
# Instituto Politécnico Nacional

# Centro de Investigación en Computación

## Lenguajes de descripción de hardware

## Práctica 3 - Circuitos secuenciales 1

Profesor:

M. en C. Osvaldo Espinosa Sosa

Por:

Ing. Ricardo Aldair Tirado Torres

Ciudad de México, 12 de abril de 2024

# Tabla de contenido

# 1. Objetivos

- Comprender la operación e implementación de circuitos secuenciales importantes como el flip-flop tipo D y el contador, así como sus variantes.
- Diferenciar la implementación de un *reset* asíncrono de uno síncrono en dos flip-flop tipo D utilizando el visor RTL.
- Aprender a describir un contador con múltiples terminales de control, como el *reset* asíncrono, *clock enable* (habilitador de la señal de reloj), carga de datos y sentido del conteo, además de la generación de un tope a este mismo conteo.

# 2. Flip-Flop tipo D con *reset* asíncrono

**Actividad 1**

Codificar un flip-flop tipo D con *reset* asíncrono. Compilar y simular. Usar el visor RTL para observar como se implementa el circuito. Configurar en la tarjeta DE2-115, asignar interruptores y un LED.

La visualización RTL del flip flop tipo D con *reset* asíncrono en Verilog se muestra en la Figura 1. En el visor se observa que la señal RST se conecta al pin CLRN que sirve para limpiar el estado del flip flop, dicho estado se presenta en el pin SCLR, indicando que al restablecer el circuito, la salida tendrá el valor de esta terminal (cero). Las señales restantes se conectan a los pines correspondientes del modulo (D, CLK y Q).

Las simulaciones para el código en Verilog se visualizan en la Figura 2. Se observa que la salida Q adquiere el valor de la entrada D unicamente cuando en la señal de reloj (CLK) hay un flanco de subida. La limpieza en el flip flop se da de manera asíncrona con el pin RST, ya que no depende de CLK, sino que cuando se tiene un flanco de subida en el *reset*, se ajusta el valor de la salida a 0.

En los Anexos se localiza la descripción del flip flop tipo D con *reset* asíncrono. Se utilizó una lista sensible para el flanco de subida de la señal de reloj y el *reset*. Dentro de esta estructura se empleó la sentencia *if* para comparar el valor de RST:

- Si es 1, se restablece el valor de la salida a 0.
- Si es 0, la salida adquiere el valor de D.

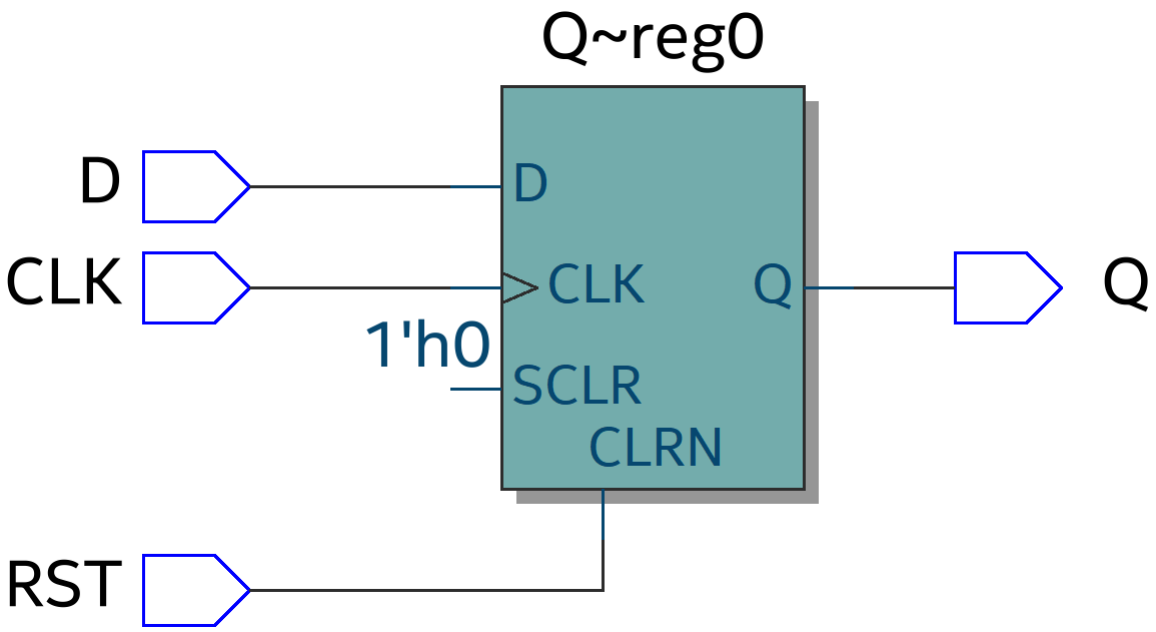


Figura 1: Diagrama RTL del flip flop tipo D con *reset* asíncrono.

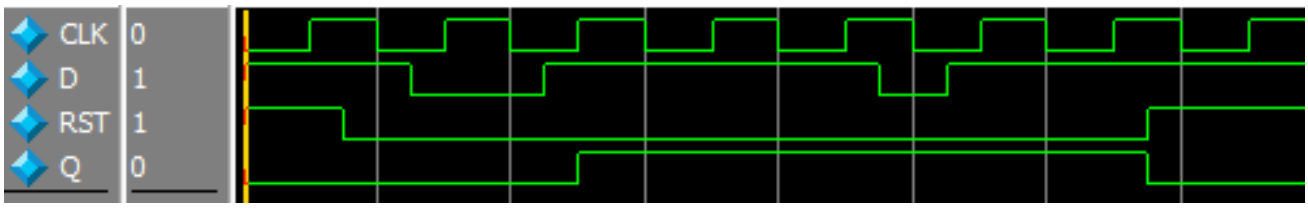


Figura 2: Simulación del flip flop tipo D con *reset* asíncrono en el visor de formas de onda de ModelSim.

# 3. Flip-Flop tipo D con *reset* síncrono

**Actividad 2**

Codificar un flip-flop tipo D con *reset* síncrono. Compilar y simular. Usar el visor RTL para observar como se implementa el circuito y mencionar diferencias con el circuito del inciso 1. Configurar en la tarjeta DE2-115, asignar interruptores y un LED.

La visualización RTL del flip flop tipo D con *reset* síncrono en Verilog se muestra en la Figura 3. En el visor se observa que la señal RST se conecta como señal de control de un multiplexor, cuya salida esta conectada a la terminal D del flip flop. Si el valor de RST es 0, entonces la salida adquiere el valor de la señal D, pero si el valor es 1, la salida será 0 (Este valor sería la representación del estado de restablecimiento). Las señales restantes se conectan a los pines correspondientes del modulo (CLK y Q).

Las simulaciones para el código en Verilog se visualizan en la Figura 4. Se observa que la salida Q adquiere el valor de la entrada D unicamente cuando en la señal de reloj (CLK) hay un flanco de subida. La limpieza en el flip flop se da de manera síncrona con el pin RST, ya que unicamente cuando CLK tiene un flanco de subida, el *reset* ajustará el valor de la salida a 0.

En los Anexos se localiza la descripción del flip flop tipo D con *reset* síncrono. Se utilizó una lista sensible para el flanco de subida de la señal de reloj. Dentro de esta estructura se empleó la sentencia *if* para comparar el valor de RST:

- Si es 1, se restablece el valor de la salida a 0.
- Si es 0, la salida adquiere el valor de D.

En la Figura 5 se observa la comparativa de simulaciones del flip flop con *reset* síncrono y asíncrono, viendo que el primero restablece el valor de la salida Q hasta que CLK tiene un flanco de subida, en cambio, el segundo hace el restablecimiento independientemente de la señal de reloj. Igualmente, se diferencian ambas implementaciones en las conexiones que se tienen en el RTL y en la descripción de cada modulo.

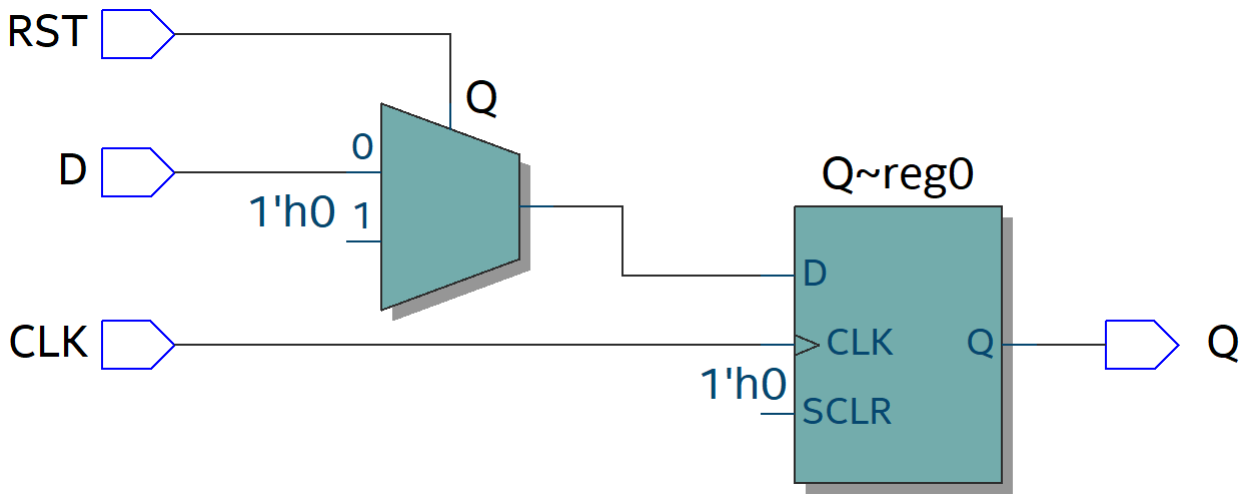


Figura 3: Diagrama RTL del flip flop tipo D con *reset* síncrono.

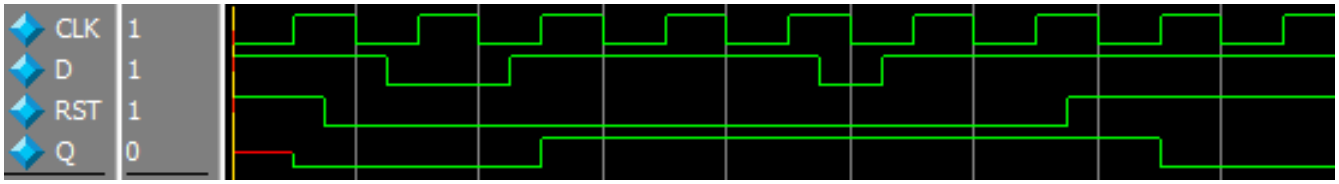


Figura 4: Simulación del flip flop tipo D con *reset* síncrono en el visor de formas de onda de ModelSim.

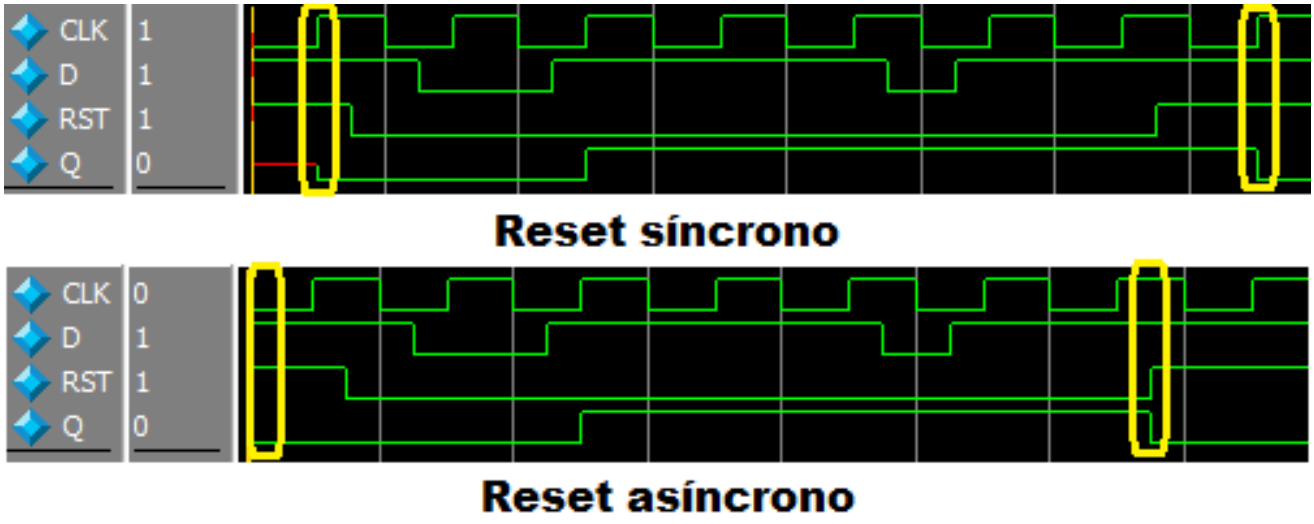


Figura 5: Comparativa del flip flop tipo D con *reset* síncrono y asíncrono en el visor de formas de onda de ModelSim. Se observa que las señales de *reset* son las mismas pero la ejecución de esta operación varía en cada caso.

# 4. Contador de 8 bits con *reset* asíncrono

**Actividad 3**

Codificar un contador con *reset* asíncrono. Compilar y simular. Usar el visor RTL para observar como se implementa el circuito. Configurar en la tarjeta DE2-115, asignar interruptores y LED's para observar la cuenta.

La visualización RTL del contador de 8 bits con *reset* asíncrono en Verilog se muestra en la Figura 6. La implementación se hace utilizando un flip flop tipo D de 8 bits junto con una instancia de sumador de 8 bits. La señal RST se conecta al pin CLRN que sirve para limpiar el estado del contador, dicho estado se presenta en el pin SCLR, indicando que al restablecer el circuito, la salida tendrá el valor de esta terminal (para este caso, cero). La señal CLK se conecta a la terminal de reloj del flip flop y la salida Q hace una retroalimentación a la entrada D, por medio de un sumador, que incrementará el valor en 1 por cada ciclo de reloj.

Las simulaciones para el código en Verilog se visualizan en la Figura 7 y Figura 8. Se observa que la salida Q incrementa su valor cuando hay un flanco de subida en el ciclo de reloj. La limpieza en el flip flop se da de manera asíncrona con el pin RST, ya que no depende de CLK, sino que cuando se tiene un flanco de subida en el *reset*, se ajusta el valor de la salida a 0. Finalmente se ve que una vez que la cuenta llega al máximo valor posible (255 para 8 bits), hay un desbordamiento ya que se pierde el bit de acarreo, haciendo que parezca que la cuenta regresa al valor inicial.

En los Anexos se localiza la descripción del contador con *reset* asíncrono. Se utilizó una lista sensible para el flanco de subida de la señal de reloj y el *reset*. Dentro de esta estructura se empleó la sentencia *if* para comparar el valor de RST:

- Si es 1, se restablece el valor de la salida a 0.
- Si es 0, la salida adquiere el valor de D más un incremento de 1.

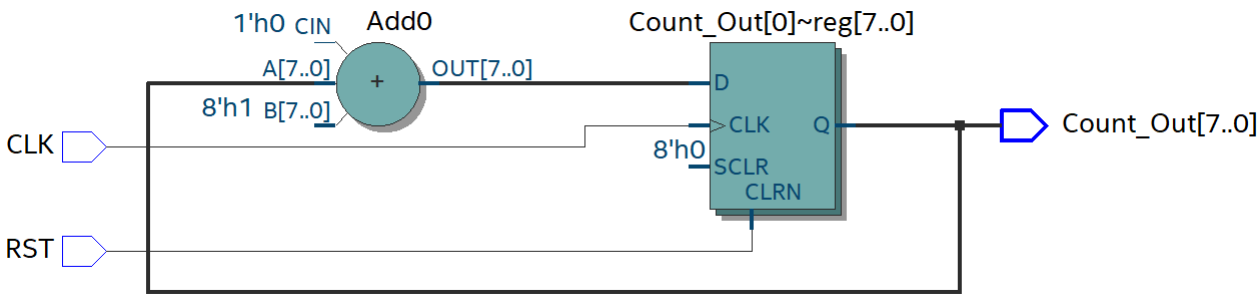


Figura 6: Diagrama RTL del contador de 8 bits con *reset* asíncrono.

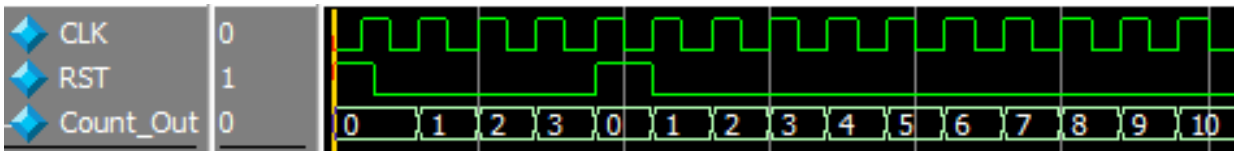


Figura 7: Simulación del contador de 8 bits con *reset* asíncrono en el visor de formas de onda de ModelSim (Uso del *reset*).

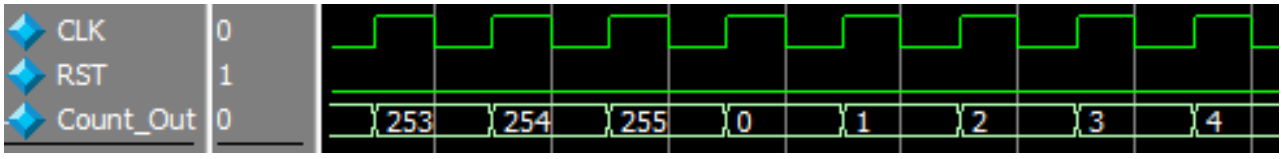


Figura 8: Simulación del contador de 8 bits con *reset* asíncrono en el visor de formas de onda de ModelSim (Reinicio de la cuenta).

# 5. Contador de 8 bits con *reset* asíncrono, *clock enable*, carga de datos, sentido y tope del conteo

**Actividad 4**

Codificar un contador con *reset* asíncrono, *clock enable*, carga de datos y sentido del conteo; agregando un tope al conteo (por ejemplo si lo quisiéramos para contar segundos, debería contar de 0 a 59). Usar el visor RTL para observar la implementación. Considerar que son varios casos de simulación (por todas las terminales de control que tiene el contador).Configurar en la tarjeta DE2-115, asignar interruptores y LED's para observar la cuenta.

La visualización RTL del contador de 8 bits en Verilog se muestra en la Figura 9. La implementación se hace utilizando un flip flop tipo D de 8 bits junto con varias instancias de sumadores, comparadores y multiplexores. La señal RST se conecta al pin CLRN que sirve para limpiar el estado del contador, dicho estado se presenta en el pin SCLR, indicando que al restablecer el circuito, la salida tendrá el valor de esta terminal (para este caso, 0). La señal CLK se conecta a la terminal de reloj del flip flop y la señal CE se conecta al pin de habilitación del modulo (ENABLE). Con respecto a los sumadores, se observa que el primero realiza la cuenta ascendente, mientras que el segundo la hace de forma descendente. La salida de cada sumador se conecta a un multiplexor que es controlado por un comparador, que evalúa si se ha llegado al valor máximo; si es así, la salida de los multiplexores será el valor de tope (para la cuenta descendente) o el valor mínimo (para la cuenta ascendente). Ambas salidas se conectan a la entrada de otro multiplexor cuya señal de control es *Up Down* (determinando el sentido de la cuenta). Finalmente la salida se dirige a un último multiplexor controlado por la señal LD, que determina si en la entrada D se tiene a la salida del último multiplexor o a la señal *Count In*.

Las simulaciones para el código en Verilog se visualizan en la Figura 10. Ahora bien, para evaluar el funcionamiento de cada característica del contador de 8 bits se hizo lo siguiente:

- En la Figura 11 se observa el funcionamiento de la señal RST (*reset*), que restablece la cuenta de manera asíncrona.
- En la Figura 12 se observa el funcionamiento de la señal CE (*clock enable*), que detiene

la cuenta debido a que deshabilita a la señal de reloj, por lo que no se incrementa (o decrementa) el último valor obtenido en la entrada.

- En la Figura 13 se observa el funcionamiento de la señal LD (*load data*), que cambia el valor de la cuenta al que se tiene en la señal *Count In*. Cabe resaltar que no se genera una cuenta (a partir del valor cargado) hasta que se pone a LD en bajo.
- En la Figura 14 se observa el funcionamiento de la señal *Up Down*, que cambia el sentido de la cuenta a forma ascendente o descendente según su valor.
- En la Figura 15 se observa el funcionamiento del tope de la cuenta, lo cual significa que, en lugar de contar hasta un valor máximo, cuando se alcanza un valor en concreto (para este ejemplo es 59), se reinicia la cuenta.

En los Anexos se localiza la descripción del contador de 8 bits. Se utilizó una lista sensible para el flanco de subida de la señal de reloj y el *reset*. Dentro de esta estructura se emplearon múltiples sentencias *if* anidados para comparar todas las señales de control y evaluar si la cuenta alcanzó el valor de tope. Es importante señalar que el nivel de prioridad es:

1. *Reset* (RST).
2. *Clock enable* (CE).
3. *Load data* (LD).
4. Sentido de la cuenta (Up Down).
5. Tope en la cuenta.

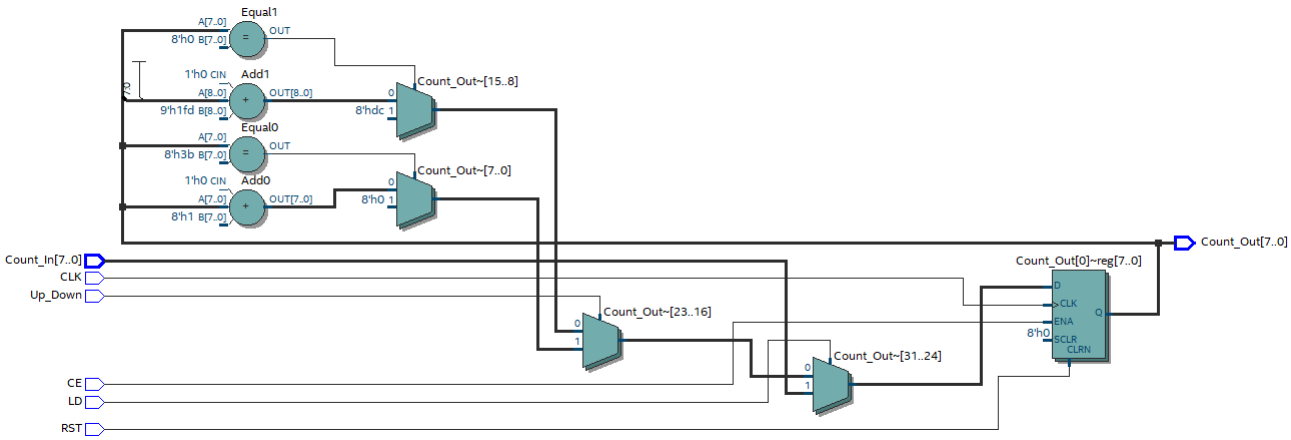


Figura 9: Diagrama RTL del contador completo de 8 bits.

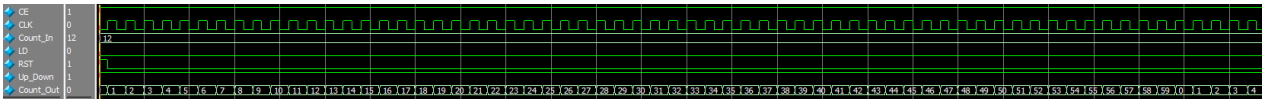

Figura 10: Simulación del contador completo de 8 bits en el visor de formas de onda de ModelSim.

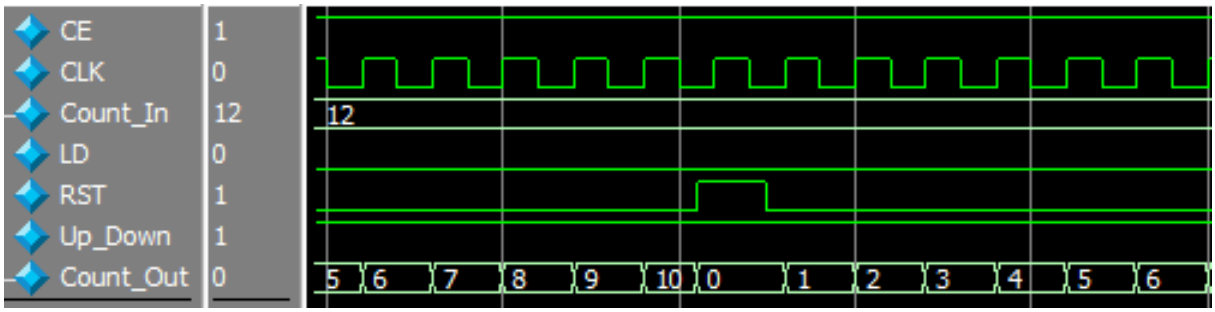


Figura 11: Funcionamiento del *reset* asíncrono, implementado en el contador completo de 8 bits.

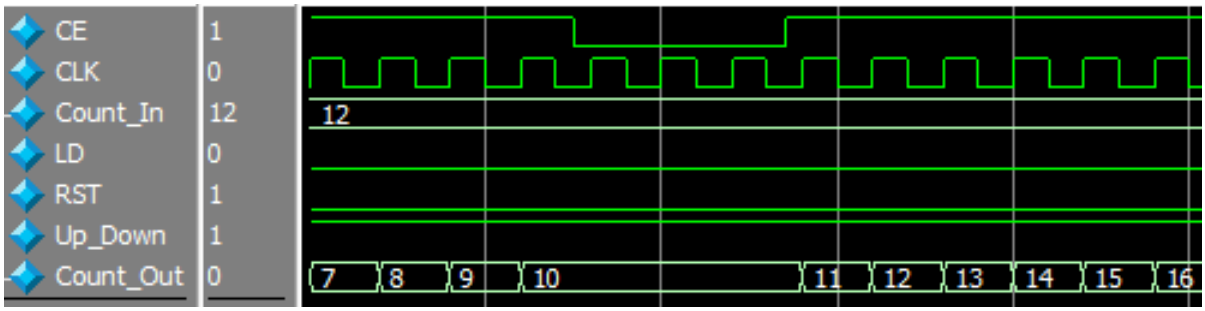


Figura 12: Funcionamiento del *clock enable*, implementado en el contador completo de 8 bits.

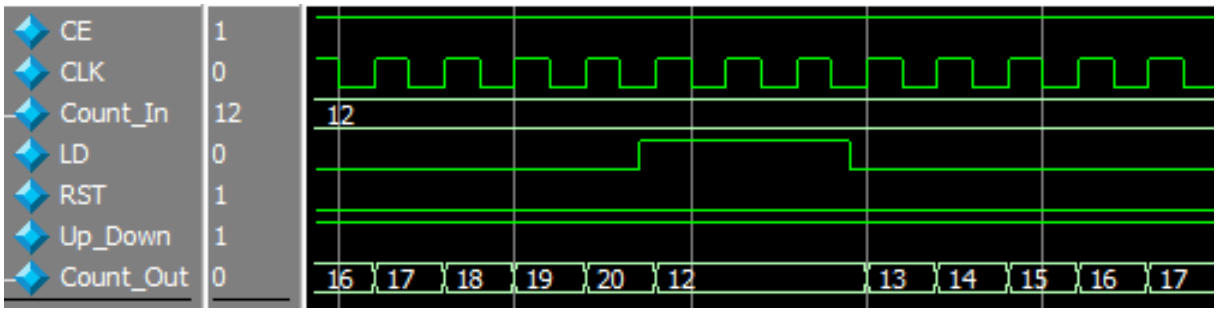


Figura 13: Funcionamiento de la carga de datos a la entrada, implementada en el contador completo de 8 bits.

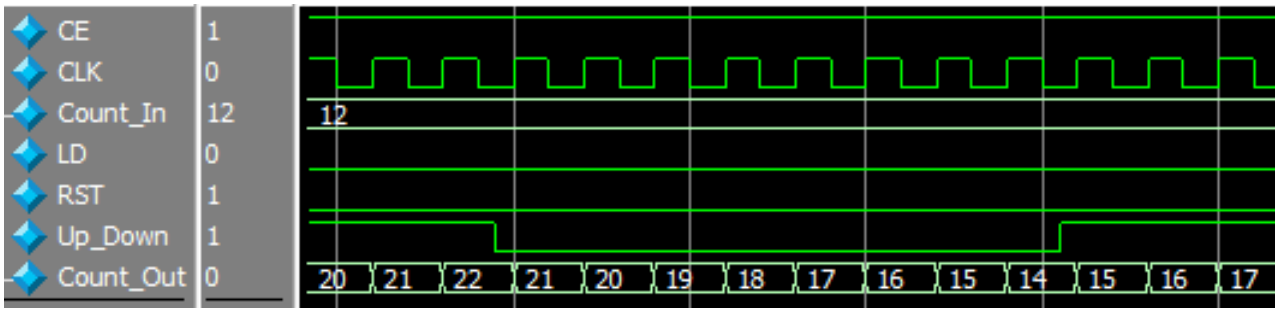


Figura 14: Funcionamiento del sentido de la cuenta, implementada en el contador completo de 8 bits.

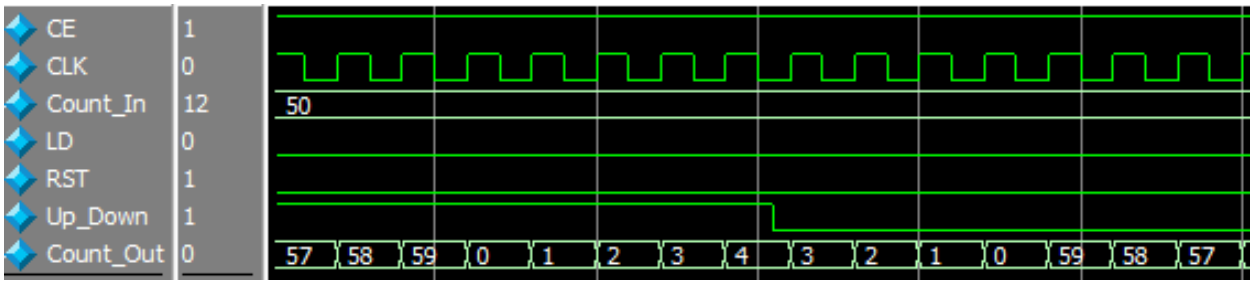


Figura 15: Funcionamiento del tope en la cuenta, implementado en el contador completo de 8 bits.

# 6. Conclusiones

En conclusión, se implementaron los 4 circuitos en lenguaje Verilog de manera exitosa.

Para los flip flop tipo D, se implementaron de manera correcta y se diferenció el funcionamiento del *reset* asíncrono del síncrono, así como sus características físicas en el visor RTL.

Para los contadores de 8 bits, se describieron de forma adecuada. Para el contador sencillo se observo como se implementa en el visor RTL, utilizando unicamente un flip flop de 8 bits junto con un sumador en el lazo de retroalimentación. Para el contador completo, se entendió como implementar la señales de control con respecto a un orden de prioridad y se observó la complejidad de este modulo al implementar no solo un flip flop de 8 bits, sino también sumadores, multiplexores y comparadores.

Se comprobó su funcionamiento utilizando las simulaciones de forma de onda en ModelSim y se asignaron los pines correspondientes en la placa de desarrollo para programar el dispositivo y realizar las pruebas pertinentes en hardware.

En los Anexos se pueden encontrar los códigos implementados junto con sus respectivos bancos de pruebas.

# 7. Anexos

## 7.1. Descripciones del hardware

```
module FF_D_Asyn_RST(
 input     CLK,
 input     RST,
 input     D,
 output reg Q
);

always @(posedge CLK or posedge RST)
begin
 if (RST)
  Q <= 1'b0;
 else
  Q <= D;
 end

endmodule
```

Programa 1: Descripción en Verilog del flip flop tipo D con *reset* asíncrono.

```
module FF_D_Syn_RST(
 input     CLK,
 input     RST,
 input     D,
 output reg Q
);

always @(posedge CLK)
begin
 if (RST)
  Q <= 1'b0;
 else
  Q <= D;
 end

endmodule
```

Programa 2: Descripción en Verilog del flip flop tipo D con *reset* síncrono.

```
module Counter_Asyn_RST(
 input        CLK,
 input        RST,
```

```
 output reg [7:0]  Count_Out
);

always @(posedge CLK or posedge RST)
begin
 if (RST)
  Count_Out <= 8'b00000000;
 else
  Count_Out <= Count_Out + 1'b1;
end

endmodule
```

Programa 3: Descripción en Verilog del contador con *reset* asíncrono.

```
module Counter_Full(
 input        CLK,
 input        RST,
 input        CE,
 input        LD,
 input        Up_Down,
 input    [7:0]  Count_In,
 output reg [7:0]  Count_Out
);

always @(posedge CLK or posedge RST)
begin
 if (RST)
 begin
  Count_Out <= 8'b00000000;
 end
 else if (CE)
 begin
  if (LD)
  begin
   Count_Out <= Count_In;
  end
  else
  begin
   if (Up_Down)
   begin
    if (Count_Out == 8'b00111011) //Valor de tope: 59
    begin
     Count_Out <= 8'b00000000;
    end
    else
```

```
32      begin
33       Count_Out <= Count_Out + 1'b1;
34      end
35     end
36     else
37     begin
38      if (Count_Out == 8'b00000000)
39      begin
40       Count_Out <= 8'b00111011;  //Valor de tope: 59
41      end
42      else
43      begin
44       Count_Out <= Count_Out - 1'b1;
45      end
46     end
47    end
48   end
49  end
50
51  endmodule
```

Programa 4: Descripción en Verilog del contador con *reset* asíncrono, *clock enable*, carga de datos, sentido y tope de la cuenta.

## 7.2. Bancos de pruebas (*Test Benches*)

```
`timescale 1 ns/ 1 ps
module FF_D_Asyn_RST_vlg_tst();
 reg CLK;
 reg D;
 reg RST;
 wire Q;

 FF_D_Asyn_RST i1 (
  .CLK(CLK),
  .D(D),
  .Q(Q),
  .RST(RST)
 );

 initial
 begin
  CLK = 0; RST = 1; D = 1;
  #15; RST = 0;
  #10; D = 0;
  #20; D = 1;
  #50; D = 0;
  #10; D = 1;
  #30; RST = 1; D = 1;
  $display("Running testbench at CIC");
 end

 always
 begin
  #10; CLK = ~CLK;
 end

endmodule
```

Programa 5: Banco de prueba para el Programa 1.

```
`timescale 1 ns/ 1 ps
module FF_D_Syn_RST_vlg_tst();
 reg CLK;
 reg D;
 reg RST;
 wire Q;

 FF_D_Syn_RST i1 (
  .CLK(CLK),
```

```
  .D(D),
  .Q(Q),
  .RST(RST)
 );

 initial
 begin
  CLK = 0; RST = 1; D = 1;
  #15; RST = 0;
  #10; D = 0;
  #20; D = 1;
  #50; D = 0;
  #10; D = 1;
  #30; RST = 1; D = 1;
  $display("Running testbench at CIC");
 end

 always
 begin
  #10; CLK = ~CLK;
 end

endmodule
```

Programa 6: Banco de prueba para el Programa 2.

```
`timescale 1 ns/ 1 ps
module Counter_Asyn_RST_vlg_tst();
 reg      CLK;
 reg      RST;
 wire [7:0]   Count_Out;

 Counter_Asyn_RST i1 (
  .CLK(CLK),
  .Count_Out(Count_Out),
  .RST(RST)
 );

 initial
 begin
  CLK=0; RST=1;
  #15;    RST=0;
  #75;    RST=1;
  #20;    RST=0;
  $display("Running testbench at CIC");
 end
```

```
21
22  always
23  begin
24   #10; CLK=~CLK;
25  end
26
27 endmodule
```

Programa 7: Banco de prueba para el Programa 3.

```
1  `timescale 1 ns/ 1 ps
2  module Counter_Full_vlg_tst();
3   reg     CE;
4   reg     CLK;
5   reg [7:0]   Count_In;
6   reg     LD;
7   reg     RST;
8   reg     Up_Down;
9   wire [7:0]   Count_Out;
10
11  Counter_Full i1 (
12   .CE(CE),
13   .CLK(CLK),
14   .Count_In(Count_In),
15   .Count_Out(Count_Out),
16   .LD(LD),
17   .RST(RST),
18   .Up_Down(Up_Down)
19  );
20
21  initial
22  begin
23   CLK = 0; RST = 1; CE = 1; LD = 0; Up_Down = 1; Count_In = 12;
24   #10; RST = 0;
25   #1300;
26   // Prueba de funcionamiento de RST
27   #95; RST = 1;
28   #20; RST = 0;
29   // Prueba de funcionamiento de CE
30   #200; CE = 0;
31   #60; CE = 1;
32   // Prueba de funcionamiento de LD
33   #200; LD = 1;
34   #60; LD = 0;
35   // Prueba de funcionamiento de Up_Down
36   #200; Up_Down = 0;
```

```
37   #160; Up_Down = 1;
38   // Prueba de funcionamiento del tope de conteo
39   #500; Count_In = 50; LD = 1;
40   #20; LD = 0;
41   #280; Up_Down = 0;
42   $display("Running testbench at CIC");
43  end
44
45  always
46  begin
47   #10; CLK = ~CLK;
48  end
49
50 endmodule
```

Programa 8: Banco de prueba para el Programa 4.