ShanghaiTech University

EE 115B: Digital Circuits

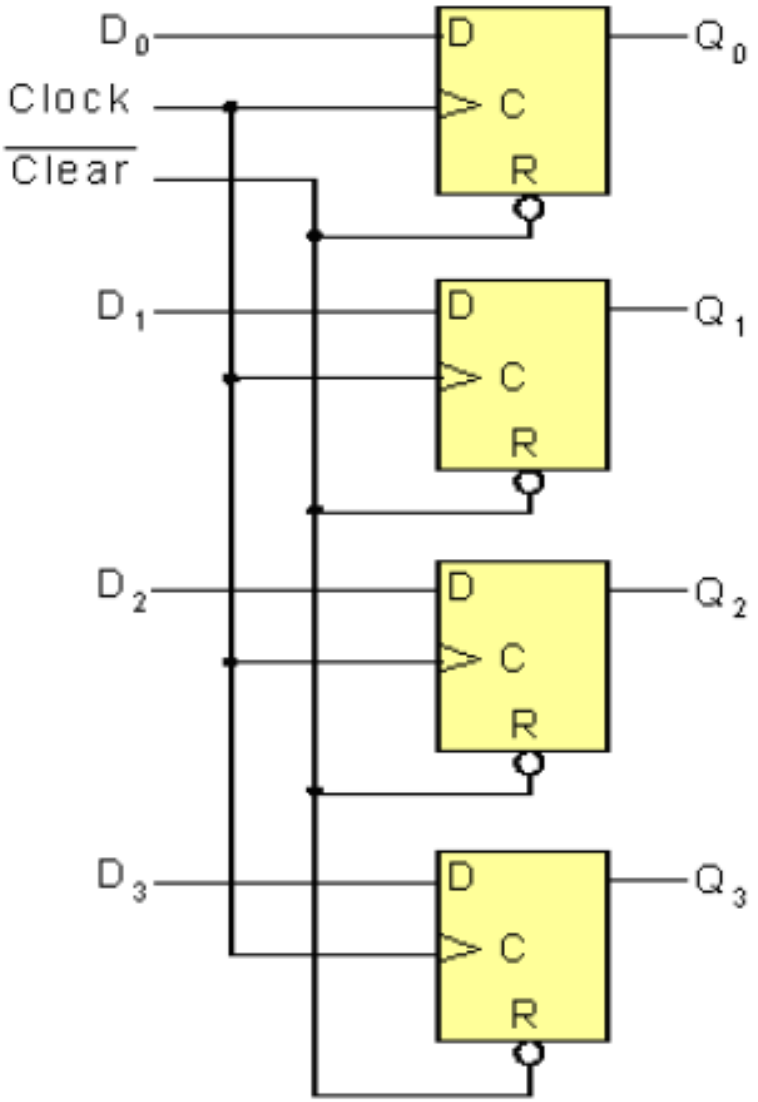
Fall 2022

Lecture 18

Hengzhao Yang
December 6, 2022

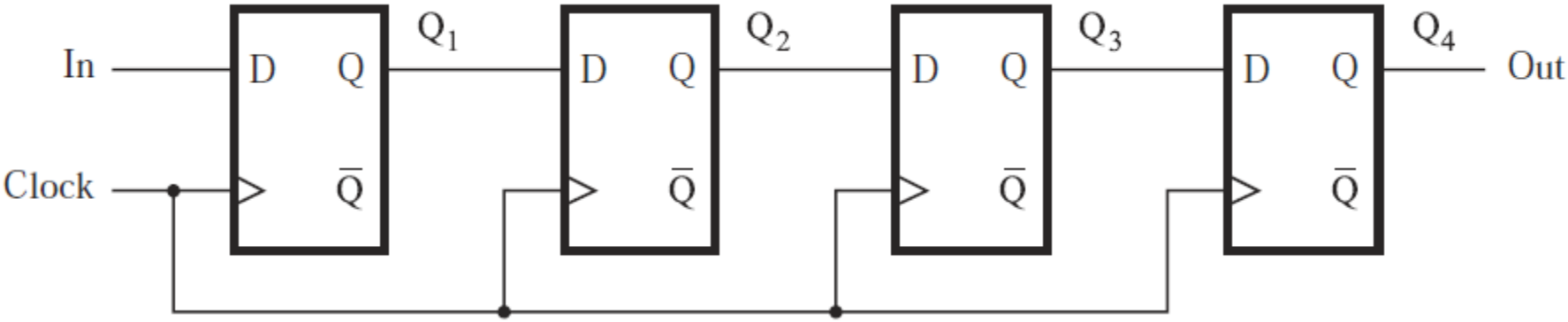
Registers

- An N-bit register is a set of N flip-flops used to store N bits of information.
- Example: 4-bit register
  - One D flip-flop for each bit
  - Same Clock and Clear signals for all 4 D flip-flops
  - Input: D_0-D_3
  - Output: Q_0-Q_3



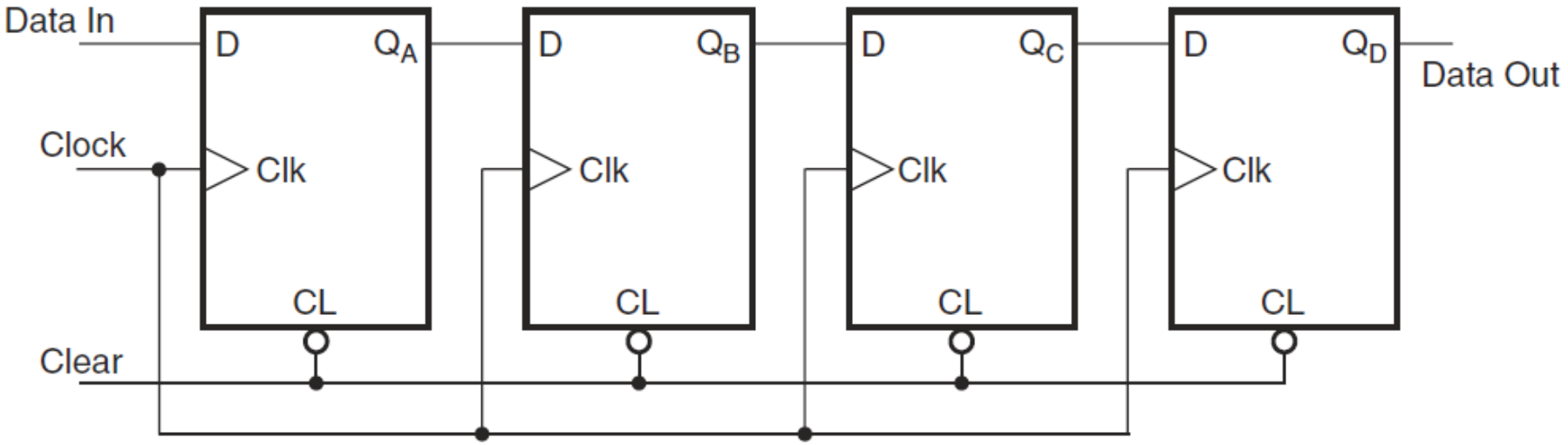
Shift registers

- Shift register: a register with the ability to shift its contents
- Example: 4-bit shift register

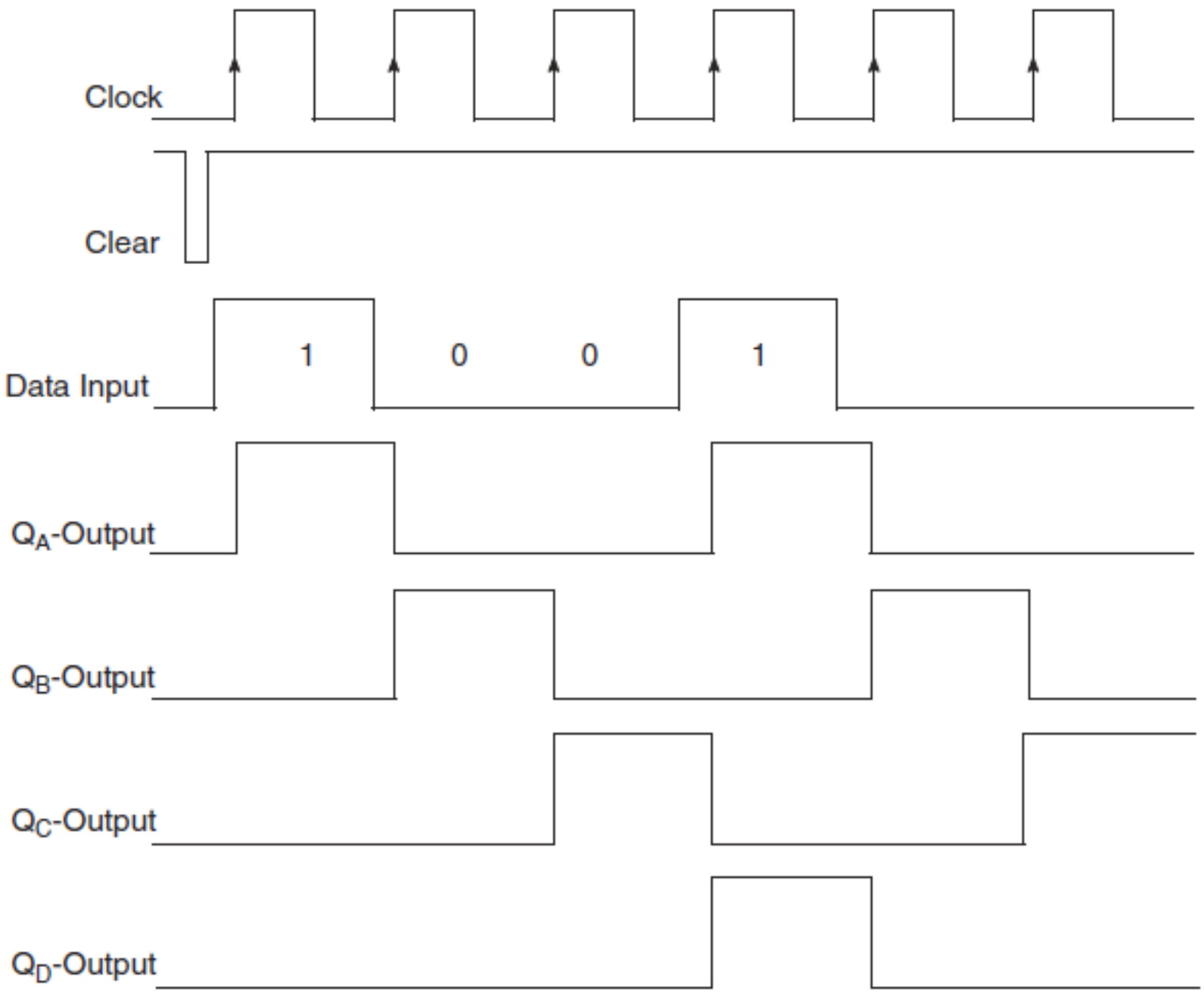


4-bit shift register

- Circuit diagram (with asynchronous clear)



- Timing diagram (next page)
  - Input: 1 (first bit), 0, 0, 1 (last bit)


Clock
Clear
Data Input
1
0
0
1
Q_A-Output
Q_B-Output
Q_C-Output
Q_D-Output

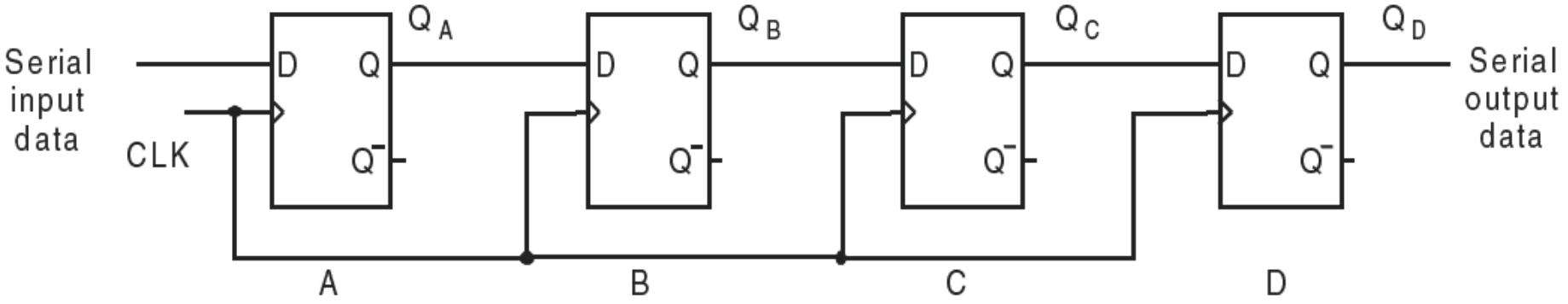
4-bit shift register

- Contents of flip-flops
  - Input is shifted to the right by one bit during each clock cycle
  - An input bit propagates to output after 4 clock cycles
  - 4 clock cycles to load input, another 4 cycles to read output

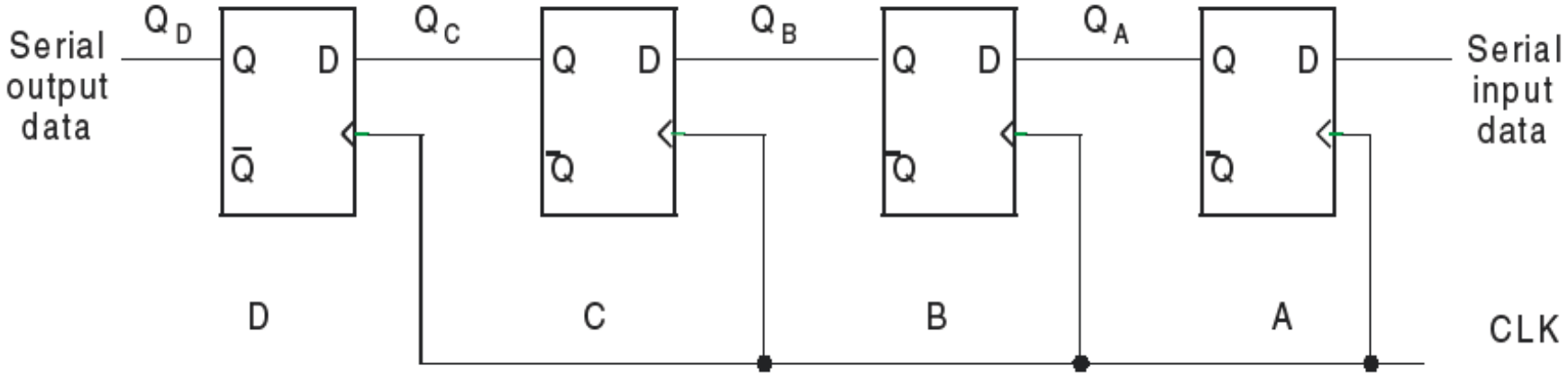
Clock	Q_A	Q_B	Q_C	Q_D
Initial contents	0	0	0	0
After first clock transition	1	0	0	0
After second clock transition	0	1	0	0
After third clock transition	0	0	1	0
After fourth clock transition	**1**	**0**	**0**	**1**
After fifth clock transition	0	1	0	0
After sixth clock transition	0	0	1	0
After seventh clock transition	0	0	0	1
After eighth clock transition	**0**	**0**	**0**	**0**

4-bit registers: shift-right and shift-left

- Shift-right



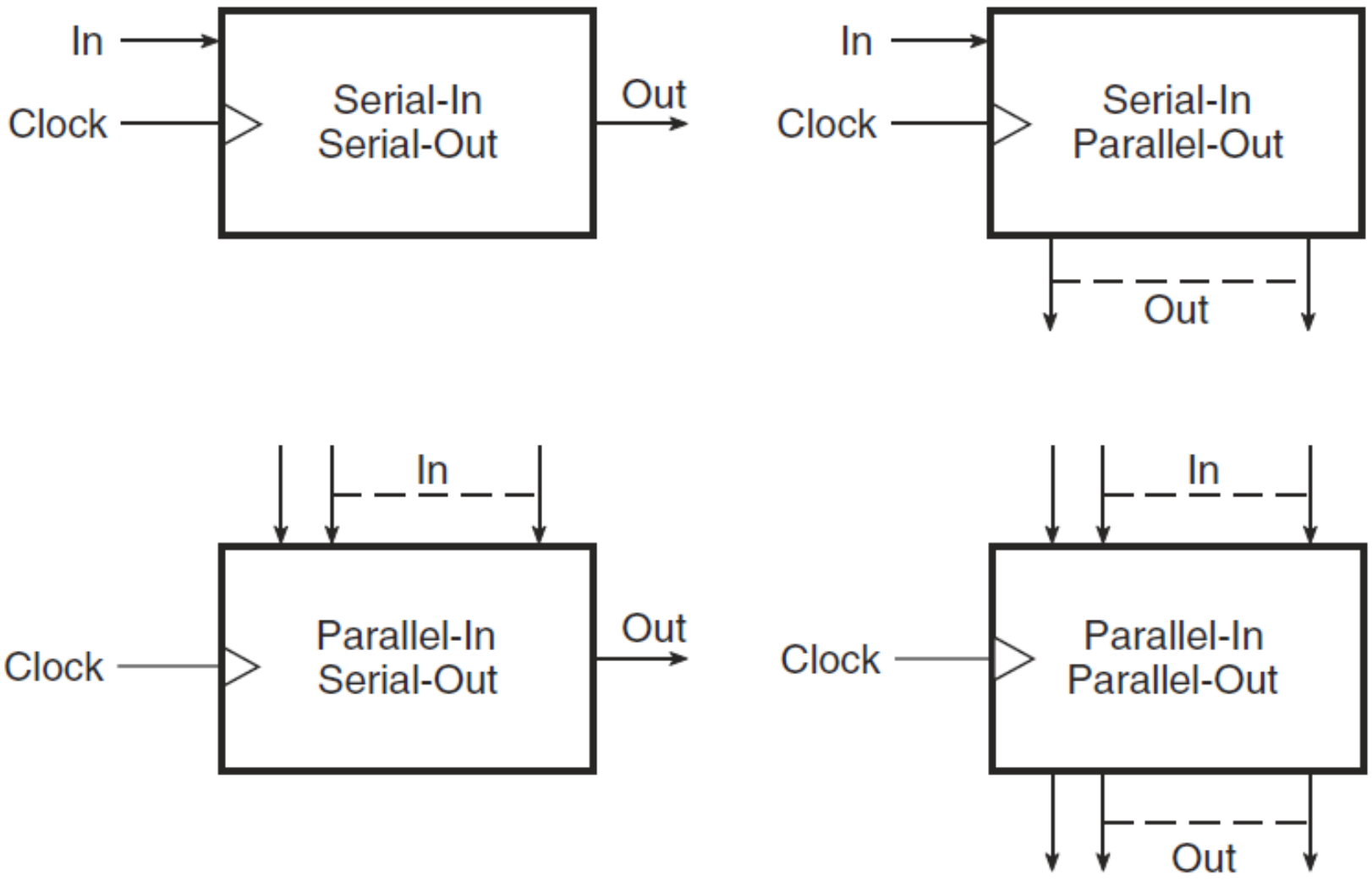
- Shift-left



Shift register types

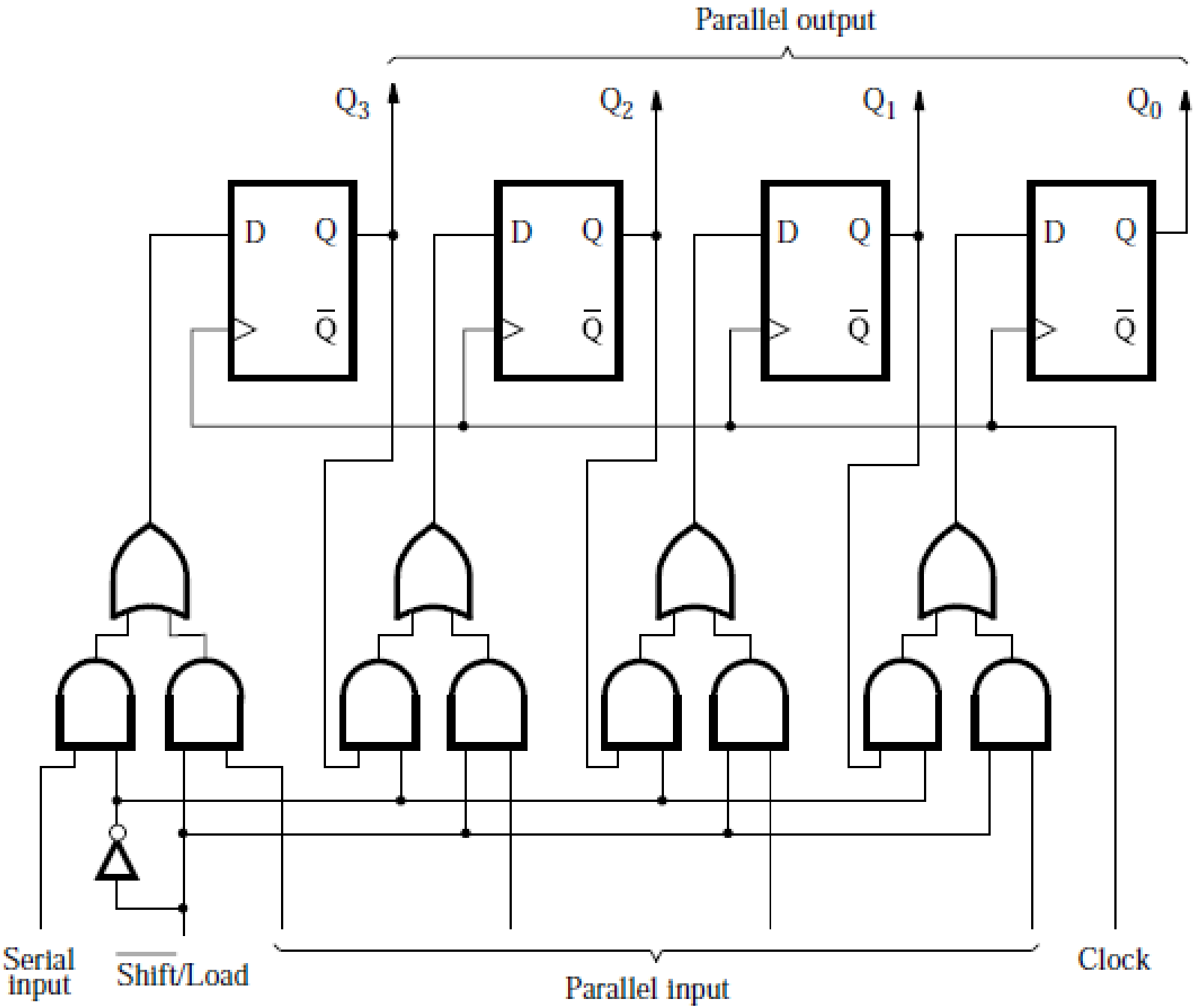
- 4 types depending on the method to load and read data
  - Serial-in serial-out (SISO)
  - Serial-in parallel-out (SIPO)
  - Parallel-in serial-out (PISO)
  - Parallel-in parallel-out (PIPO)

Shift register types



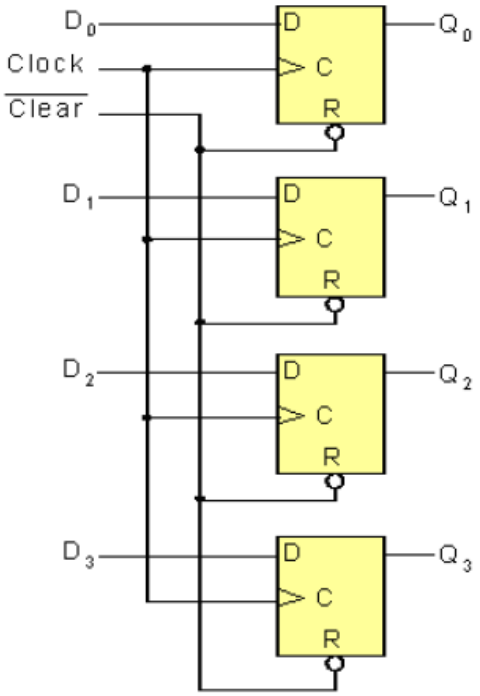
Parallel-access (parallel-loading) shift register

- Circuit structure
  - Top (D flip-flops): register
  - Bottom (2 AND + 1 OR gates): 2-to-1 mux
- Mode selection signal: $\overline{Shift}/Load$
  - Value 0: shift register
    - Serial input is selected: serial input to D_3 (D input of leftmost flip-flop), Q_3 to D_2, Q_2 to D_1, Q_1 to D_0
    - Shift-right register
  - Value 1: parallel loading
    - Parallel input is selected


Parallel output
Q_3
Q_2
Q_1
Q_0
D
Q
$\overline{Q}$
Serial input
$\overline{\text{Shift/Load}}$
Parallel input
Clock

Code for 8-bit register with asynchronous clear

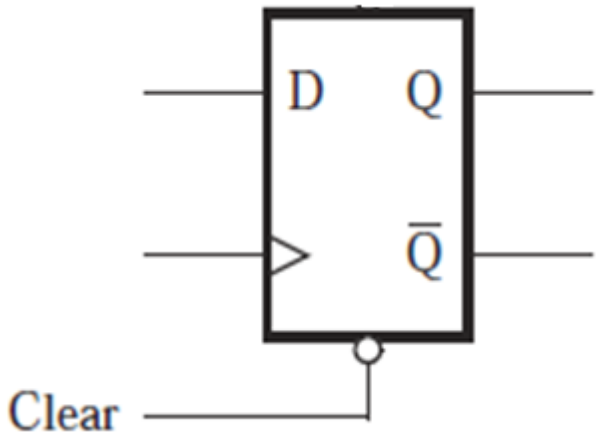
```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;

ENTITY reg8 IS
    PORT ( D              : IN   STD_LOGIC_VECTOR(7 DOWNTO 0) ;
           Resetn, Clock  : IN   STD_LOGIC ;
           Q              : OUT  STD_LOGIC_VECTOR(7 DOWNTO 0) ) ;
END reg8 ;

ARCHITECTURE Behavior OF reg8 IS
BEGIN
    PROCESS ( Resetn, Clock )
    BEGIN
        IF Resetn = '0' THEN
            Q <= "00000000" ;
        ELSIF Clock'EVENT AND Clock = '1' THEN
            Q <= D ;
        END IF ;
    END PROCESS ;
END Behavior ;
```



Comparison: Code for D flip-flop with asynchronous reset (clear)

- Asynchronous reset (clear): active low

```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;

ENTITY flipflop IS
      PORT ( D, Resetn, Clock   : IN     STD_LOGIC ;
                Q               : OUT    STD_LOGIC) ;
END flipflop ;

ARCHITECTURE Behavior OF flipflop IS
BEGIN
      PROCESS ( Resetn, Clock )
      BEGIN
            IF Resetn = '0' THEN
                  Q <= '0' ;
            ELSIF Clock'EVENT AND Clock = '1' THEN
                  Q <= D ;
            END IF ;
      END PROCESS ;
END Behavior ;
```



Code for N-bit register with asynchronous clear

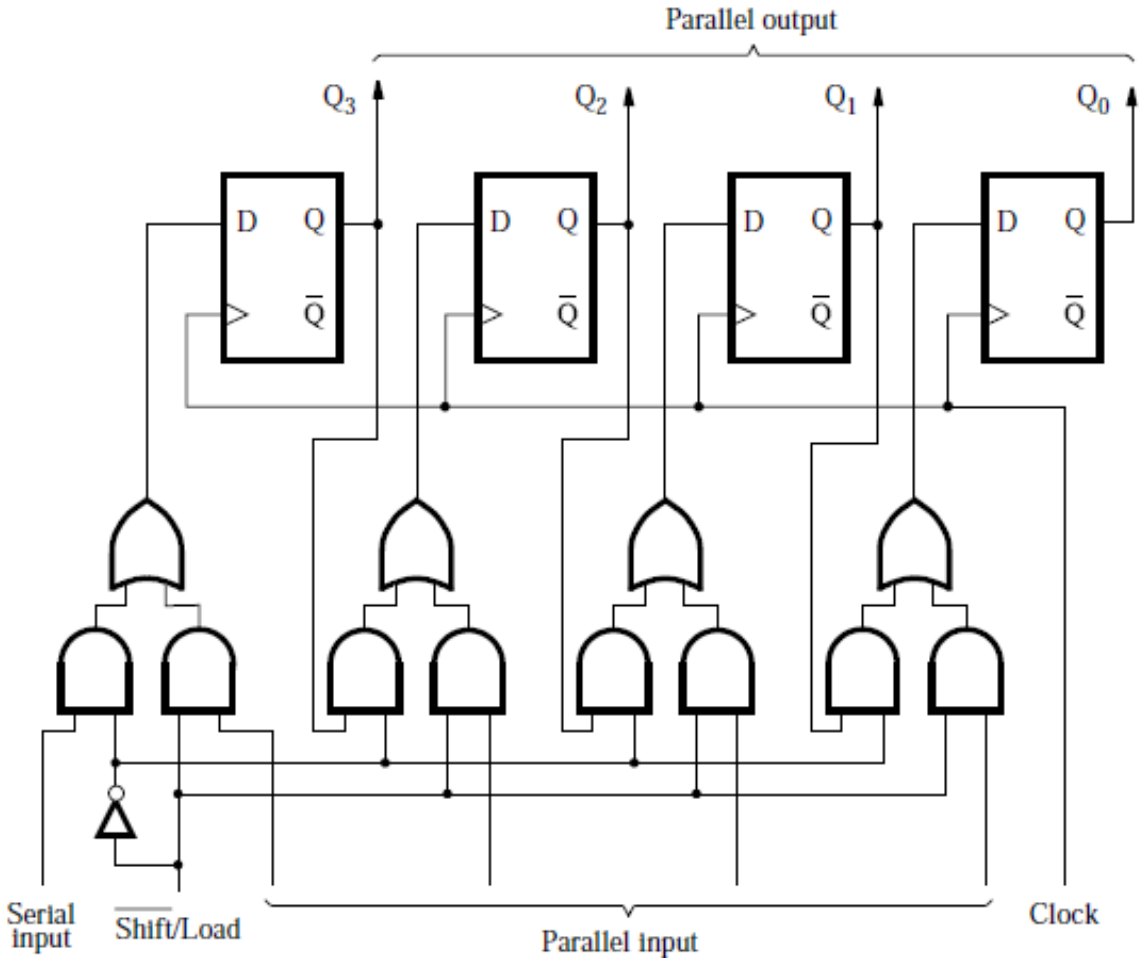
```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;

ENTITY regn IS
    GENERIC ( N : INTEGER := 16 ) ;
    PORT ( D              : IN   STD_LOGIC_VECTOR(N–1 DOWNTO 0) ;
           Resetn, Clock  : IN   STD_LOGIC ;
           Q              : OUT  STD_LOGIC_VECTOR(N–1 DOWNTO 0) ) ;
END regn ;

ARCHITECTURE Behavior OF regn IS
BEGIN
    PROCESS ( Resetn, Clock )
    BEGIN
        IF Resetn = '0' THEN
            Q <= (OTHERS => '0') ;
        ELSIF Clock'EVENT AND Clock = '1' THEN
            Q <= D ;
        END IF ;
    END PROCESS ;
END Behavior ;
```

Code for N-bit register with asynchronous clear

- GENERIC (N: INTEGER := 16)
  - Define a parameter (N), its type (INTEGER), and its value (16)
  - Assignment operator: ":="
- Q <= (OTHERS => '0')
  - Reset all bits of Q to 0 regardless of the bit length of Q
  - For N=16, this statement is equivalent to Q<="0000000000000000"

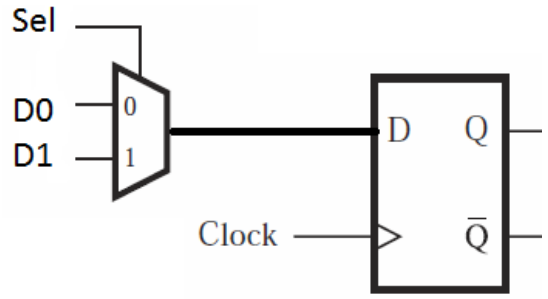
Structural code for 4-bit parallel-access shift register

- Component: a D flip-flop with a 2-to-1 MUX on D input



Structural code for 4-bit parallel-access shift register: component

- Sel: mode selection
  - Sel=0: D=D0 (left-hand AND is active, left signal is selected: serial input or preceding Q)
  - Sel=1, D=D1 (right-hand AND is active, right signal is selected: parallel input)




```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;

ENTITY muxdff IS
    PORT ( D0, D1, Sel, Clock  : IN   STD_LOGIC ;
           Q                   : OUT  STD_LOGIC ) ;
END muxdff ;

ARCHITECTURE Behavior OF muxdff IS
BEGIN
    PROCESS
    BEGIN
        WAIT UNTIL Clock'EVENT AND Clock = '1' ;
        IF Sel = '0' THEN
            Q <= D0 ;
        ELSE
            Q <= D1 ;
        END IF ;
    END PROCESS ;
END Behavior ;
```

Structural code for 4-bit parallel-access shift register: entire circuit

- R: parallel input, L: mode selection, w: serial input

```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;

ENTITY shift4 IS
    PORT ( R            : IN     STD_LOGIC_VECTOR(3 DOWNTO 0) ;
           L, w, Clock  : IN     STD_LOGIC ;
           Q            : BUFFER STD_LOGIC_VECTOR(3 DOWNTO 0) ) ;
END shift4 ;

ARCHITECTURE Structure OF shift4 IS
    COMPONENT muxdff
        PORT ( D0, D1, Sel, Clock  : IN  STD_LOGIC ;
               Q                   : OUT STD_LOGIC ) ;
    END COMPONENT ;
BEGIN
    Stage3: muxdff PORT MAP ( w, R(3), L, Clock, Q(3) ) ;
    Stage2: muxdff PORT MAP ( Q(3), R(2), L, Clock, Q(2) ) ;
    Stage1: muxdff PORT MAP ( Q(2), R(1), L, Clock, Q(1) ) ;
    Stage0: muxdff PORT MAP ( Q(1), R(0), L, Clock, Q(0) ) ;
END Structure ;
```

Behavioral code for 4-bit parallel-access shift register

```
1   LIBRARY ieee ;
2   USE ieee.std_logic_1164.all ;
3   ENTITY shift4 IS
4       PORT ( R       : IN     STD_LOGIC_VECTOR(3 DOWNTO 0) ;
5              Clock   : IN     STD_LOGIC ;
6              L, w    : IN     STD_LOGIC ;
7              Q       : BUFFER STD_LOGIC_VECTOR(3 DOWNTO 0) ) ;
8   END shift4 ;
9   ARCHITECTURE Behavior OF shift4 IS
10  BEGIN
11      PROCESS
12      BEGIN
13          WAIT UNTIL Clock'EVENT AND Clock = '1' ;
14          IF L = '1' THEN
15              Q <= R ;
16          ELSE
17              Q(0) <= Q(1) ;
18              Q(1) <= Q(2);
19              Q(2) <= Q(3) ;
20              Q(3) <= w ;
21          END IF ;
22      END PROCESS ;
23  END Behavior ;
```

Behavioral code for N-bit parallel-access shift register

```
1   LIBRARY ieee ;
2   USE ieee.std_logic_1164.all ;
3   ENTITY shiftn IS
4       GENERIC ( N : INTEGER := 8 ) ;
5       PORT ( R       : IN     STD_LOGIC_VECTOR(N–1 DOWNTO 0) ;
6              Clock   : IN     STD_LOGIC ;
7              L, w    : IN     STD_LOGIC ;
8              Q       : BUFFER STD_LOGIC_VECTOR(N–1 DOWNTO 0) ) ;
9   END shiftn ;
10  ARCHITECTURE Behavior OF shiftn IS
11  BEGIN
12      PROCESS
13      BEGIN
14          WAIT UNTIL Clock'EVENT AND Clock = '1' ;
15          IF L = '1' THEN
16              Q <= R ;
17          ELSE
18              Genbits: FOR i IN 0 TO N-2 LOOP
19                  Q(i) <= Q(i + 1) ;
20              END LOOP ;
21              Q(N-1) <= w ;
22          END IF ;
23      END PROCESS ;
24  END Behavior ;
```

FOR LOOP: generate a set of sequential statements

Counters

- Asynchronous (serial or ripple): different stages are driven by different clocks
  - Up-counters
  - Down-counters
- Synchronous (parallel): different stages are driven by the same clock
  - Up-counters
  - Down-counters

Asynchronous up-counter with T flip-flops

- A 3-bit up-counter
  - It counts the number of clock cycles
  - Stage 1 is driven by Clock, stages 2 and 3 are driven by preceding output complement
  - Counter output: $Q_2Q_1Q_0$ (MSB to LSB)
  - Counter modulus (MOD number): number of different logic states represented by counter
  - Modulo-8 or MOD-8: states 0-7
  - Asynchronous (serial or ripple): three stages respond after different delays

Asynchronous up-counter with T flip-flops