Registers

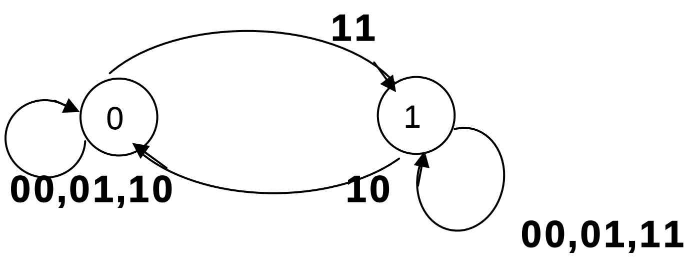
Register is a set of FFs connected in such a way as to load serial or/and parallel data and store it.

Parallel Registers:

Example:
Design a 3 bit parallel in-parallel out register of 3-bit register.

One part of the parallel Load register:
INPUTS: Output= Q
L= Load
I =Input
When Load =1 then data I, is loaded to the register
When Load=0 the value of Q is loaded in the register.

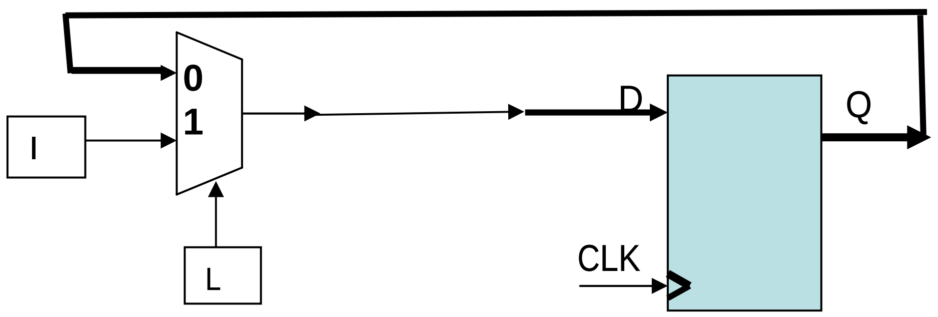
The following state diagram is then derived where input **L,I**



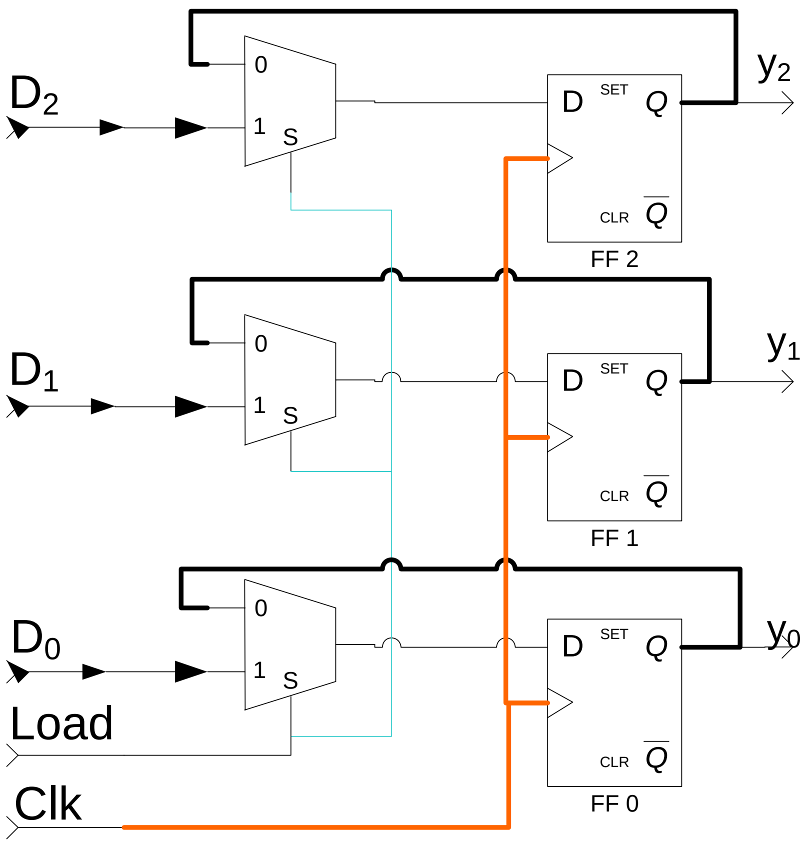
Translate into state transition table

Q	L I =00	L I=01	L I=11	L I= 10	
0	0	0	1	0	
1	1	1	1	0	

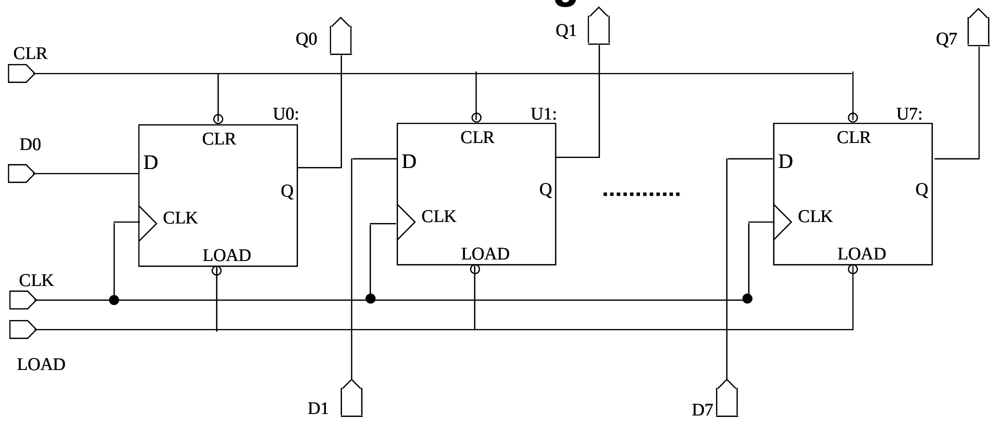
Q+ = L' Q + L I   which is MUX equation
using a D-Flip Flop, Q+ = D= = L' Q + L I



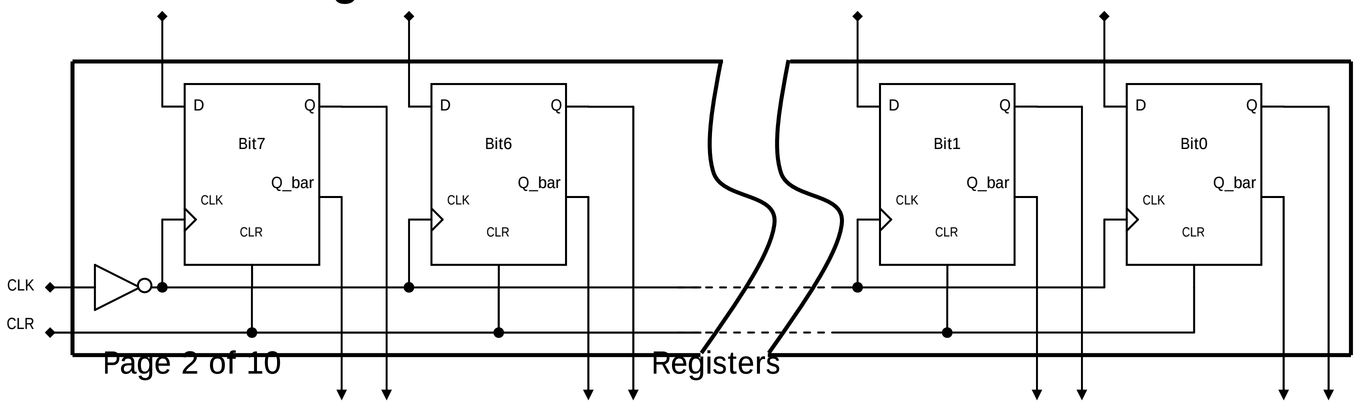
Now have multiple of the design as many bits as you like. Below is 3-bit parallel register designed with the above method.



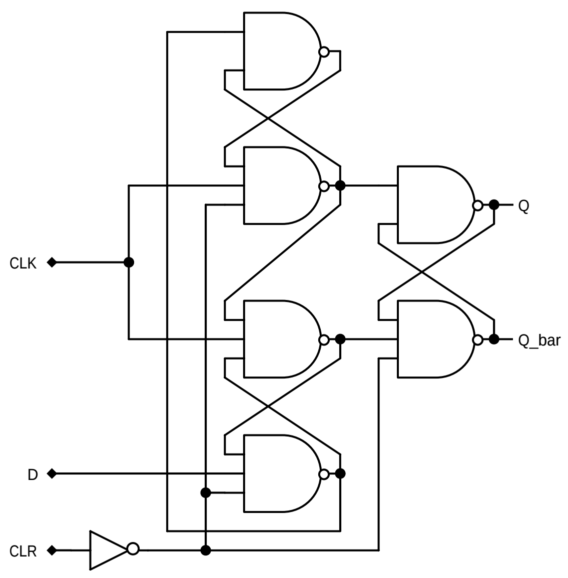
Parallel In-Parallel Out register.

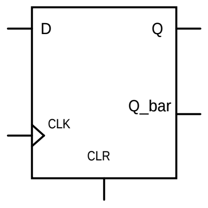


An 8-bit register

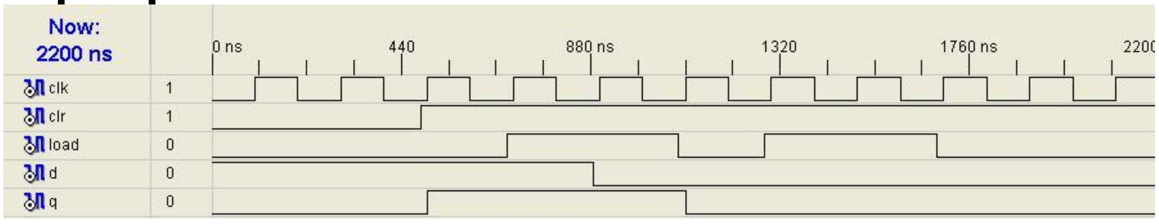


Symbols

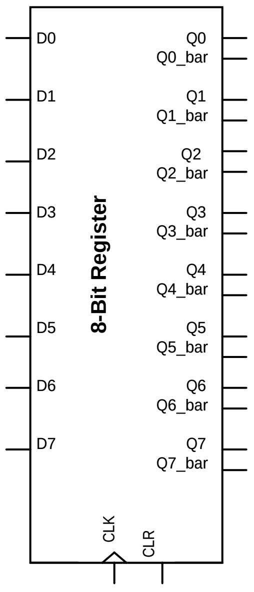




FlipFlop Simulation

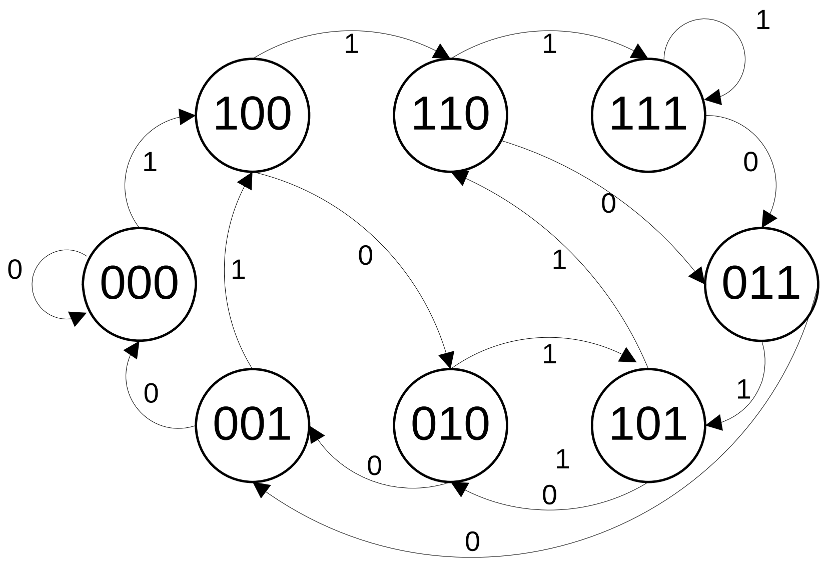


Symbol 0f 8-bit parallel load / parallel out register



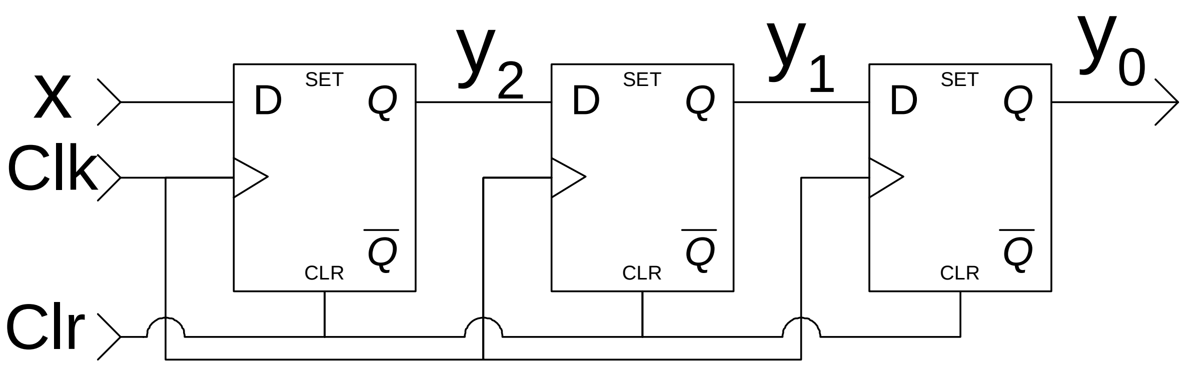
Example: Design a 3-bit L->R shift register.

→→
000 x=1 => 100 x=0 => 000
2^3 states



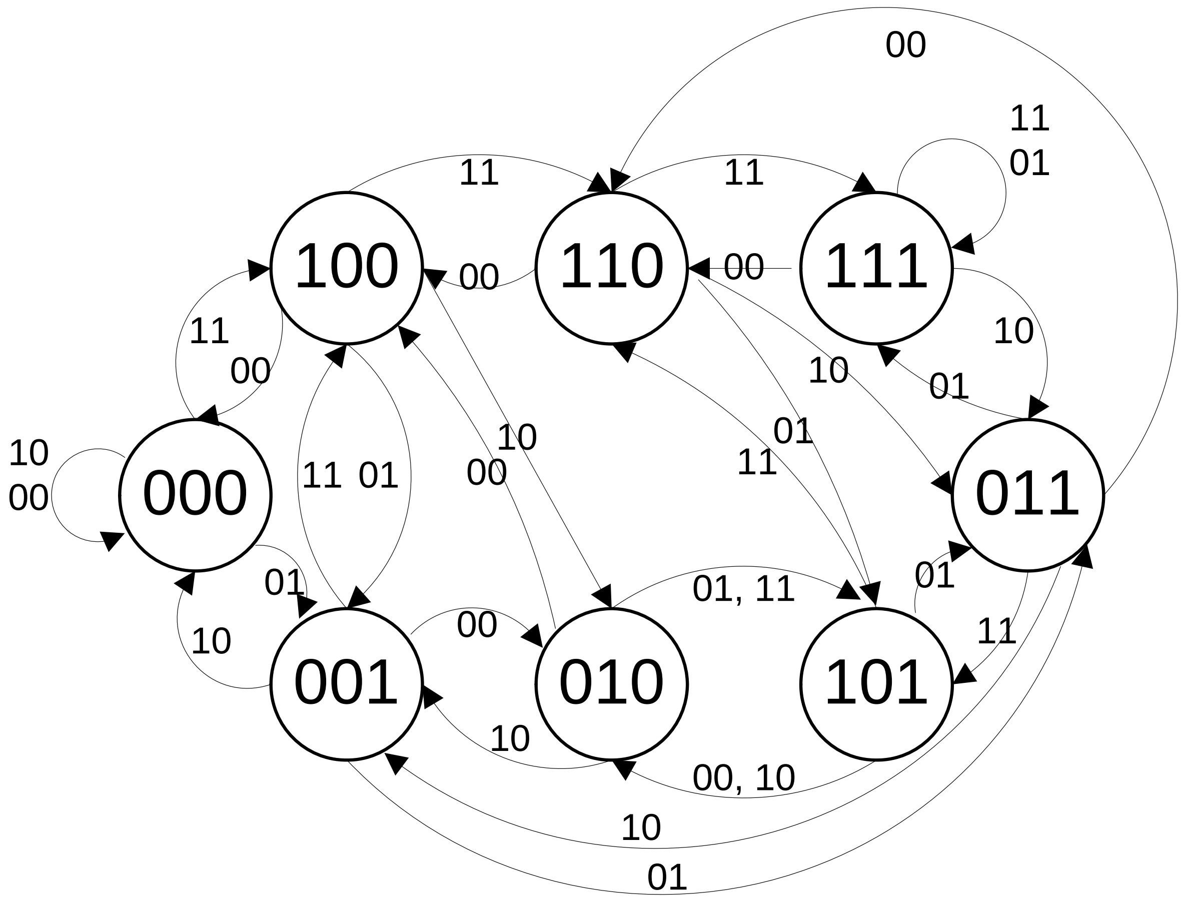
Present state			Next state					
			x=1			x=0		
y_2	y_1	y_0	y_2^+	y_1^+	y_0^+	y_2^+	y_1^+	y_0^+
0	0	0	1	0	0	0	0	0
0	0	1	1	0	0	0	0	0
0	1	0	1	0	1	0	0	1
0	1	1	1	0	1	0	0	1
1	0	0	1	1	0	0	1	0
1	0	1	1	1	0	0	1	0
1	1	0	1	1	1	0	1	1
1	1	1	1	1	1	0	1	1

For registers it is customary to use D-FFs. Therefore, using D-FFs for the above table:
$D_2 = y_2^+; D_1 = y_1^+; D_0 = y_0^+.$



Example 2.
Design 3-bit left-to-right and right-to-left shift register

Control signal:
C=1: L → R
C=0: R → L



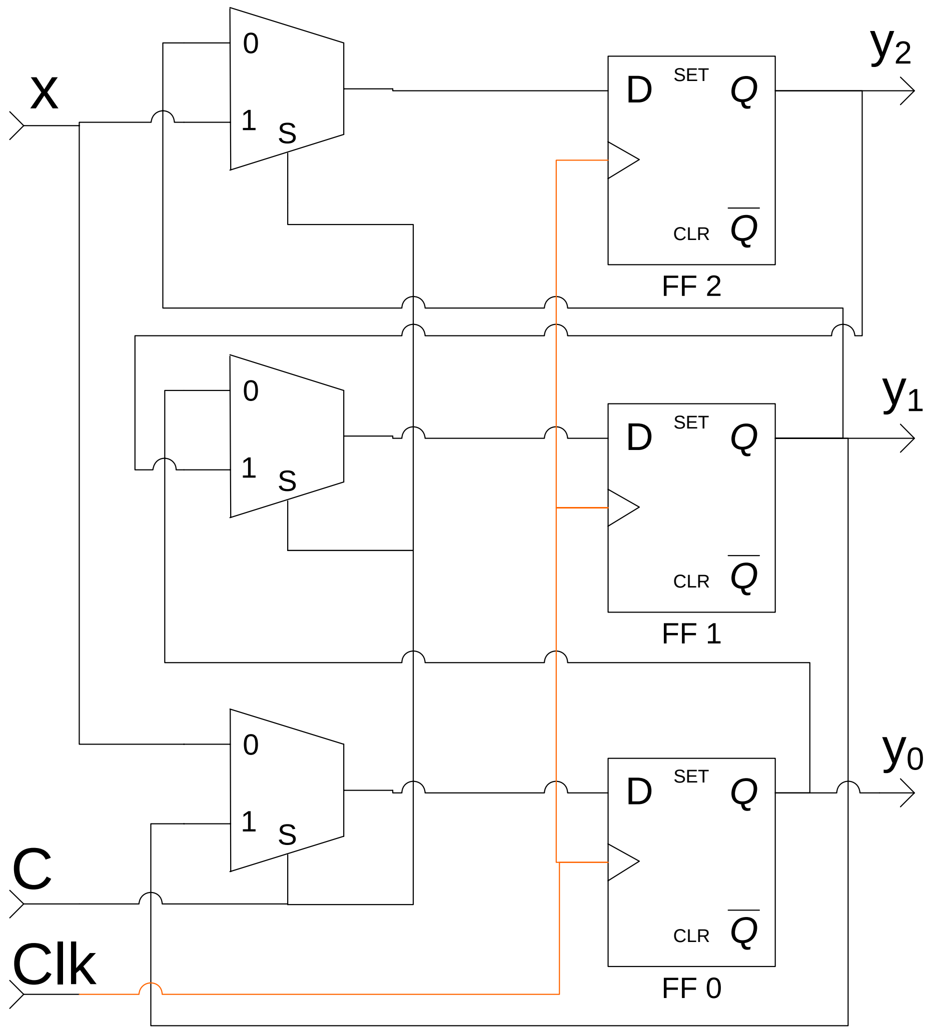
From the state diagram we will fill the state table out and follow standard implementation procedure.

Present state			Next state											
			C=1 (L→R)						C=0 (R→L)					
			x=1			x=0			x=1			x=0		
y_2	y_1	y_0	y_2^+	y_1^+	y_0^+	y_2^+	y_1^+	y_0^+	y_2^+	y_1^+	y_0^+	y_2^+	y_1^+	y_0^+
0	0	0	1	0	0	0	0	0	0	0	1	0	0	0
0	0	1	1	0	0	0	0	0	0	1	1	0	1	0
0	1	0	1	0	1	0	0	1	1	0	1	1	0	0
0	1	1	1	0	1	0	0	1	1	1	1	1	1	0
1	0	0	1	1	0	0	1	0	0	0	1	0	0	0
1	0	1	1	1	0	0	1	0	0	1	1	0	1	0
1	1	0	1	1	1	0	1	1	1	0	1	1	0	0
1	1	1	1	1	1	0	1	1	1	1	1	1	1	0

$y_2^+ = C\,x + C'\,y_1$

$y1 = Cy_2 + C'\,y_0\,'$

$y_0 = C\,y_1 + C'\,x$



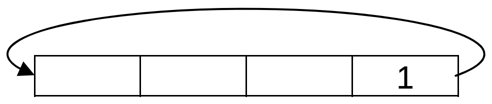
Other counters

BCD Counter:

0-1-2-3-4-5-6-7-8-9-0-1-...

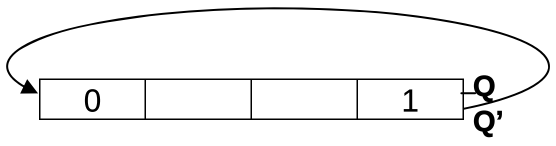
Present state	Next state
0000	0001
0001	0010
0010	0011
0011	0100
0100	0101
0101	0110
0110	0111
0111	1000
1000	1001
1001	0000
x	x

Ring counter



0001
1000
0100
0010
0001
1000

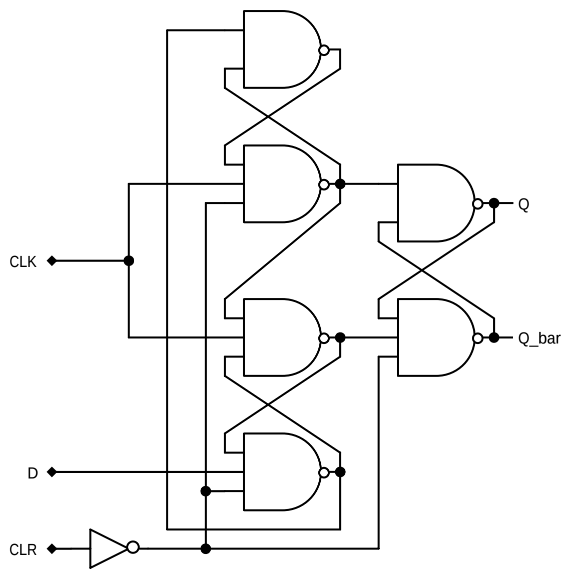
Johnson counter

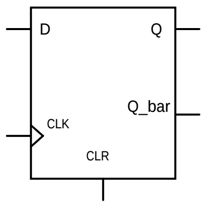


0000
1000
1100
1110
1111
0111

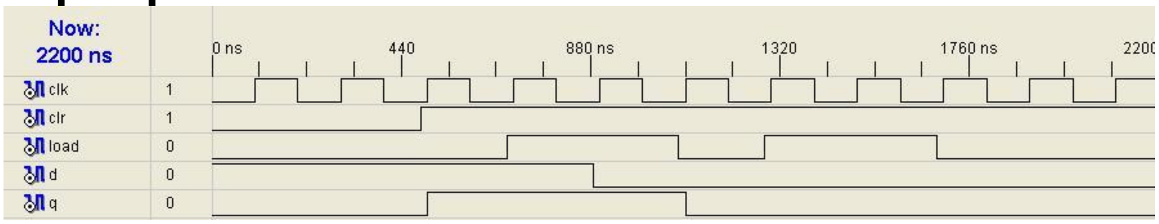
== ====General information for interested students==================

Symbols

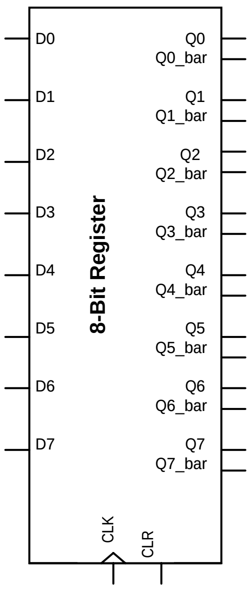




FlipFlop Simulation



Symbol 0f 8-bit parallel load / parallel out register



Simulation of some of the registers:

The 8-bit parallel load shift register is composed of multiplexers and D flip-flops, refer to figure. The multiplexers select whether to load the data or to simply pass the previous register value. If neither the load or shift enable inputs are high the registers keep their current values. If both inputs are high then the load input takes precedence. Otherwise the circuit either loads or shifts at each clock cycle with respect to the input that is high. If the circuit is shifting then a 0 is introduced at the leftmost bit.

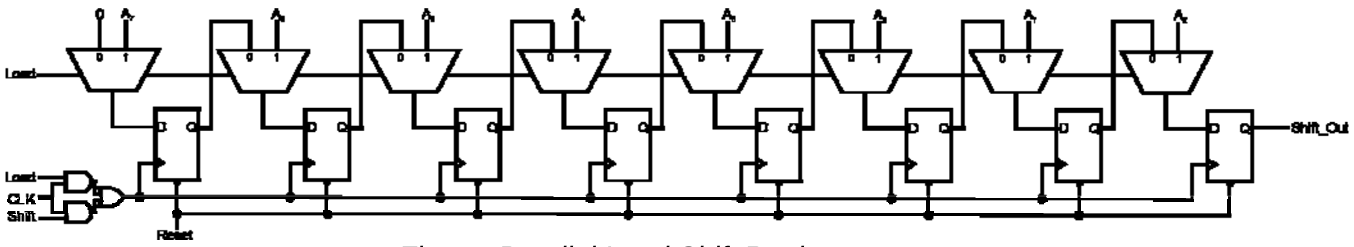
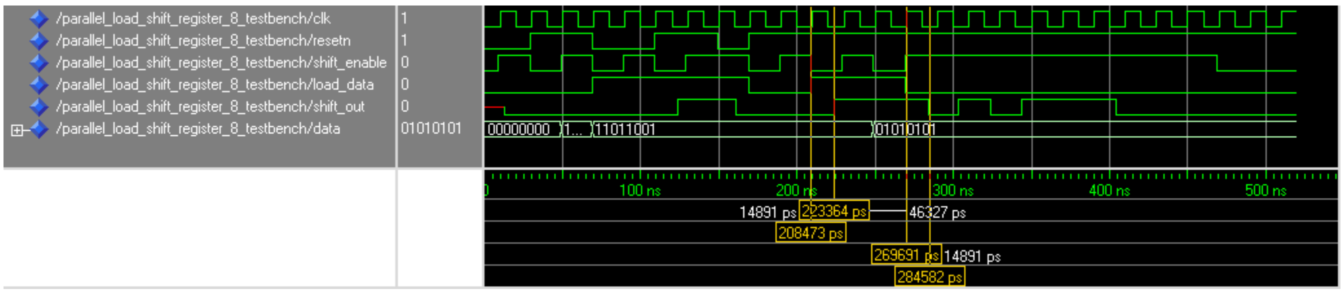


Figure Parallel Load Shift Register.

The result of simulating the circuit is shown in figure below.



```
/parallel_load_shift_register_8_testbench/clk           1
/parallel_load_shift_register_8_testbench/resetn        1
/parallel_load_shift_register_8_testbench/shift_enable  0
/parallel_load_shift_register_8_testbench/load_data     0
/parallel_load_shift_register_8_testbench/shift_out     0
/parallel_load_shift_register_8_testbench/data          01010101
```

Figure Parallel Load Shift Register Simulation.

Parallel Load Parallel Output Register

Similar to the previous register, the parallel load parallel output register is composed of multiplexers and D flip-flops. In this case, however, the circuit either loads new data or does not change its output. Refer to figure below for the schematic.

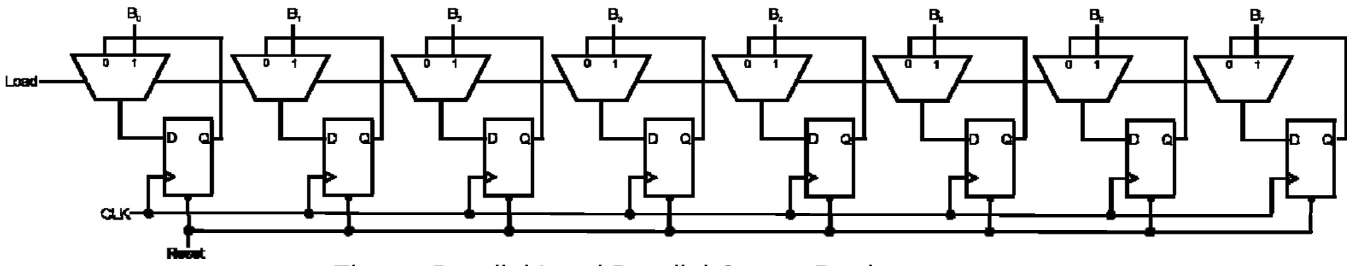
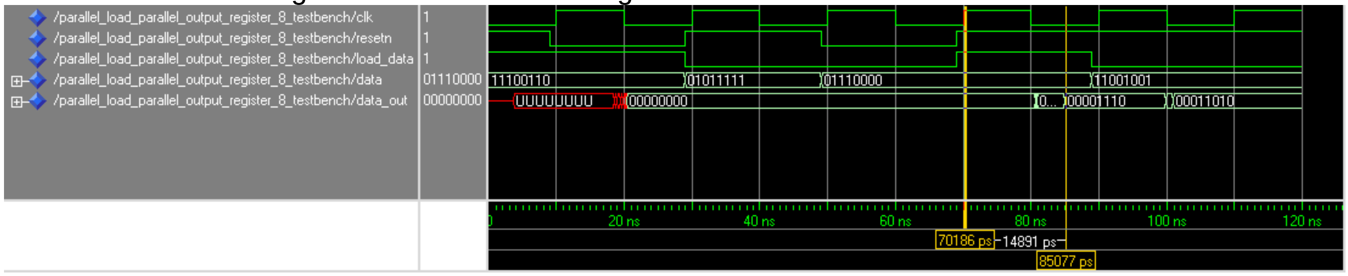


Figure Parallel Load Parallel Output Register.

The result of simulating the circuit is shown in figure below.



```
/parallel_load_parallel_output_register_8_testbench/clk        1
/parallel_load_parallel_output_register_8_testbench/resetn     1
/parallel_load_parallel_output_register_8_testbench/load_data  1
/parallel_load_parallel_output_register_8_testbench/data       01110000
/parallel_load_parallel_output_register_8_testbench/data_out   00000000
```

Figure Parallel Load Parallel Output Register Simulation.

Serial-Input Parallel Output Register

The final register circuit in the multiplier is the serial-input parallel output register. This register stores the result of each succeeding column addition and outputs the result in parallel. The circuit is very similar to the other two registers however it is configured to shift a data bit into the MSB and to output the register state when it is not shifting data as shown in its 8-bit implementation below.

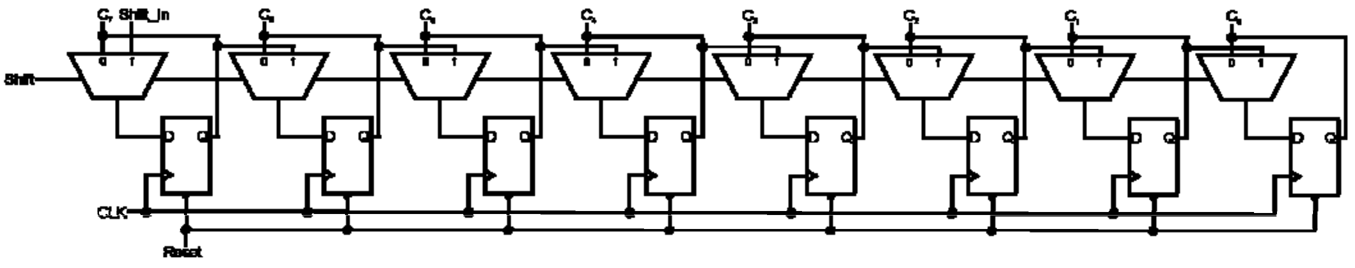
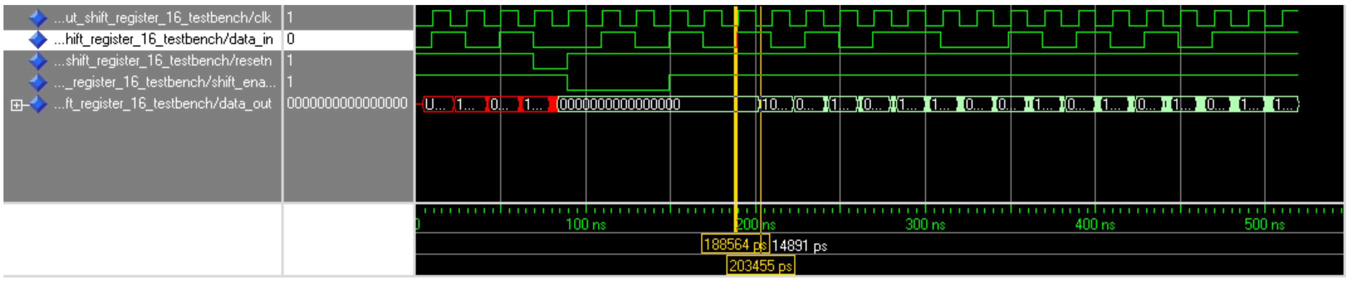


Figure Serial-Input Parallel Output Register.

The result of simulating the circuit is shown in figure below.



```
...ut_shift_register_16_testbench/clk        1
...hift_register_16_testbench/data_in        0
...shift_register_16_testbench/resetn        1
..._register_16_testbench/shift_ena...       1
...ft_register_16_testbench/data_out         0000000000000000
```

Figure Serial-Input Parallel Output Register Simulation.