Estructura de Computadores

Segundo curso

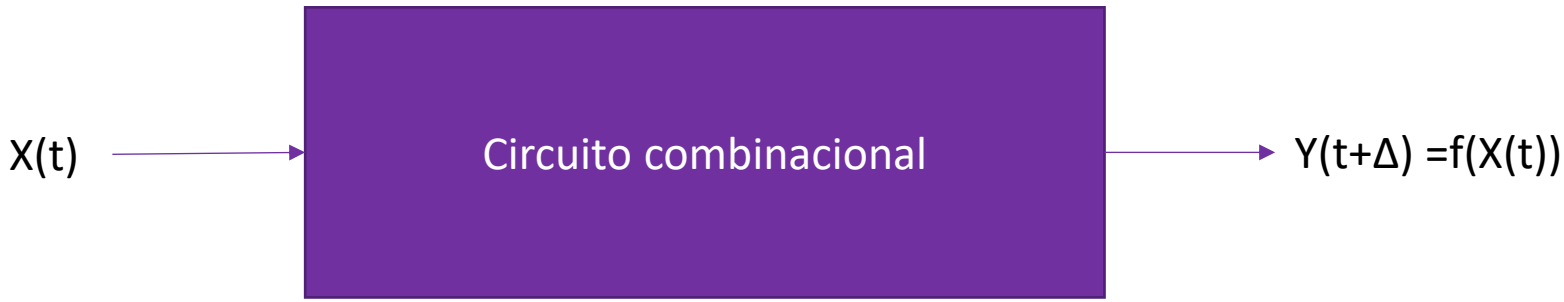
Prácticas

Actividades previas

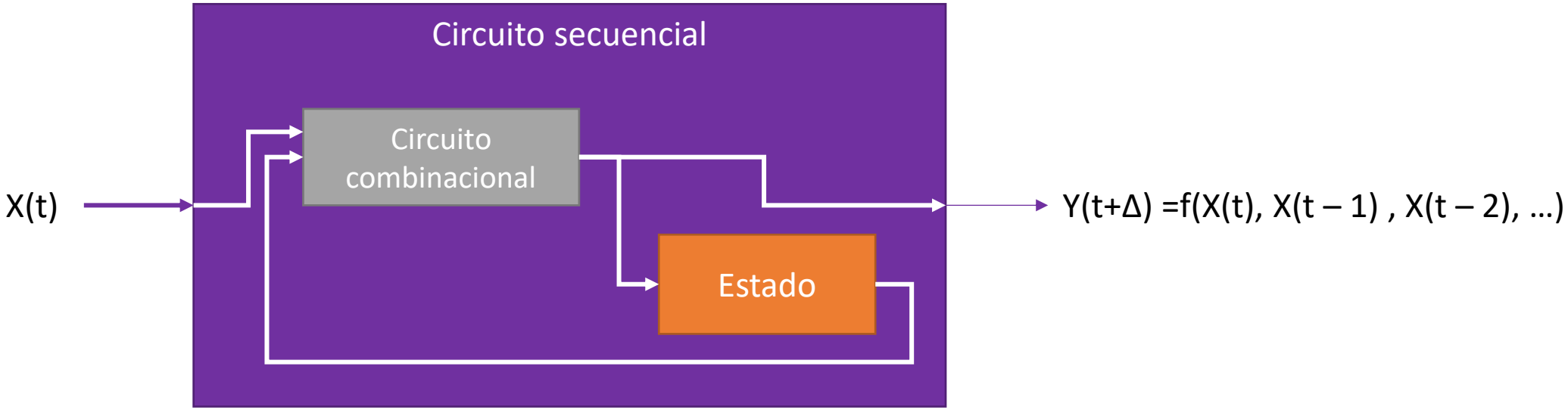
Práctica 2

Contenidos

- Modelado de circuitos lógicos secuenciales
- La señal de reloj
- Sistemas síncronos activados por flanco
  - Ejemplo: Biestables
- Asignaciones no bloqueantes
- Herramientas útiles de verilog
- Máquinas de estados finitos


X(t)
Circuito combinacional
Y(t+Δ) =f(X(t))

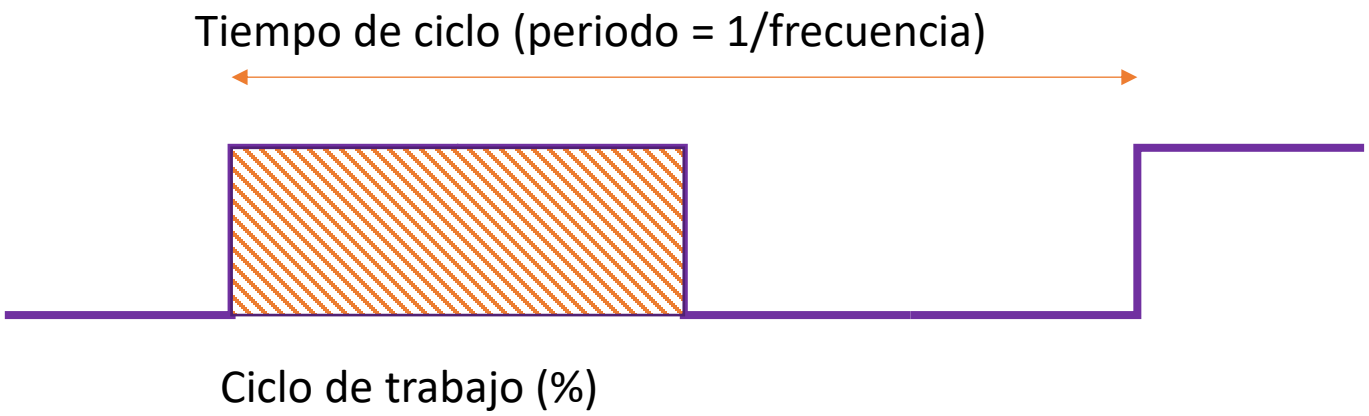
Modelado de sistemas secuenciales



Si necesitamos un circuito con 2^k estados,

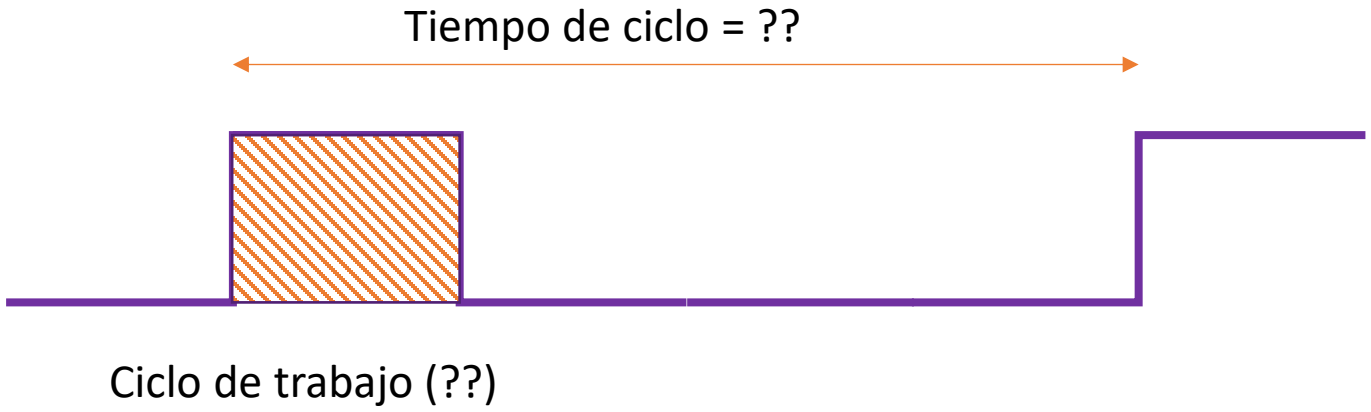
¿de cuántos elementos de estado binarios deberíamos disponer?

- Sistemas síncronos



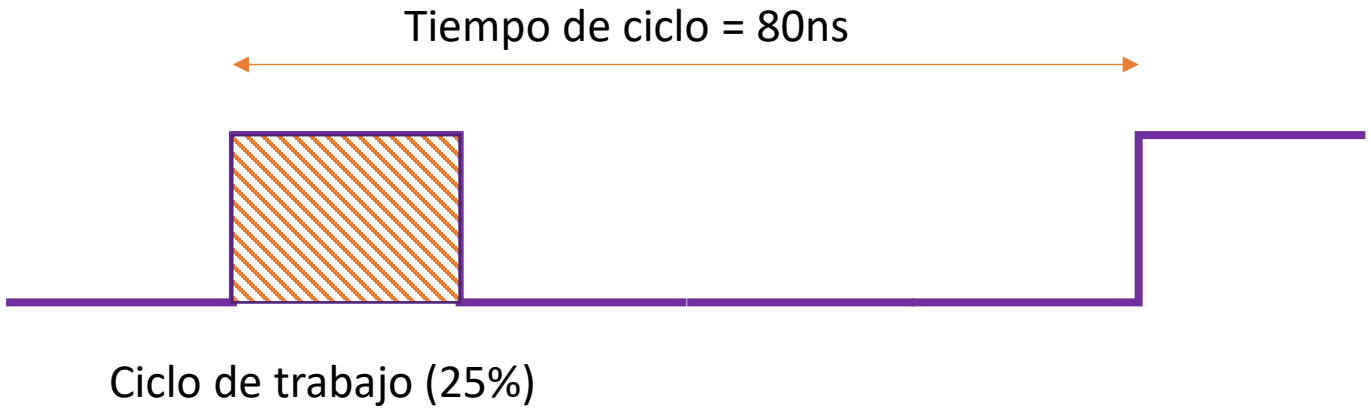
Señal de reloj en verilog

```
`timescale 1 ns / 10 ps
reg clk;

// generación de reloj clk
always //siempre activo, no hay condición de activación
begin
   clk = 1;
   #20;
   clk = 0;
   #60;
end
```

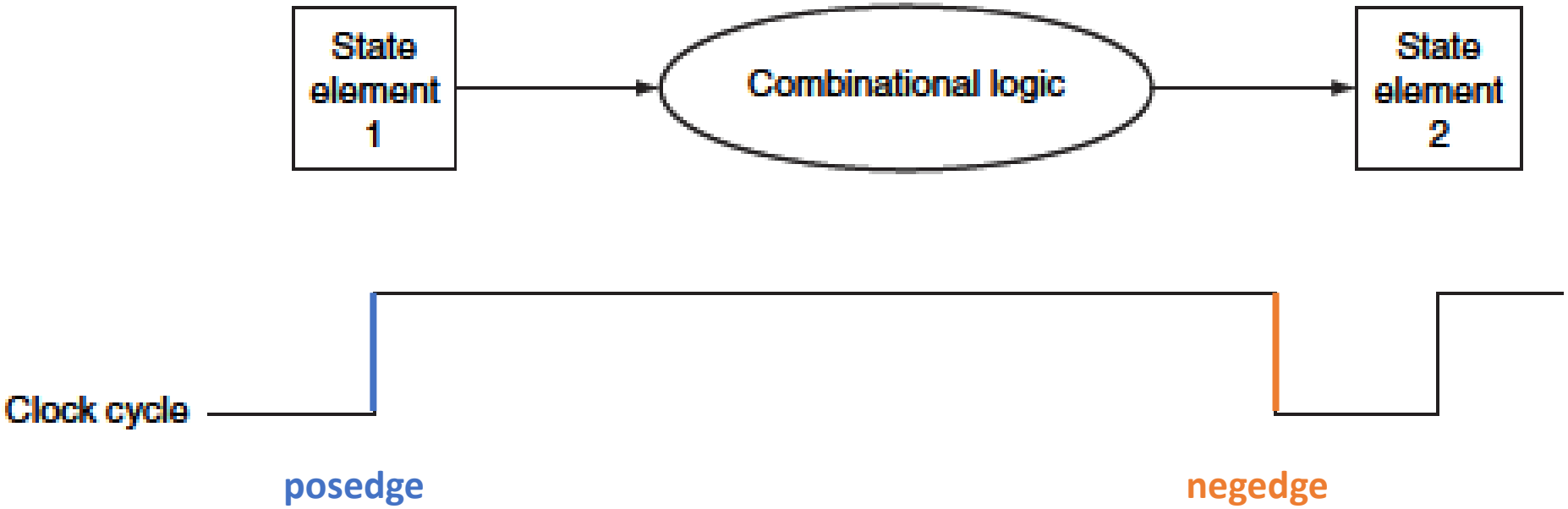


Señal de reloj en verilog

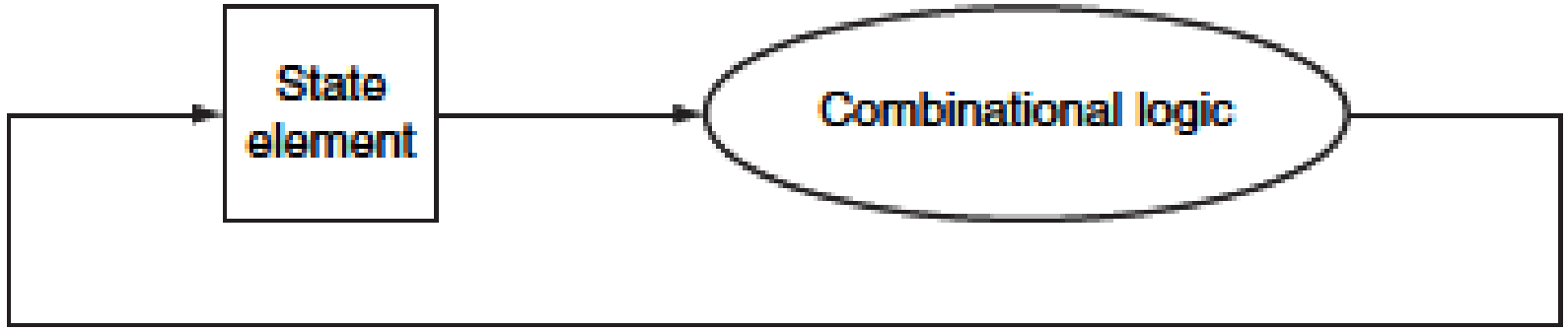
```
`timescale 1 ns / 10 ps
reg clk;

// generación de reloj clk
always //siempre activo, no hay condición de activación
begin
   clk = 1;
   #20;
   clk = 0;
   #60;
end
```



Sistemas síncronos activados por flanco

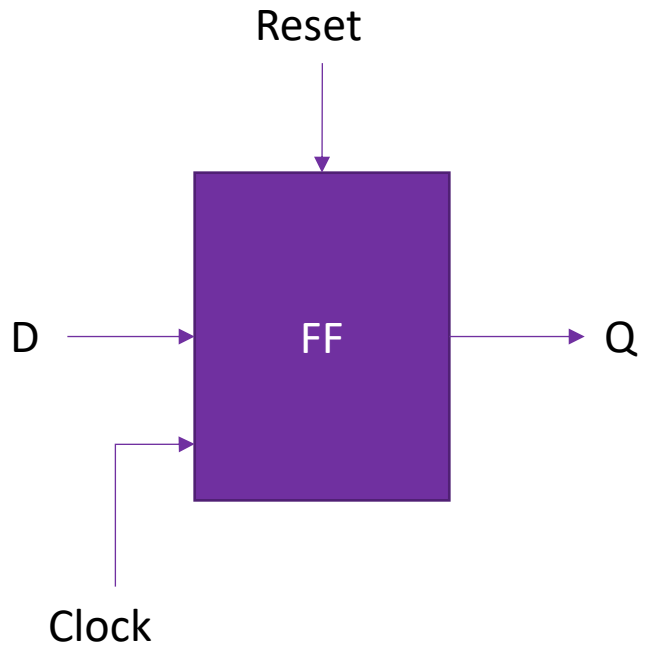


Sistemas síncronos activados por flanco



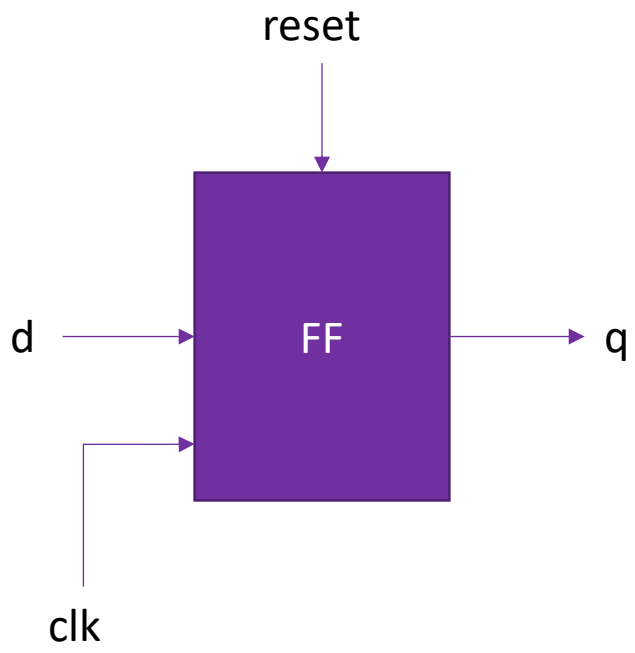
¡“Carreras”!

Ejemplo de elemento de almacenamiento



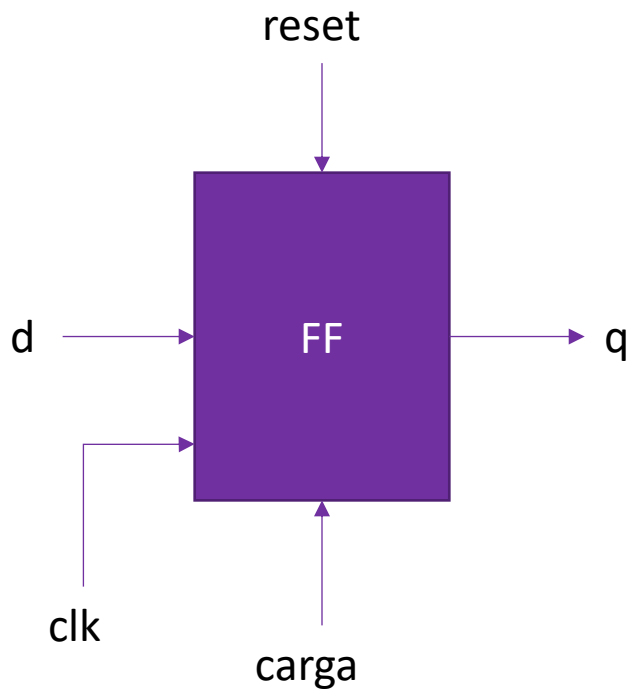
Reloj	D	Q	Q_{previo}
Flanco Subida	0	0	X
Flanco Subida	1	1	X
0	X	Q_{previo}	
1	X	Q_{previo}	

Implementación en verilog del FF




```
module ffd (output reg q, input wire clk, reset,
d);
//reset asíncrono, ya que es independiente del
flanco del reloj

always @(posedge clk, posedge reset)
//--> cuando se de el flanco en clk o reset
  if (reset)
    q <= 1'b0; //asignación no bloqueante q = 0
  else
    q <= d; //asignación no bloqueante q = d

endmodule
```

Implementación en verilog del FF (con línea de carga)




```
module ffd (output reg q, input wire clk, reset,
carga, d);
//reset asíncrono, ya que es independiente del
flanco del reloj

always @(posedge clk, posedge reset)
//--> cuando se de el flanco en clk o reset
  if (reset)
    q <= 1'b0; //asignación no bloqueante q = 0
  else
    if (carga)
      q <= d; //asignación no bloqueante q = d

endmodule
```

Asignaciones bloqueantes vs no bloqueantes

Bloqueante

```
reg [1:0] a, b;
initial
begin
    a = 1;
    b = 2;
    #10;
    a = b;
    b = a;
    #10;
end
```

- En general, se usa en las partes combinacionales de los circuitos

No bloqueante

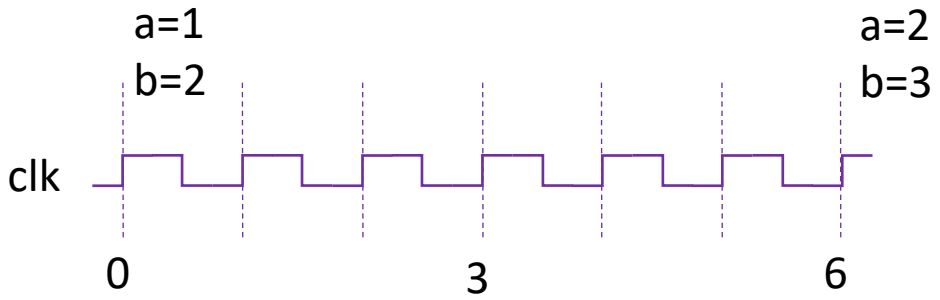
```
reg [1:0] a, b;
initial
begin
    a <= 1;
    b <= 2;
    #10;
    a <= b;
    b <= a;
    #10;
end
```

- En general, se usa para dar valor a los elementos de estado de los circuitos secuenciales

Asignaciones bloqueantes vs no bloqueantes: retardo intra-asignación

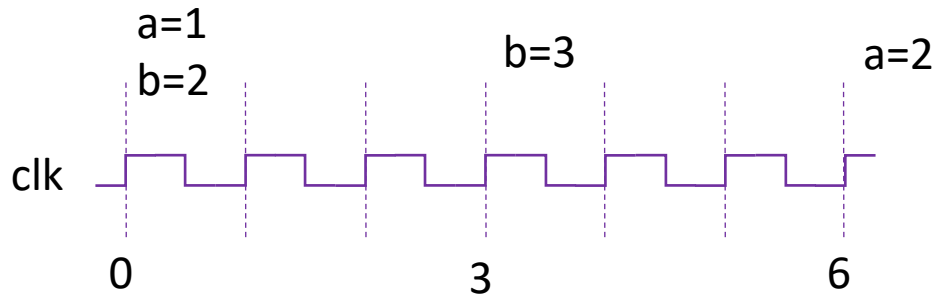
Bloqueante

```
reg [1:0] a, b;
initial
begin
    a = 1;
    b = 2;
    #3;
    a = #3 b;
    b = 3;
end
```



No bloqueante

```
reg [1:0] a, b;
initial
begin
    a <= 1;
    b <= 2;
    #3;
    a <= #3 b;
    b <= 3;
end
```



“Parámetros” en verilog

- Declaración del módulo

```
module nombremodulo #(parameter nombreparam = valorpordefecto)
(input wire a, input wire b, …);
```

- Instanciación del módulo

```
nombremodulo #(valor_real) pepemodulo(test_a, test_b, …);
```

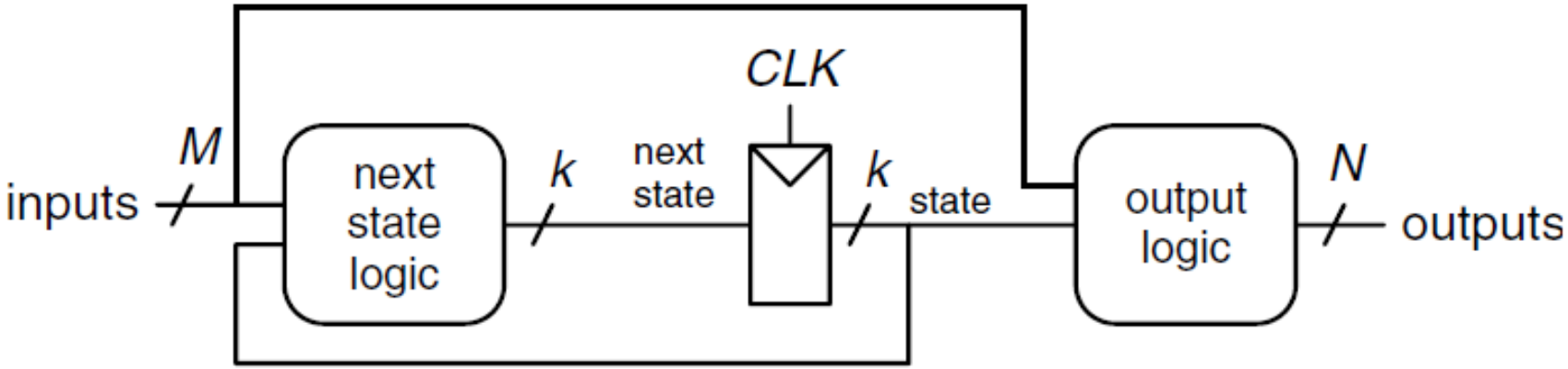

“Parámetros” en verilog: ejemplo

```
// Multiplexor de dos entradas y una salida, de tamaño
configurable. Si la señal de selección "s" vale 1, se
pone el valor de d1 en la salida; d0 si vale 0
module mux2 #(parameter WIDTH = 8)
             (output wire [WIDTH-1:0] Y,
              input  wire [WIDTH-1:0] D0, D1,
              input  wire             s);

  assign Y = s ? D1 : D0;
endmodule

//Se instancia con:
mux2 #(4) mimux(t_Y, t_D0, t_D1, t_s);
```

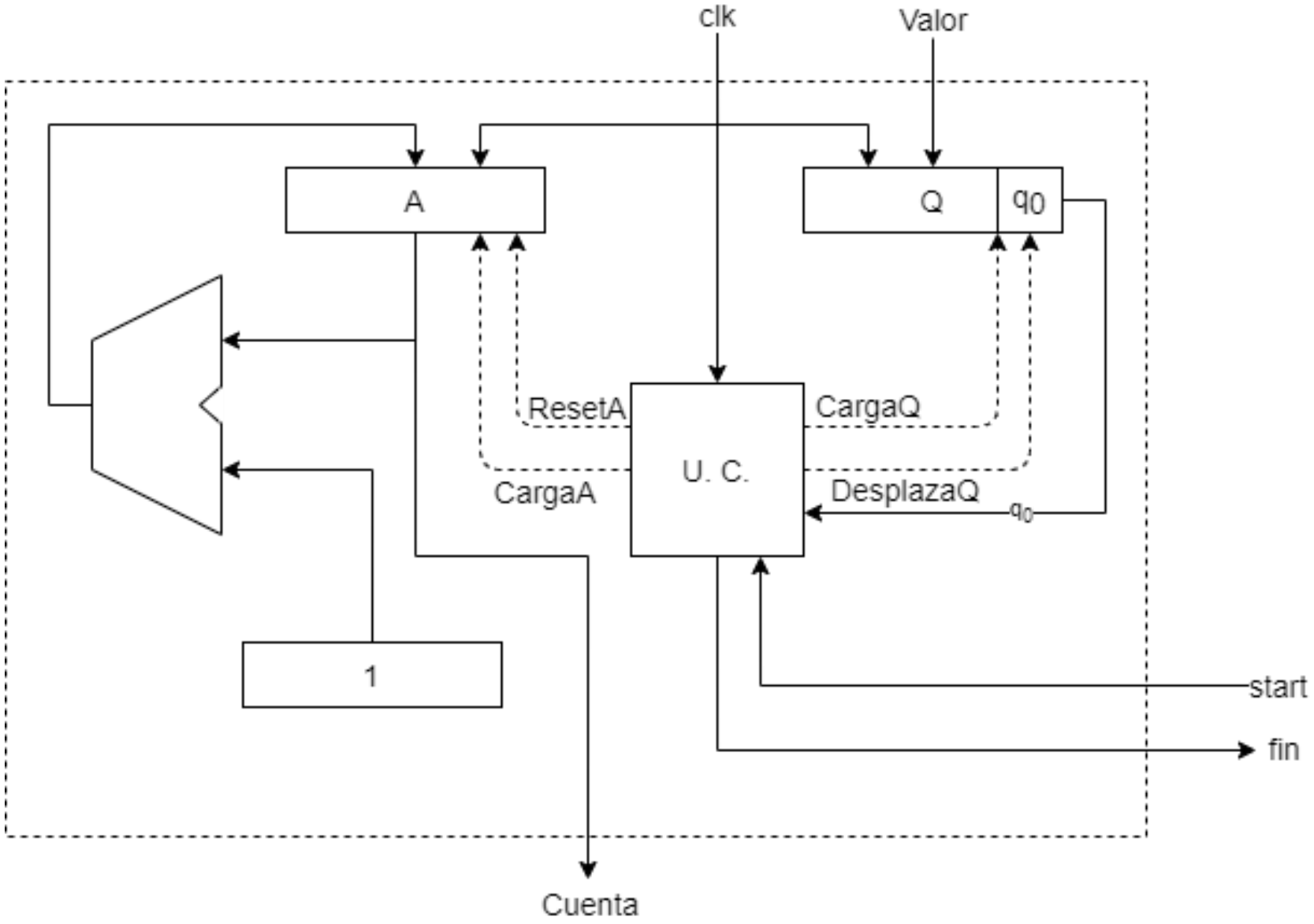

“Parámetros” en verilog: ejemplo

```
module ffd #(parameter retardo = 1)(input wire clk, reset, d,
output reg q);
//reset asíncrono, es independiente del flanco del reloj

always @(posedge clk, posedge reset)
  if (reset)
    q <= #retardo 1’b0; //asignación no bloqueante q = 0 con
retardo
  else
    q <= #retardo d; //asignación no bloqueante a q = d con
retardo
endmodule

//Se instancia con:
ffd #(3) miffd(t_clock, t_reset, t_d, t_q);
```


Máquina de estados finitos



Actividad práctica: ejemplo de máquina de estados



Actividad práctica: ejemplo de máquina de estados

```
module cuenta1(input wire [2:0] Valor, input wire start,
clk, output wire [3:0] Cuenta, output wire fin);
```

```
module uc(input wire q0, start, clk, output wire CargaQ,
DesplazaQ, ResetA, CargaA, Fin)
```

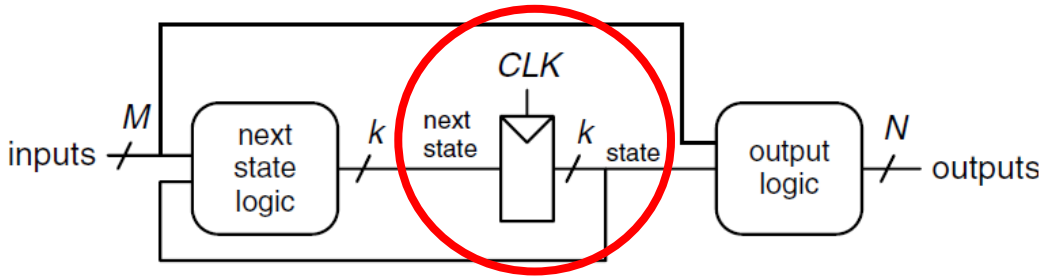

Actividad práctica: ejemplo de máquina de estados

Estado	Salidas a activar	Observaciones	Siguiente Estado
S0	CargaQ, ResetA	Estado inicial de carga del operando en Q e inicialización de A	S1
S1	Si q0==1, CargaA	Carga A con la salida del sumador	S2
S2	DesplazaQ	Actualizar q0	S3
S3	Si q0==1, CargaA	Carga A con la salida del sumador	S4
S4	DesplazaQ	Actualizar q0	S5
S5	Si q0==1, CargaA	Carga A con la salida del sumador	S6
S6	Fin	Estado final de la operación	S6

Actividad práctica: ejemplo de máquina de estados

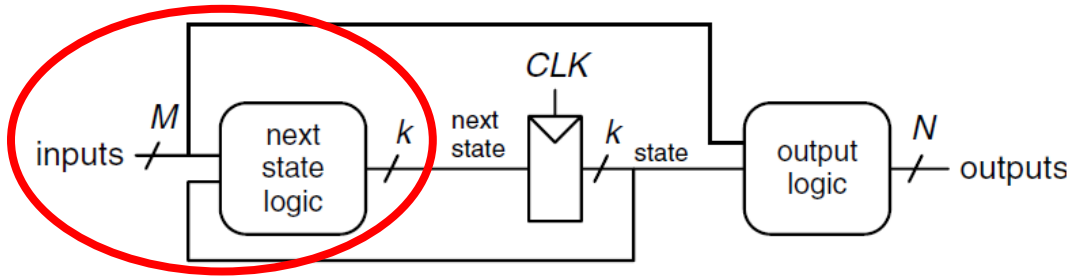
```
module uc(input wire q0, start, clk, output wire CargaQ, DesplazaQ,
ResetA, CargaA, Fin)

reg [2:0] state, nextstate;
//Variables para los estados actual y siguiente
//Tres bits son suficientes para los siete estados

//Codificación de los estados

parameter S0 = 3'b000;
//declaracion de constantes que representan estados
parameter S1 = 3'b001;
parameter S2 = 3'b010;
parameter S3 = 3'b011;
parameter S4 = 3'b100;
parameter S5 = 3'b101;
parameter S6 = 3'b110;
```


```
// Registro de estado, cambia en cada flanco ciclo
de reloj por el nuevo estado o
// se inicia en caso de flanco de subida de start al
estado inicial

always @ (posedge clk, posedge start)
  if (start)
    state <= S0;
  else
    state <= nextstate;
```

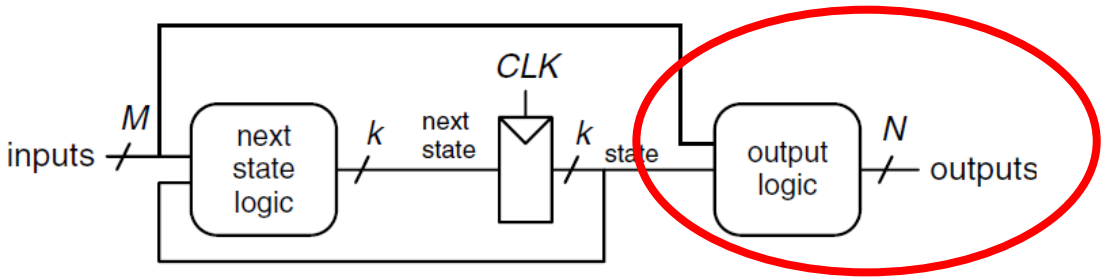


Actividad práctica: ejemplo de máquina de estados

```
//Función de Transición

always @(*)
// (*) significa cualquier cambio en alguna variable del bloque
  case (state)
    S0: nextstate = S1;
    S1: nextstate = S2;
    S2: nextstate = S3;
    S3: nextstate = S4;
    S4: nextstate = S5;
    S5: nextstate = S6;
    S6: nextstate = S6;
    default: nextstate = S0;
  endcase
```



```
// Función de Salida

assign CargaQ = (state == S0)? 1:0;
//a 1 si el estado es S0
assign DesplazaQ = ((state == S2)|(state == S4))? 1:0;
assign ResetA = (state == S0)? 1:0;
assign CargaA = (q0 & ((state == S1)|(state ==
S3)|(state == S5)))? 1:0;
assign Fin = (state == S6)? 1:0;


endmodule
```



Actividad práctica: ejemplo de máquina de estados

- A partir de los ficheros suministrados con esta actividad, realizar un módulo testbench cuenta1_tb.v que produzca las entradas al módulo global:
  - la señal de reloj como se ha visto antes,
  - un breve pulso inicial en la señal de start (ponerla a alta y un breve retardo después, a baja),
  - la entrada del valor a contar
- y observar la salida de Fin y de Cuenta.
- En alguno de los ficheros de los módulos principales se ha incluido el código de los módulos de los que éstos dependen para reducir el número de ficheros.

Actividad práctica: ejemplo de máquina de estados

- Observar también las señales de la unidad de control, usando el gtkwave que permite ‘abrir’ el módulo global cuenta1 y acceder a sus señales internas.
- Suponiendo que en la sentencia timescale el valor del retardo (primer parámetro) es 1ns, intentar decidir experimentalmente cuál sería la frecuencia máxima de reloj (o el periodo mínimo) al que podría funcionar este circuito.