CONTADORES Y REGISTROS

Técnicas Digitales I

Luis Eduardo Toledo

UTN

REGISTROS

Un ***registro*** es un grupo de flip-flops, cada uno de los cuales comparte un clock común y es capaz de almacenar un bit de información .

Un registro de ***n-bits*** consiste de un grupo de ***n*** flip-flops capaz de almacenar ***n*** bits de información binaria.

En su definición mas amplia, un registro consiste de un grupo de flip-flops junto a compuertas que afectan su operación. Los flip-flops mantienen la información binaria y las compuertas definen como la información se transfiere en el registro.

UTN

DESCRIPCIÓN DE UN REGISTRO EN VERILOG

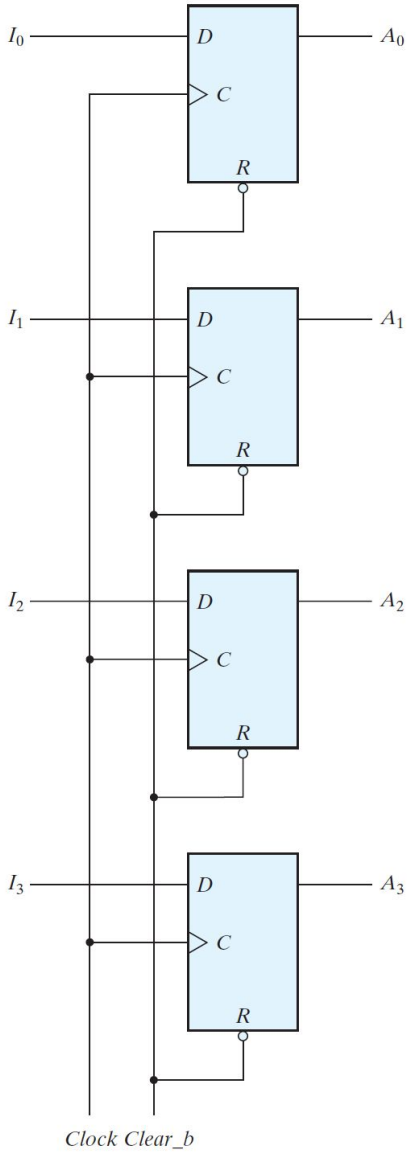
El registro mas simple es uno que consiste solamente de flip-flops sin ninguna compuerta.

La descripción muestra un registro construido con 4 flip-flops D para formar un registro de almacenamiento de datos de 4 bits.

```
module flop (input        clk,
             input  [3:0] d,
             output reg [3:0] q);
 always @ (posedge clk)
   q <= d;
endmodule
```

Circuito sintetizado

ESQUEMÁTICO DE UN REGISTRO DE 4 BITS CON RESET ASINCRÓNICO




```
module flopr (input Clock,
              input Clear_b,
              input [3:0] I,
              output reg [3:0] A);
// asynchronous reset
always @ (posedge Clock, negedge Clear_b)
  if (~Clear_b) A <=  4'b0;
  else A <= I;
endmodule
```

ESQUEMÁTICO DE UN REGISTRO DE 4 BITS CON CARGA EN PARALELO

```
module flopenr (input Clock,
                Input Load,
                input [3:0] I,
                output reg [3:0] A);
  always @ (posedge Clock)
    if (Load) A <= I;
endmodule
```

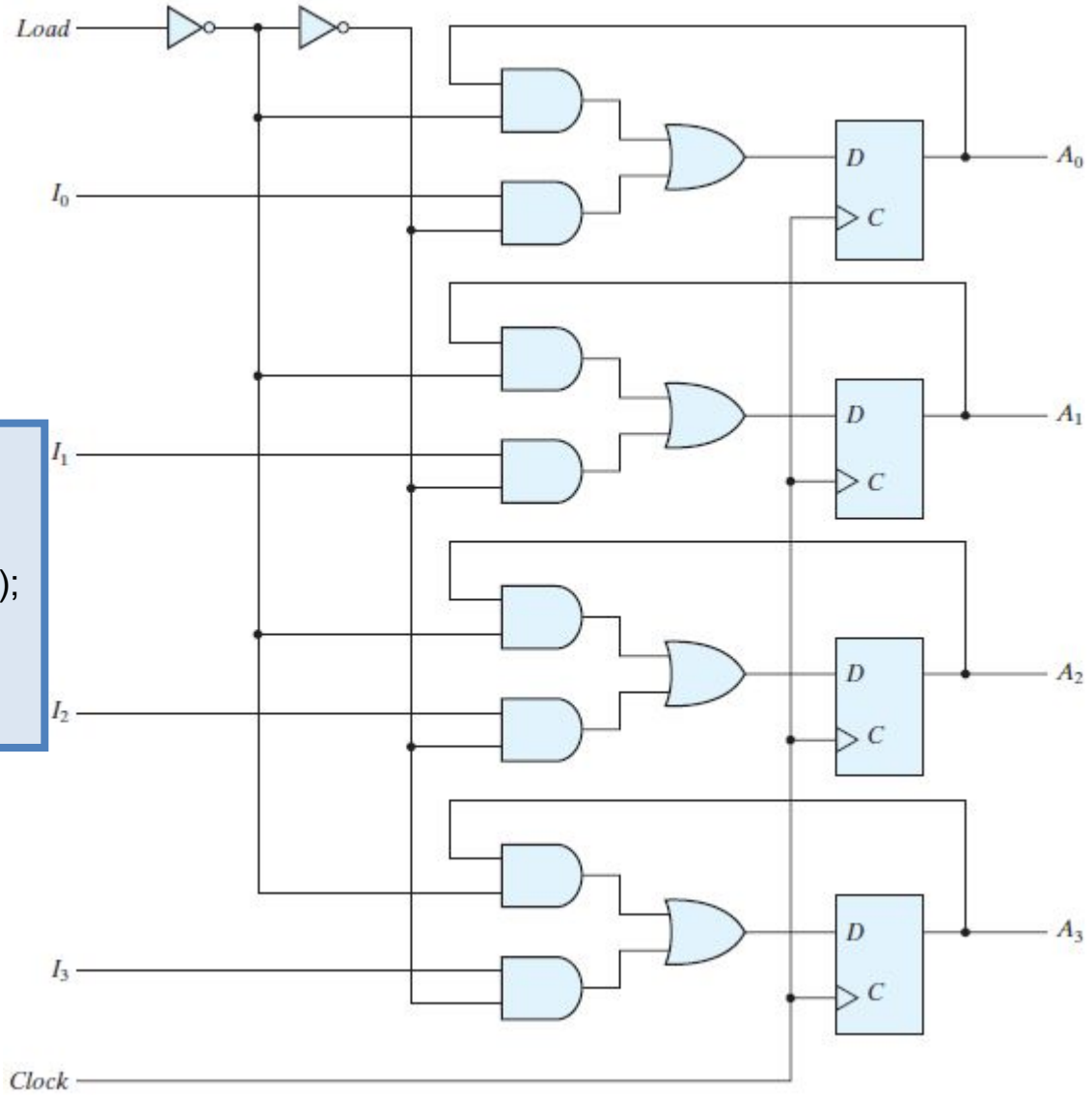
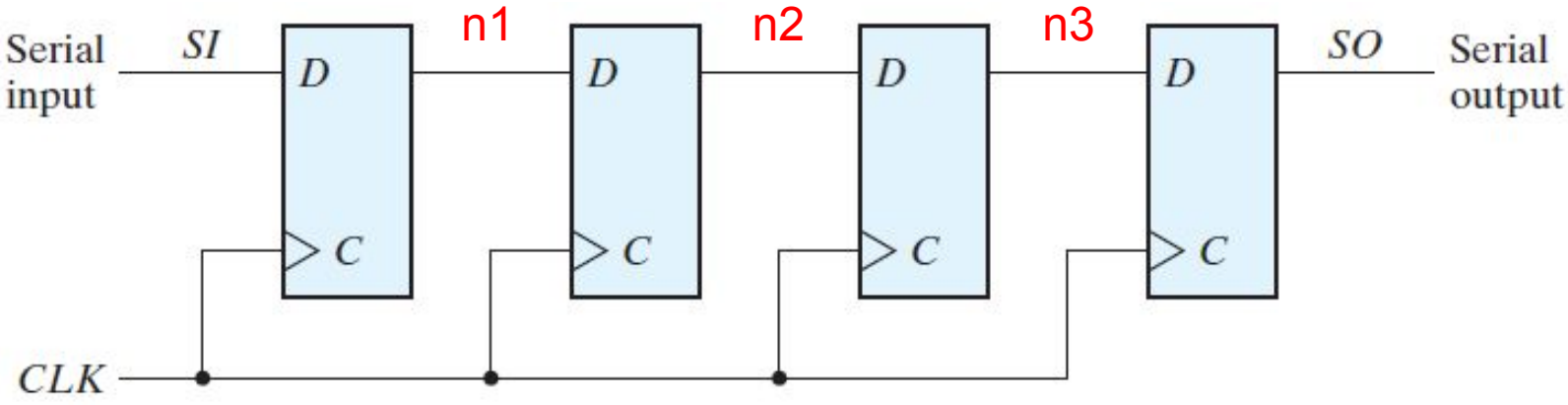
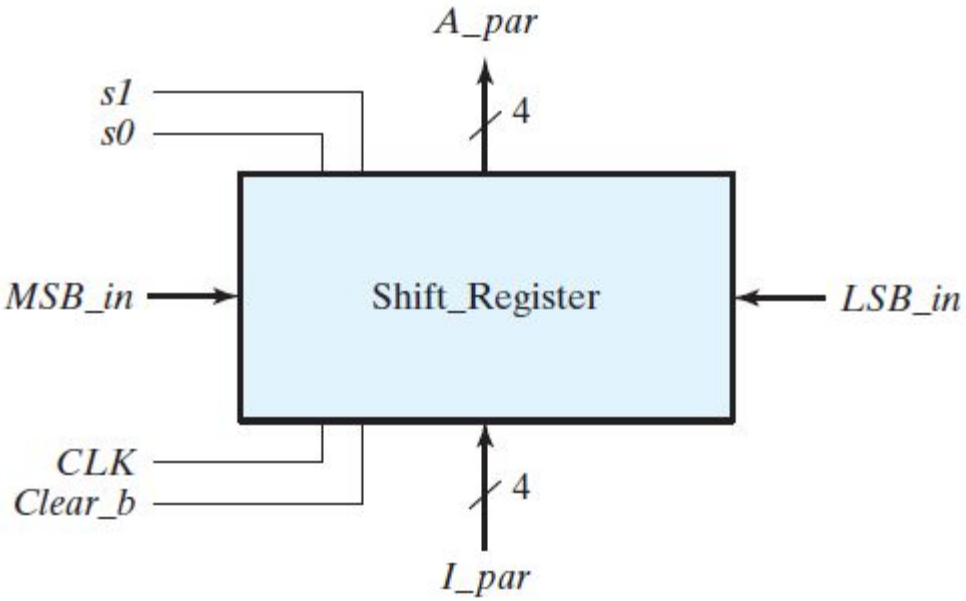


DIAGRAMA DE UN REGISTRO DE DESPLAZAMIENTO DE 4 BITS



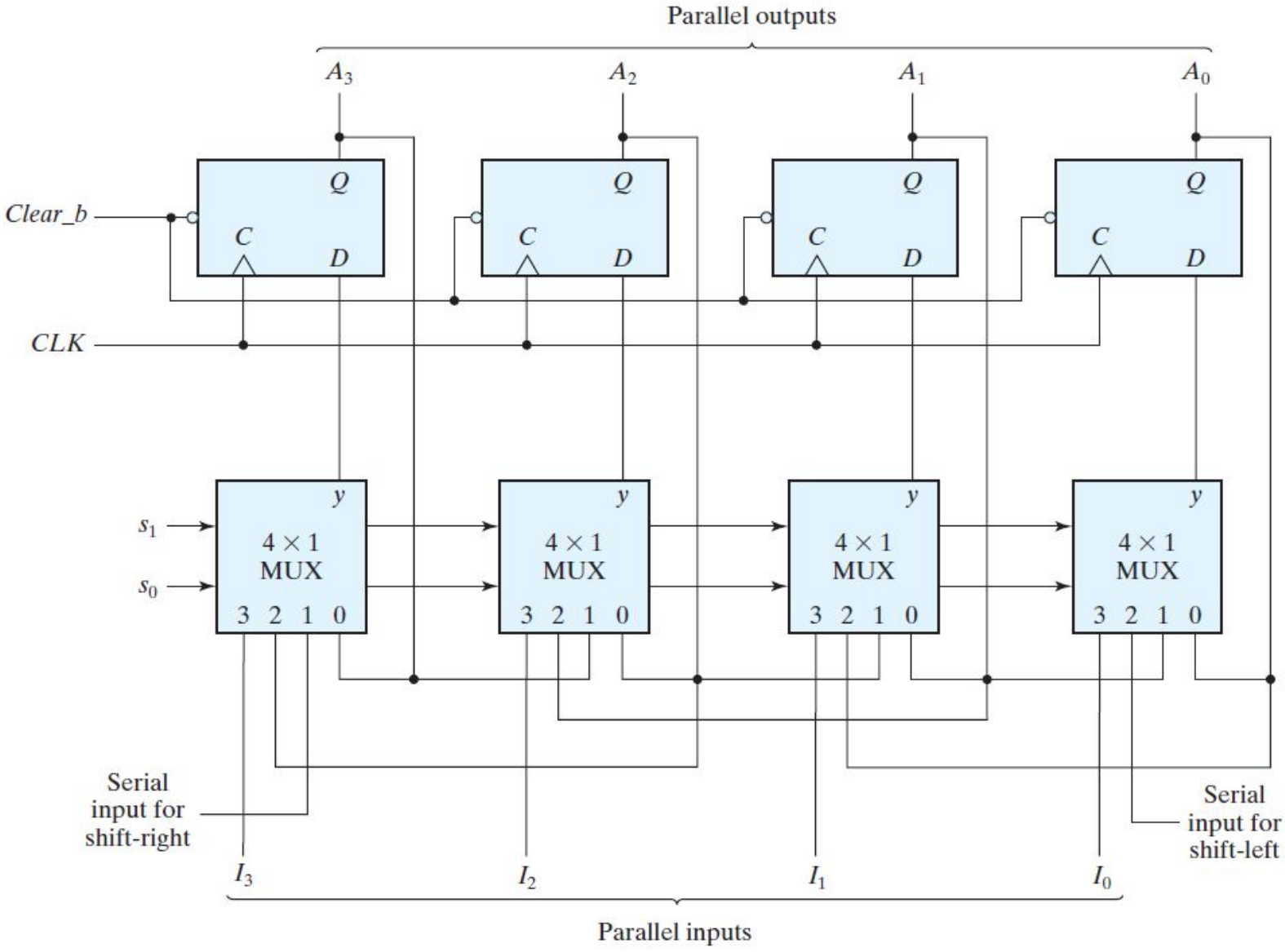

```
module shift (input CLK,
              input SI,
              output reg SO);
  reg n1, n2, n3;
  always @ (posedge CLK)
    begin
      n1 <= SI;
      n2 <= n1;
      n3 <= n2;
      SO <= n3;
    end
endmodule
```

DIAGRAMA DE UN REGISTRO DE DESPLAZAMIENTO UNIVERSAL DE 4 BITS



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

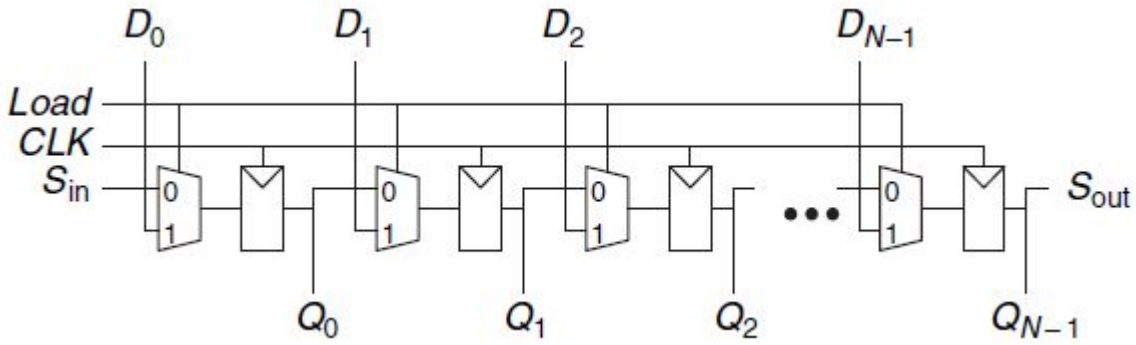
DIAGRAMA DE UN REGISTRO DE DESPLAZAMIENTO UNIVERSAL DE 4 BITS



DESCRIPCIÓN VERILOG DE UN REGISTRO DE DESPLAZAMIENTO UNIVERSAL DE 4 BITS

```
// Behavioral description of a 4-bit universal shift register
// Fig. 6.7 and Table 6.3
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
```

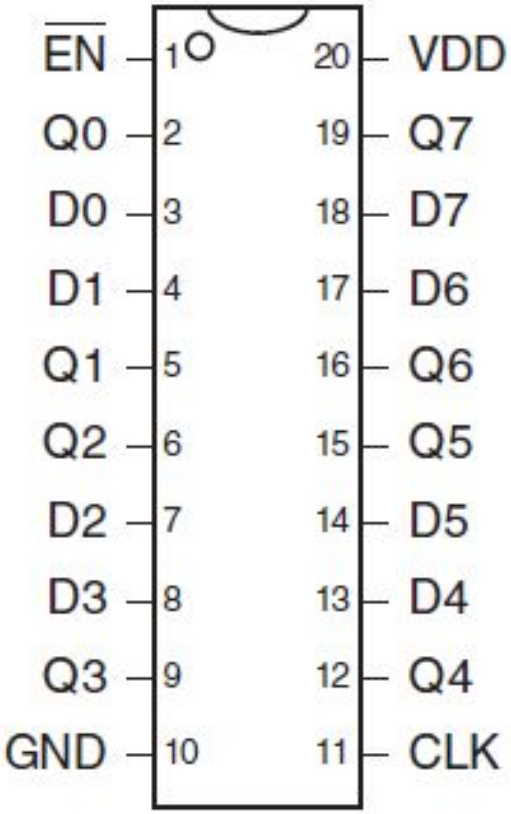
DIAGRAMA DE UN REGISTRO DE DESPLAZAMIENTO DE N BITS




```
module shiftreg # (parameter N = 8)
                  (input          clk,
                   input          reset, load,
                   input          sin,
                   input          [N-1:0] d,
                   output reg     [N-1:0] q,
                   output         sout);
  always @ (posedge clk, posedge reset)
    if (reset)       q <= 0;
    else if (load)   q <= d;
    else             q <= {q[N-2:0], sin};

  assign sout = q[N-1];
endmodule
```

REGISTRO DE 8 BITS COMERCIAL CON HABILITACIÓN Y SU EQUIVALENCIA HDL



74377 Register

8-bit Enableable Register

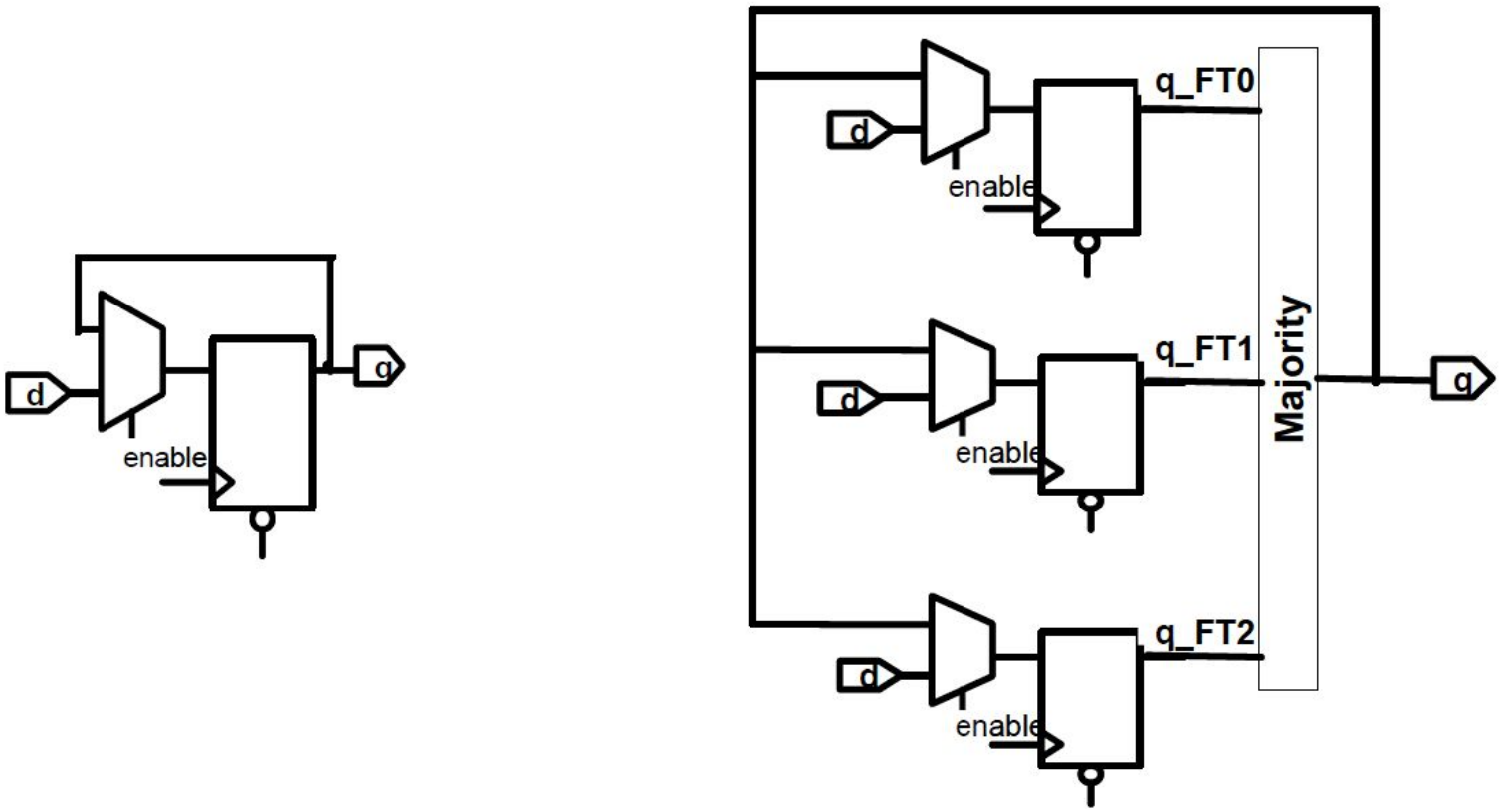
CLK:	clock
$D_{7:0}$:	data
$Q_{7:0}$:	output
ENb:	enable

```
always_ff @(posedge CLK)
  if (~ENb) Q <= D;
```

always @(posedge CLK)
if (~ENb) Q <= D;

EJEMPLO DE REGISTRO TOLERANTE A FALLAS

TMR (*Triple Modular Redundancy).* Redundancia hardware pasiva



a) b)

Mecanismo de votación por mayoría, TMR. (a) Módulo original. (b) Módulo triplicado con votación

CONTADORES

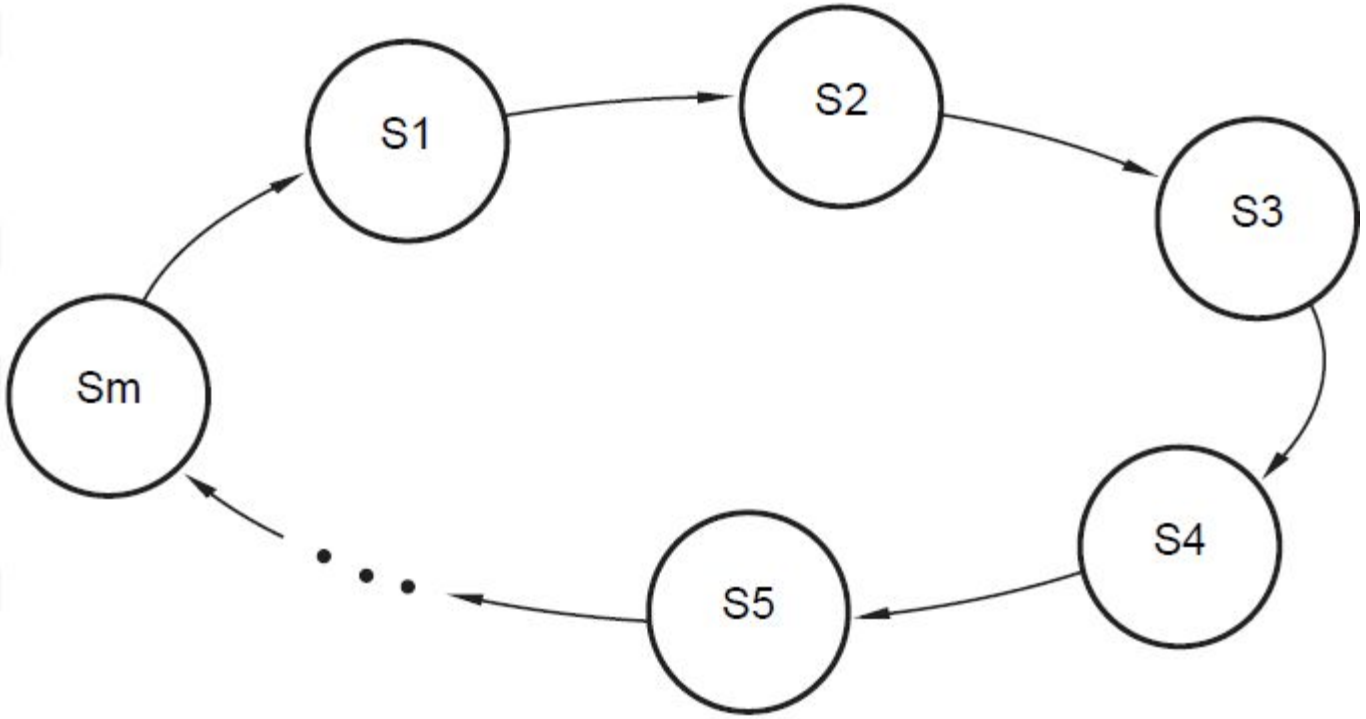
ALGUNOS CONCEPTOS.

Un contador es esencialmente un registro que va a través de una secuencia predeterminada de estados binarios. Las compuertas en el contador están conectadas de tal forma que producen la secuencia prescripta de estados.

A pesar de que los contadores son un tipo especial de registros, es común diferenciarlos dándoles un nombre diferente.

CONTADORES

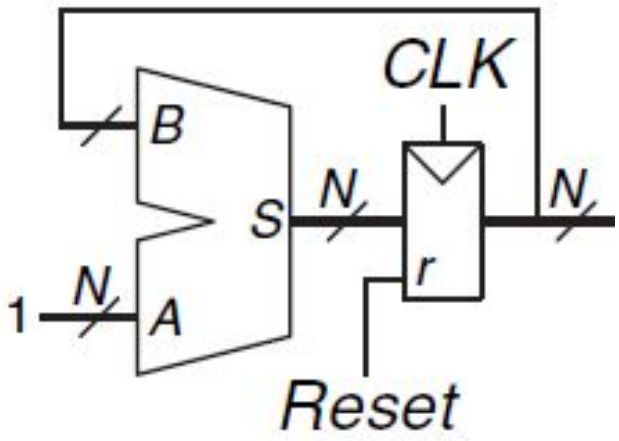
El nombre de contador se usa para cualquier circuito secuencial sincronizado cuyo diagrama de estados contiene un solo ciclo.



Un contador con ***m*** estados se llama ***contador módulo m*** o, algunas veces, un ***contador divisor por m***.

CONTADOR BINARIO DE *n* BITS

El ***contador binario de n bits*** es probablemente el tipo de contador que se usa con mayor frecuencia.

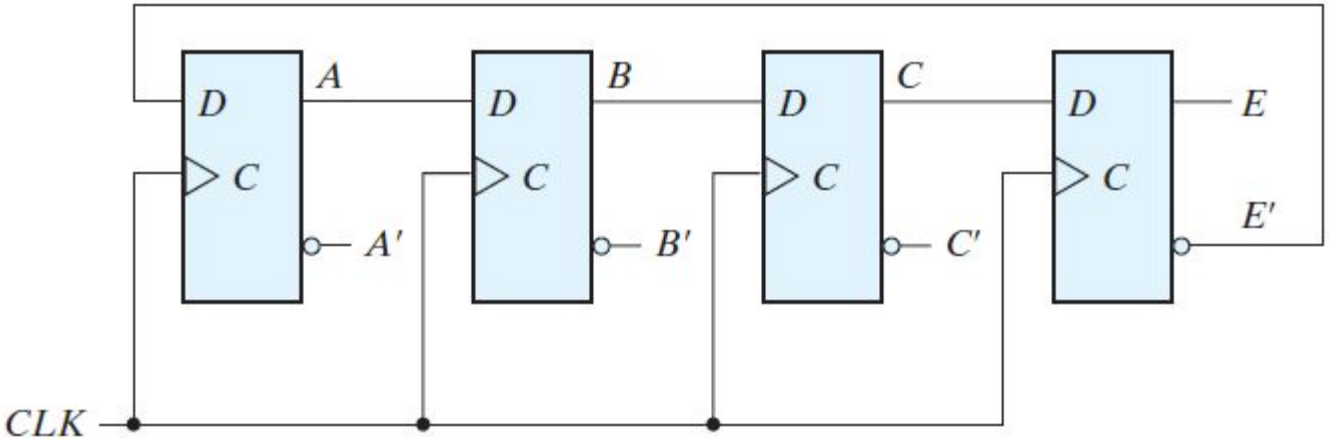



```
module counter #(parameter N = 8)
                (input          clk,
                 input          reset,
                 output reg     [N–1:0] q);

  always @(posedge clk, posedge reset)
    if (reset) q <= 0;
    else q <= q + 1;
endmodule
```

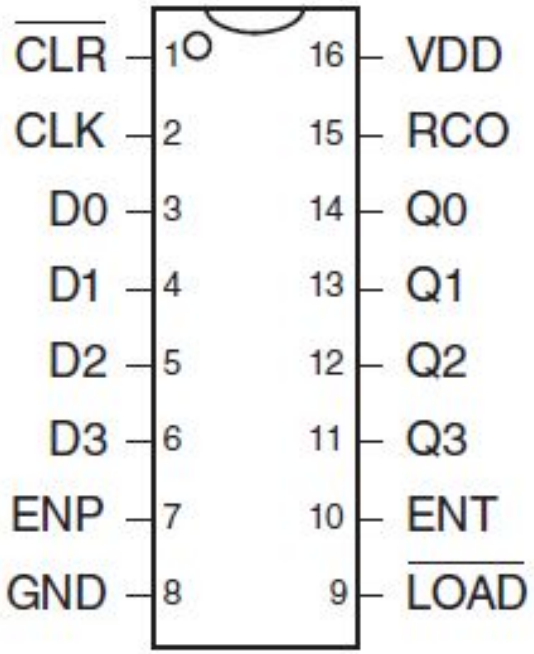
Tiene ***n*** flip-flops y 2^n estados, que se recorren en la secuencia 0, 1, 2, ..., 2^n-1, 0, 1, … Cada uno de los estados anteriores se codifica como el correspondiente entero binario de ***n*** bits.

CONTADOR BINARIO DE JOHNSON



Sequence number	Flip-flop outputs			
	A	*B*	*C*	*E*
1	0	0	0	0
2	1	0	0	0
3	1	1	0	0
4	1	1	1	0
5	1	1	1	1
6	0	1	1	1
7	0	0	1	1
8	0	0	0	1

CONTADOR DE 4 BITS COMERCIAL Y SU EQUIVALENCIA HDL



74161/163 Counter

4-bit Counter

CLK:	clock
$Q_{3:0}$:	counter output
$D_{3:0}$:	parallel input
CLRb:	async reset (161)
	sync reset (163)
LOADb:	load Q from D
ENP, ENT:	enables
RCO:	ripple carry out

```
always @(posedge CLK) // 74163
  if (~CLRb) Q <= 4'b0000;
  else if (~LOADb) Q <= D;
  else if (ENP & ENT) Q <= Q+1;

assign RCO = (Q == 4'b1111) & ENT;
```