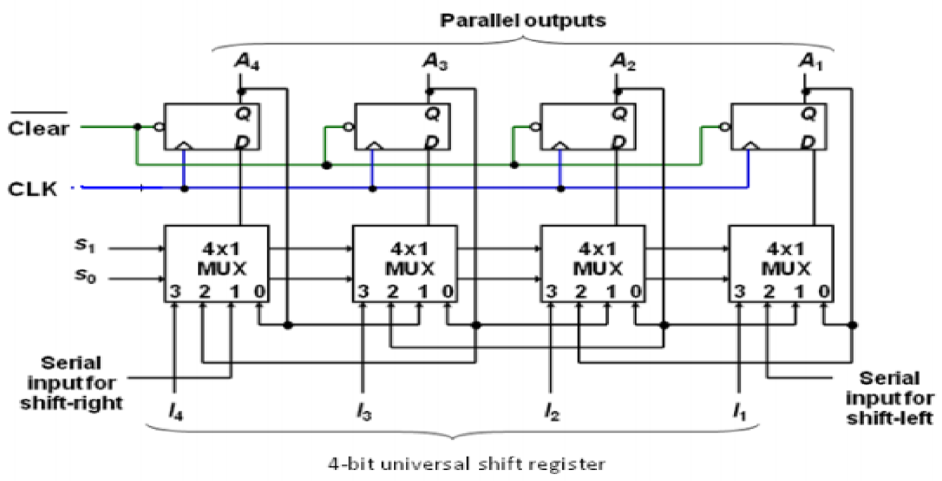
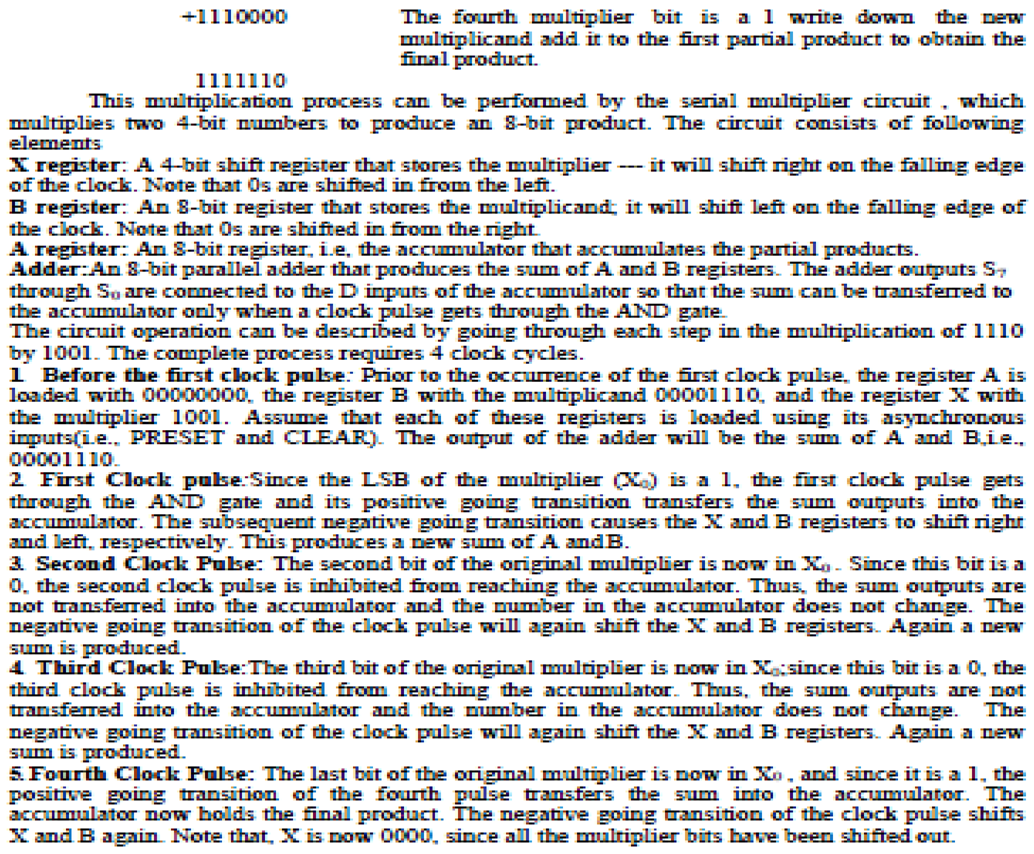

Parallel outputs
A4
A3
A2
A1
Clear
CLK
Q
D
s1
s0
4x1 MUX
3 2 1 0
Serial input for shift-right
I4
I3
I2
I1
Serial input for shift-left
4-bit universal shift register

Figure: logic diagram 4-bit universal shift register

Binary Multiplier:



+1110000	The fourth multiplier bit is a 1 write down the new multiplicand add it to the first partial product to obtain the final product.
1111110	

This multiplication process can be performed by the serial multiplier circuit , which multiplies two 4-bit numbers to produce an 8-bit product. The circuit consists of following elements

X register: A 4-bit shift register that stores the multiplier --- it will shift right on the falling edge of the clock. Note that 0s are shifted in from the left.

B register: An 8-bit register that stores the multiplicand; it will shift left on the falling edge of the clock. Note that 0s are shifted in from the right.

A register: An 8-bit register, i.e, the accumulator that accumulates the partial products.

Adder:An 8-bit parallel adder that produces the sum of A and B registers. The adder outputs S_7 through S_0 are connected to the D inputs of the accumulator so that the sum can be transferred to the accumulator only when a clock pulse gets through the AND gate.

The circuit operation can be described by going through each step in the multiplication of 1110 by 1001. The complete process requires 4 clock cycles.

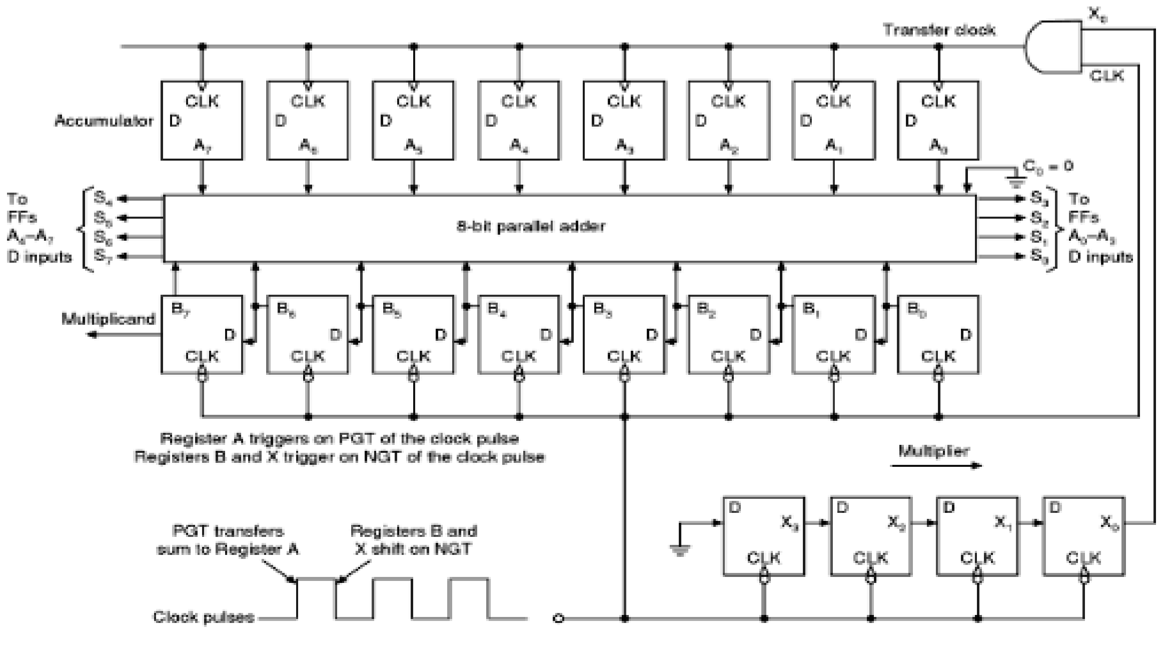
1. **Before the first clock pulse**: Prior to the occurrence of the first clock pulse, the register A is loaded with 00000000, the register B with the multiplicand 00001110, and the register X with the multiplier 1001. Assume that each of these registers is loaded using its asynchronous inputs(i.e., PRESET and CLEAR). The output of the adder will be the sum of A and B,i.e., 00001110.
2. **First Clock pulse**:Since the LSB of the multiplier (X_0) is a 1, the first clock pulse gets through the AND gate and its positive going transition transfers the sum outputs into the accumulator. The subsequent negative going transition causes the X and B registers to shift right and left, respectively. This produces a new sum of A and B.
3. **Second Clock Pulse:** The second bit of the original multiplier is now in X_0 . Since this bit is a 0, the second clock pulse is inhibited from reaching the accumulator. Thus, the sum outputs are not transferred into the accumulator and the number in the accumulator does not change. The negative going transition of the clock pulse will again shift the X and B registers. Again a new sum is produced.
4. **Third Clock Pulse**:The third bit of the original multiplier is now in X_0;since this bit is a 0, the third clock pulse is inhibited from reaching the accumulator. Thus, the sum outputs are not transferred into the accumulator and the number in the accumulator does not change. The negative going transition of the clock pulse will again shift the X and B registers. Again a new sum is produced.
5. **Fourth Clock Pulse:** The last bit of the original multiplier is now in X_0 , and since it is a 1, the positive going transition of the fourth pulse transfers the sum into the accumulator. The accumulator now holds the final product. The negative going transition of the clock pulse shifts X and B again. Note that, X is now 0000, since all the multiplier bits have been shifted out.

Binary Multipliers:

In binary multiplication by the paper and pencil method, is modified somewhat in digital machines because a binary adder can add only two binary numbers at a time.
In a binary multiplier, instead of adding all the partial products at the end, they are added two at a time and their sum accumulated in a register (the accumulator register). In addition, when the multiplier bit is a 0,0s are not written down and added because it does not affect the final result. Instead, the multiplicand is shifted left by one bit.

The multiplication of 1110 by 1001 using this process is

Multiplicand	1110	
Multiplier	1001	
	1110	The LSB of the multiplier is a 1; write down the multiplicand; shift the multiplicand one position to the left (1 1 1 0 0)
	1110	The second multiplier bit is a 0; write down the previous result 1110; shift the multiplicand to the left again (1 1 1 0 0 0)



Transfer clock
X_0
CLK
Accumulator
D
CLK
A_7
A_6
A_5
A_4
A_3
A_2
A_1
A_0
$C_0 = 0$
To FFs A_4–A_7 D inputs
S_4 S_5 S_6 S_7
8-bit parallel adder
S_3 S_2 S_1 S_0
To FFs A_0–A_3 D inputs
Multiplicand
B_7 B_6 B_5 B_4 B_3 B_2 B_1 B_0
Register A triggers on PGT of the clock pulse
Registers B and X trigger on NGT of the clock pulse
Multiplier
X_3 X_2 X_1 X_0
PGT transfers sum to Register A
Registers B and X shift on NGT
Clock pulses

Figure 4.31 Logic diagram of a binary multiplier.

THE END

(.....The end of education is character....)