# REGISTRO DE DESPLAZAMIENTO

**El SHIFT, es una conexion en cascada de una serie de FF. Por cada pulso de reloj la información se desplaza (shift) de un FF a otro FF**

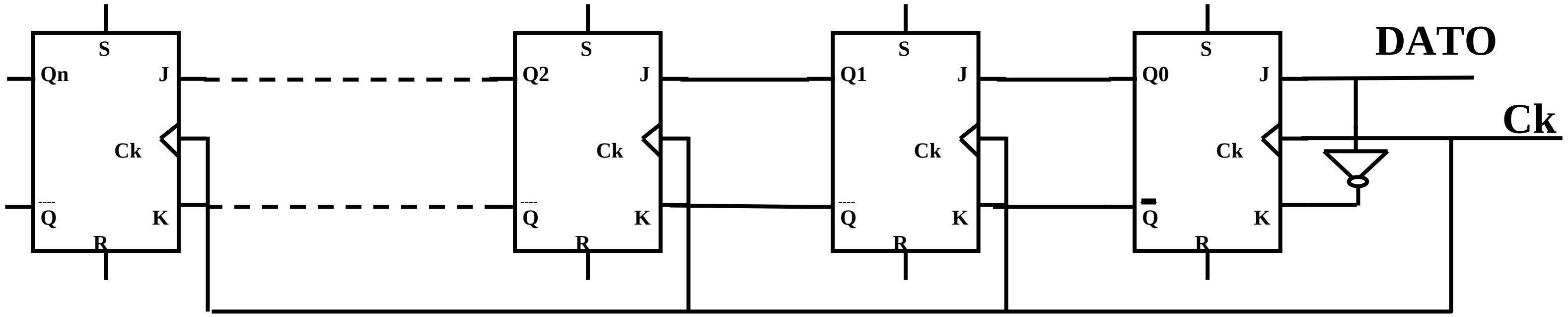
**CLASIFICACIÓN:**

| ENTRADA | SALIDA |
|---|---|
| Serie | Serie |
| Serie | Paralelo |
| Paralelo | Serie |
| Paralelo | Paralelo |

**SHIFT RIGH**

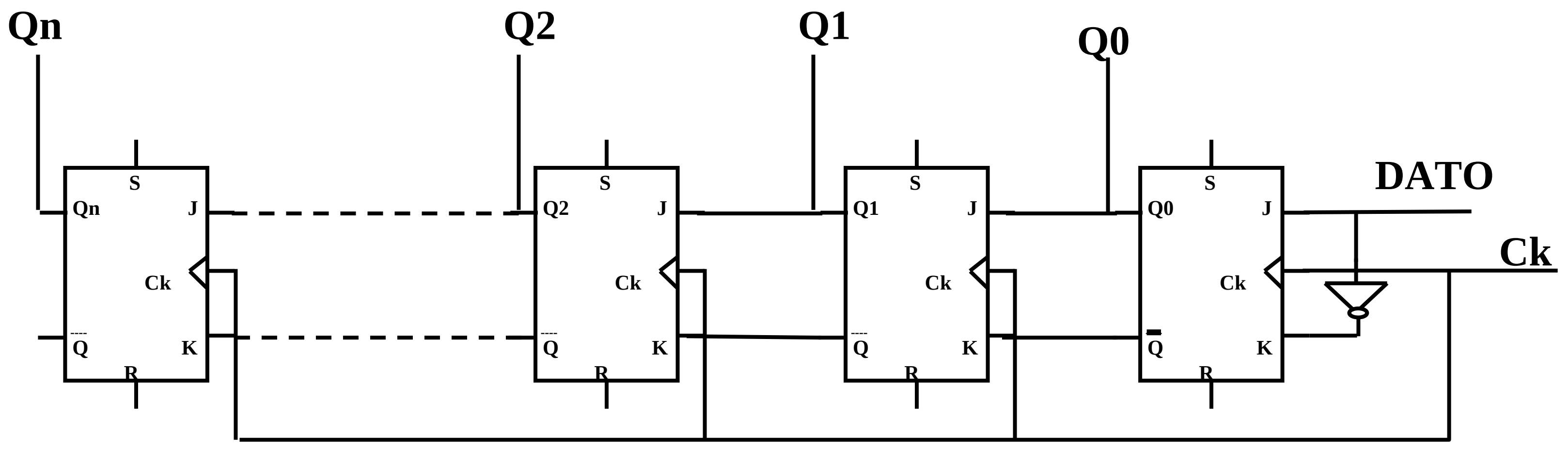
**SHIFT LEFT**

**SHIFT SERIE/SERIE**



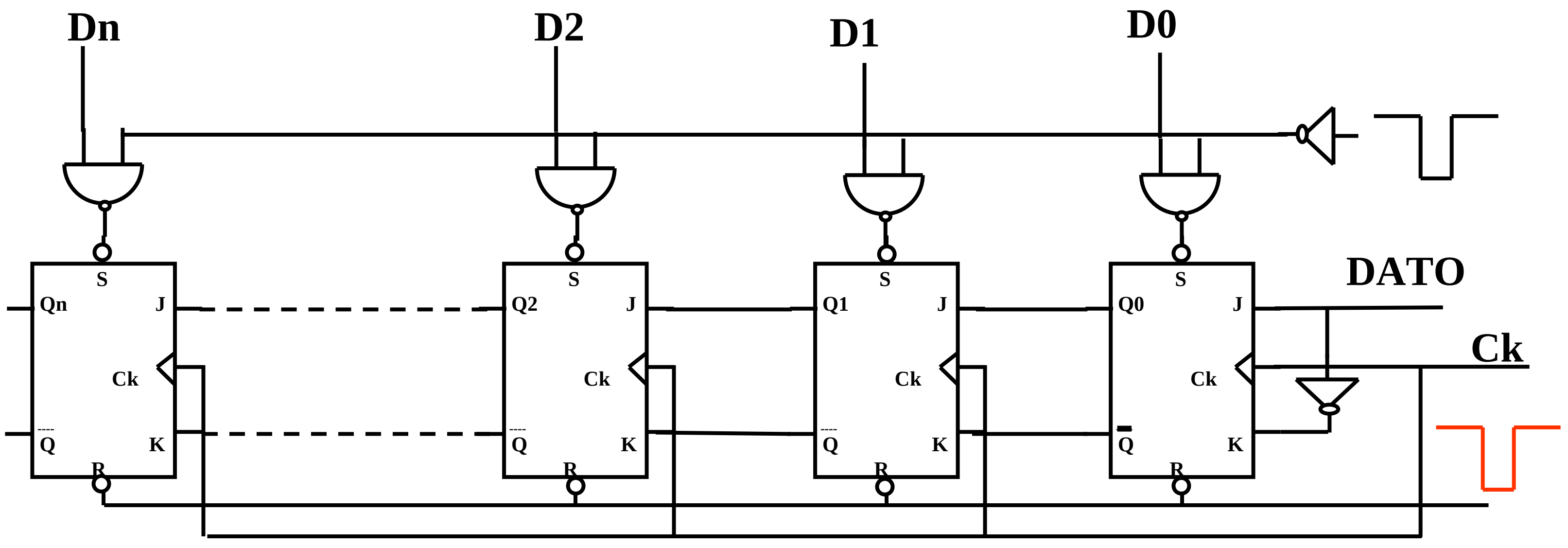
**CONECTAR LAS ENTRADAS ASINCRONAS**

# SHIFT SERIE/PARALELO



CONECTAR LAS ENTRADAS ASINCRONAS

## SHIFT  ENTRADA PARALELO



```
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') ; --equivale a  Q<="0000000000000000"
		ELSIF Clock'EVENT AND Clock = '1' THEN
			Q <= D ;
		END IF ;
	END PROCESS ;
END Behavior ;
```

Figure 7.47 Code for an *n*-bit register with asynchronous clear

```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;
ENTITY shift4 IS
    PORT (  R       : IN        STD_LOGIC_VECTOR(3 DOWNTO 0) ;
            Clock   : IN        STD_LOGIC ;
            L, w    : IN        STD_LOGIC ;
            Q       : BUFFER    STD_LOGIC_VECTOR(3 DOWNTO 0) ) ;
END shift4 ;

ARCHITECTURE Behavior OF shift4 IS
BEGIN
    PROCESS
    BEGIN
        WAIT UNTIL Clock'EVENT AND Clock = '1' ;
        IF L = '1' THEN
            Q <= R ;
        ELSE                        --Las 4 lineas que siguen estan programadas
            Q(0) <= Q(1) ;          --para ocurrir solo despues de que todas las
            Q(1) <= Q(2);           --instrucciones del proceso han sido evaluadas,
            Q(2) <= Q(3) ;          --por consiguiente los 4 FF cambian simultaneamente
            Q(3) <= w ;
        END IF ;
    END PROCESS ;
END Behavior ;
```

Figure 7.50 Alternative code for a shift register

```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;
ENTITY shift4 IS
	PORT (	R		: IN		STD_LOGIC_VECTOR(3 DOWNTO 0) ;
		Clock		: IN		STD_LOGIC ;
		L, w		: IN		STD_LOGIC ;
		Q		: BUFFER	STD_LOGIC_VECTOR(3 DOWNTO 0) ) ;
END shift4 ;

ARCHITECTURE Behavior OF shift4 IS
BEGIN
	PROCESS
	BEGIN
		WAIT UNTIL Clock'EVENT AND Clock = '1' ;
		IF L = '1' THEN
			Q <= R ;
		ELSE				--Identico al anterior
			Q(3) <= w ;
			Q(2) <= Q(3) ;
			Q(1) <= Q(2);
			Q(0) <= Q(1) ;
		END IF ;
	END PROCESS ;
END Behavior ;
```

Figure 7.51 Code that reverses the ordering of statements

```
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 ;
```

Figure 7.46 Code for an eight-bit register with asynchronous clear

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

ENTITY trin IS
	GENERIC ( N : INTEGER := 8 ) ;
	PORT (	X	: IN	STD_LOGIC_VECTOR(N-1 DOWNTO 0) ;
		E	: IN	STD_LOGIC ;
		F	: OUT	STD_LOGIC_VECTOR(N-1 DOWNTO 0) ) ;
END trin ;

ARCHITECTURE Behavior OF trin IS
BEGIN
	F <= (OTHERS => 'Z') WHEN E = '0' ELSE X ;
END Behavior ;
```

Figure 7.63 Code for an *n*-bit tri-state buffer

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

ENTITY shiftr IS  -- left-to-right shift register with async reset
	GENERIC ( K : INTEGER := 4 ) ;
	PORT (	Resetn, Clock, w	: IN		STD_LOGIC ;
		Q			: BUFFER	STD_LOGIC_VECTOR(1 TO K) ) ;
END shiftr ;

ARCHITECTURE Behavior OF shiftr IS
BEGIN
	PROCESS ( Resetn, Clock )
	BEGIN
		IF Resetn = '0' THEN
			Q <= (OTHERS => '0') ;
		ELSIF Clock'EVENT AND Clock = '1' THEN
			Genbits: FOR i IN K DOWNTO 2 LOOP
				Q(i) <= Q(i-1) ;
			END LOOP ;
			Q(1) <= w ;
		END IF ;
	END PROCESS ;
END Behavior ;
```

Figure 7.64 Code for the shift-register controller

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

ENTITY regn IS
    GENERIC ( N : INTEGER := 8 ) ;
    PORT (  R           : IN    STD_LOGIC_VECTOR(N-1 DOWNTO 0) ;
            Rin, Clock  : IN    STD_LOGIC ;
            Q           : OUT   STD_LOGIC_VECTOR(N-1 DOWNTO 0) ) ;
END regn ;

ARCHITECTURE Behavior OF regn IS
BEGIN
    PROCESS
    BEGIN
        WAIT UNTIL Clock'EVENT AND Clock = '1' ;
        IF Rin = '1' THEN
            Q <= R ;
        END IF ;
    END PROCESS ;
END Behavior ;
```

Figure 7.62 Code for an *n*-bit register with enable