Universidad de Costa Rica
Facultad de Ingeniería
Escuela de Ingeniería Eléctrica
IE0523 - Circuitos Digitales II
I ciclo 2023

Tarea 2

Descripción estructural del sumador sincrónico de 4 bits: síntesis automática

Isaí David Vargas Ovares B88263
Grupo 1

Profesor: Enrique Coen Alfaro

23 de abril de 2023

1. Resumen

En tareas anteriores se realizó un sumador binario sincronizado por una señal de reloj, con modos de suma, resta, mantener el valor anterior y limpiar el sumador. Esto teniendo el módulo sumador de 4 bits, en este caso este realiza las operaciones bit por bit, como se señala en [1] las operaciones bit a bit son esenciales en la programación de sistemas embebidos y en el diseño de circuitos digitales. Analizar bit a bit significa que se procesa cada bit de los operandos individualmente, esto con el propósito de mejorar la eficiencia en términos de velocidad de ejecución y uso de memoria.

Se establecieron algunas pruebas, al igual que en anteriores tareas para comprobar el correcto funcionamiento del sumador, en este caso se probaron los distintos modos de funcionamiento del sumador (suma, resta, mantener valor y limpiar), imprimiendo los resultados en terminal para analizarlos, además en este caso se realiza el proceso de síntesis para producir una descripción estructural genérica (RTLIL), una descripción estructural usando componentes comercialmente disponibles, además del análisis al introducir retardos en los componentes utilizados.

2. Descripción Arquitectónica

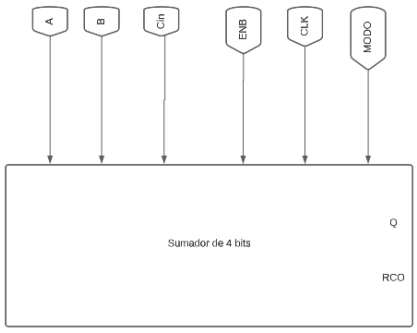

Figura 1: Diagrama de bloques con las señales más importantes.

Se diseña un Sumador binario sincrónico de 4, capaz de realizar suma, resta, mantener valor anterior y limpiar el sumador. El módulo sumador de 4 bits, este tiene como entradas:

- CLK – Entrada de reloj del sumador. El flanco activo de la señal CLK es el flanco creciente
- ENB – Entrada de habilitación del sumador.
- MODO[1:0] – Entrada de modo que consta de dos líneas y sirve para definir cuál será el próximo estado del sumador al llegar el flanco activo del reloj en la entrada CLK
- A[3:0] – Entrada de datos A consta de 4 líneas.
- B[3:0] – Entrada de datos B consta de 4 líneas.

Para las salidas se tiene:

- Q[3:0] – Salida Q que consta de cuatro líneas que indican el estado presente del sumador.
- RCO – Salida de llevo “Ripple-Carry Out” que indica cuando el sumador llega a su cuenta límite.

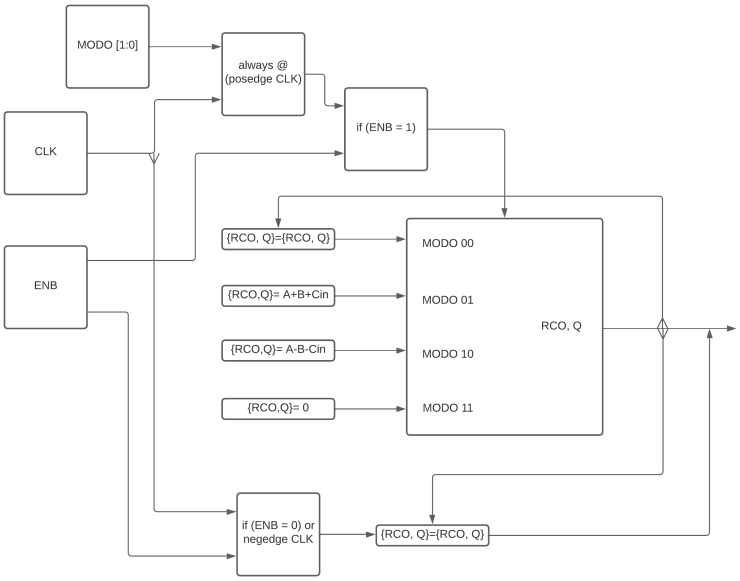

Figura 2: Interior módulo sumador.

Como se muestra en la Figura 2, se toman las consideraciones de funcionamiento del módulo sumador:

- La entrada MODO define si se realiza la suma, resta, mantener valor anterior o limpiar.
  - MODO = 00 → Q
  - MODO = 01 → A+B
  - MODO = 10 → A-B
  - MODO = 11 → 0
- Si ENB=1 el sumador funciona normalmente, si ENB=0 se mantiene el estado actual.
- Se sincroniza la selección de modo con CLK

Como se puede ver en la Figura 3, dentro de este módulo Sumador de 4 bits, donde la salida de estos se asigna a la Salida Q y RCO dependiendo de la señal de ENB, MODO y el flanco del reloj. Con respecto a esta parte, respecto a la tarea 1, se realizaron algunos cambios con el fin de simplificar el código y principalmente debido a que se tuvieron problemas al sintetizar con la herramienta yosys varios módulos.

Adicionalmente se utilizó un archivo tipo liberty para el proceso de síntesis, con el fin de especificar los componentes a utilizar, esto acompañado de un archivo llamado run.ys, esto para indicar el procedimiento paso a paso en yosys, esto dentro del entorno de EDAplayground.

```
module Sumador(A,B,Cin,CLK,ENB,MODO,Q,RCO);

  input [3:0] A,B;//Entradas de los nibbles a operar
  input Cin,CLK,ENB;//Entrada del acarreo inicial, el reloj y enable
  input [1:0] MODO;//Seleccion del modo de operacion
  output reg [3:0] Q;//Salida de la operacion
  output reg RCO;//Acarreo de salida


    //Se ejecuta siempre que se tenga un flanco positivo de reloj
    always @ (posedge CLK) begin
      //Se ejecuta siempre que se tenga ENB=1
      if (ENB==1'b1) begin
        //Se asigna a la salida el resultado segun la operacion seleccionada en MODO
        case (MODO)
          2'b00: begin
            Q = Q;
          end
          2'b01: begin
            {RCO,Q} = A + B + Cin;
          end
          2'b10: begin
            {RCO,Q} = A - B - Cin;
          end
          2'b11: begin
            Q = 0;
            RCO=0;
          end
        endcase
      end
      else begin
        Q = Q;
      end
    end
endmodule
```

Figura 3: Módulo Sumador de 4 bits.

3. Plan de Pruebas

Se realizaron las mismas pruebas que en la tarea 1. Adicionalmente se realizaron las siguientes pruebas al proceso de síntesis.

- Prueba 1, suma de 4 bits. El sumador inicia en cero y se definen valores para las entradas A y B. Se envía un flanco activo de reloj y el sumador realiza la operación correspondiente. Se debe repetir este proceso para varias posibles sumas, incluyendo casos que generen acarreo, para verificar el correcto funcionamiento de la señal RCO. Un ejemplo de cómo se puede estructurar la prueba se muestra en la siguiente secuencia:
  - Establecer MODO[1:0]=11 para limpiar el contador.
  - Poner ENB=1.
  - Enviar flanco activo en CLK. Con esto se pone el contador a cero.
  - Establecer MODO[1:0]=01. Pone modo de suma.
  - Enviar flanco activo en CLK. El estado de contador debería pasar a Q=A+B.
  - Variar los valores de A y B para observar distintos resultados, incluyendo casos con acarreo donde RCO=1.
- Prueba 2, resta de 4 bits. El sumador inicia en cero y se definen valores para las entradas A y B para ejecutar la resta correspondiente. También se debe verificar que la señal RCO se pone en 1 cuando se cumple la condición de rebase, ¿en qué estado debería ponerse RCO cuando A es menor que B?
- Prueba 3, mantener el valor en modo 00. El sumador realiza una operación inicial y se modifica el modo para ponerlo en 00. Luego se envían varios flancos positivos del reloj y se verifica que la salida se mantiene con el resultado de la última operación realizada.
- Prueba 4, mantener el valor cuando ENB = 0. El sumador realiza una operación inicial y después de completada se pone ENB=0. Luego se envían varios flancos positivos del reloj y se verifica que la salida se mantiene con el resultado de la última operación realizada.

- Prueba 5, limpiar el contador. Verificar que si el contador se encuentra en cualquier estado inicial posible, es posible regresar su salida a cero poniendo la señal de MODO=2'b11 y sosteniéndola durante un ciclo de reloj.

- Prueba 6: Síntesis de alto nivel del diseño conductual del sumador sincrónico de 4 bits. Esto produce una descripción estructural genérica (RTLIL) que no depende de una tecnología en particular. Los componentes usados en esta descripción corresponden a los de la biblioteca interna del sintetizador Yosys.

- Prueba 7: Síntesis con una biblioteca con una tecnología comercial disponible para realizar el mapeo tecnológico del diseño. Esto produce una descripción estructural usando componentes comercialmente disponibles

- Prueba 8: Cambiar en el testebench el archivo utilizado para correr la prueba, este caso sustituyendo el **design.sv** por **suamdor_synth.v** que corresponde al resultado del proceso de síntesis. Esto se comprueba por el análisis de los diagramas de tiempo. Esto con el uso de GTKwave.

- Prueba 9: Modificar los archivos de la biblioteca para que la simulación tome en cuenta retardos. Verifique que el diseño sintetizado con la librería modificada también funciona. Para esto se agregan retardos en la biblioteca de componentes (**cmos_cells.v** utilizada en el testbench. Se prueban tiempos de reloj muy bajos para observar el comportamiento.

- Prueba 10: Número de componentes usados: NAND, NOR, NOT y FFs

4. Instrucciones de utilización de la simulación

Esta tarea fue desarrollada en el entorno de EDAplayground por lo que basta con ingresar al siguiente link y correr la simulación.

- https://www.edaplayground.com/x/XJjT

5. Resultados

- **Prueba 1**: Se puede notar al inicio no se tienen cargados los valores de A y B, por esto se muestra x, después de realizar una limpieza (MODO=11) se ponen 0, sin embargo se muestra un error en el resultado al mostrar una x donde no debería, se ingresan A y B y se ejecuta una suma (MODO=01), la cual da el resultado correcto, luego se cambia el acarreo obteniendo el resultado esperado.:

```
Prueba #1, suma de 4 bits.
VCD info: dumpfile test.vcd opened for output.
Operando A: xxxx
Operando B: xxxx
Acarreo de entrada: x
ENB = 1
Resultado de la operacion 11: 00x0
Rebase: 0

Operando A: 1110
Operando B: 1001
Acarreo de entrada: 0
ENB = 1
Resultado de la operacion 01: 0111
Rebase: 1

Operando A: 0110
Operando B: 1001
Acarreo de entrada: 1
ENB = 1
Resultado de la operacion 01: 0000
Rebase: 1
```

Figura 4: Prueba 1. Resultados en terminal.

- **Prueba 2**: Se inicia con realizar una limpieza, se ponen 0 los resultados, se ingresan A y B y se ejecuta una resta, la cual da el resultado correcto.

```
Prueba #2, resta de 4 bits.
Operando A: 0110
Operando B: 1001
Acarreo de entrada: 1
ENB = 1
Resultado de la operacion 11: 0000
Rebase: 0

Operando A: 1110
Operando B: 1001
Acarreo de entrada: 0
ENB = 1
Resultado de la operacion 10: 0101
Rebase: 0
```

Figura 5: Prueba 2. Resultados en terminal.

- **Prueba 3**: Se inicia con realizar una suma, se cambia el modo a 00 (mantener el valor anterior), la cual da el resultado correcto.

```
Prueba #3, mantener el valor en modo 00.
Operando A: 1110
Operando B: 1001
Acarreo de entrada: 0
ENB = 1
Resultado de la operacion 01: 0111
Rebase: 1

Operando A: 1110
Operando B: 1001
Acarreo de entrada: 0
ENB = 1
Resultado de la operacion 00: 0111
Rebase: 1
```

Figura 6: Prueba 3. Resultados en terminal.

- **Prueba 4**: Se inicia con realizar una suma, se cambia el ENB a 0 (Se debe mantener el valor anterior), la cual da el resultado correcto.

```
Prueba #4, mantener el valor cuando ENB = 0.
Operando A: 1110
Operando B: 1001
Acarreo de entrada: 0
ENB = 1
Resultado de la operacion 01: 0111
Rebase: 1

Operando A: 1110
Operando B: 1001
Acarreo de entrada: 0
ENB = 0
Resultado de la operacion 01: 0111
Rebase: 1
```

Figura 7: Prueba 4. Resultados en terminal.

- **Prueba 6**: Se produce una descripción estructural genérica (RTLIL) que no depende de una tecnología en particular. Los componentes usados en esta descripción corresponden a los de la biblioteca interna del sintetizador Yosys.

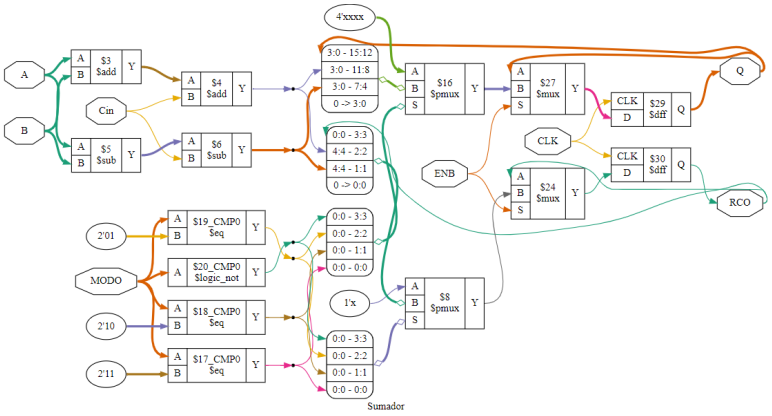

Figura 8: Prueba 6. Descripción estructural genérica.

- **Prueba 7**: Se cambia la biblioteca con una tecnología comercial disponible para realizar el mapeo tecnológico del diseño. En este caso se utilizan dos opciones distintas siendo la primera la librería ABC de yosys en EDAplayground, luego se realiza con el archivo **cmos_cells.lib**. En ambos se tuvieron digramas muy grandes, pero en la primera prueba se tuvieron mejores resultados, ya que en la segunda el diagrama se realiza en vertical lo cual es muy difícil de interpretar.

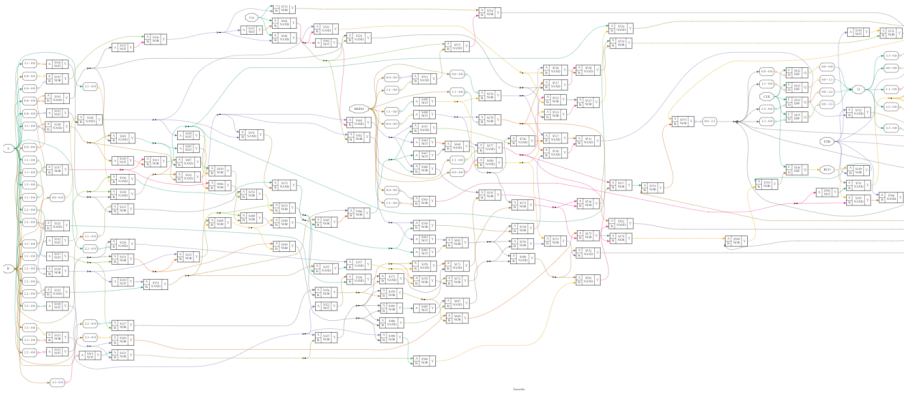

Figura 9: Prueba 7. Descripción estructural con librería ABC de yosys en EDAplayground.

Figura 10: Prueba 7. Descripción estructural con el archivo **cmos_cells.lib**.

- **Prueba 8**: Se utiliza el archivo generado en síntesis(**suamdor_synth.v**) para comparar los resultados con el de diseño(**design.sv**) a la hora de ejecutar el testbench. Como se puede ver se tienen resultados casi idénticos, a excepción por un valor desconocido en el inicio de Q[1], que corresponde al resultado de la Prueba 1, ya que en esta se tiene un bug que no se pudo corregir y se desconoce la diferencia entre el archivo de síntesis y el diseño que puede generar esto.

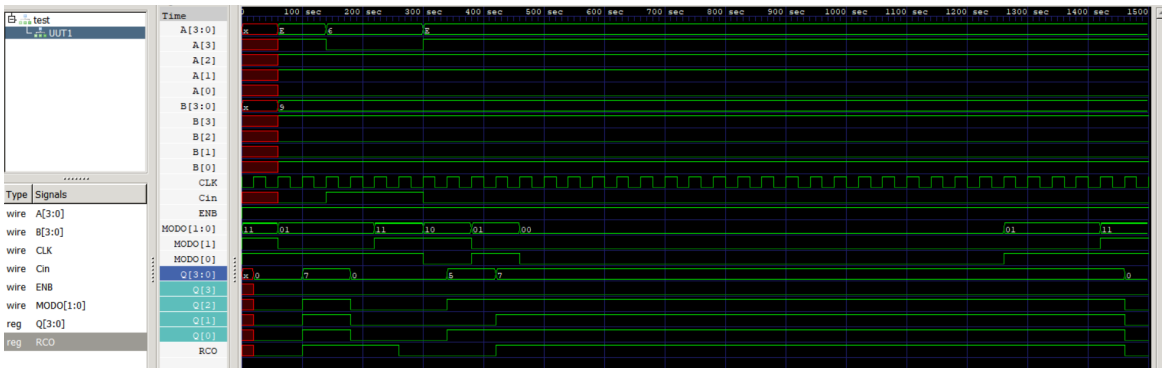

Figura 11: Prueba 8. Diagramas de tiempo utilizando **design.sv**.

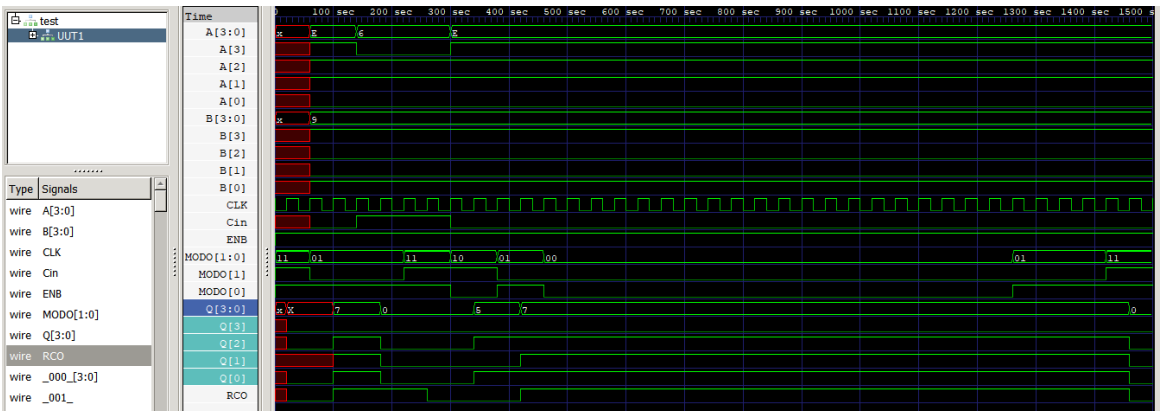

Figura 12: Prueba 8. Diagramas de tiempo utilizando **suamdor_synth.v**.

- **Prueba 9**: Se agregan retardos en la biblioteca de componentes (**cmos_cells.v** utilizada en el testbench. Primero se prueba con un tiempo de reloj alto y como se puede ver no se tiene direncia con los resultados de la prueba anterior, pero cuando de prueban tiempos de reloj muy bajos o menores a los tiempos de los componentes, se tienen muchos fallos y se pierden totalmente los resultados y se empieza a oscilar.

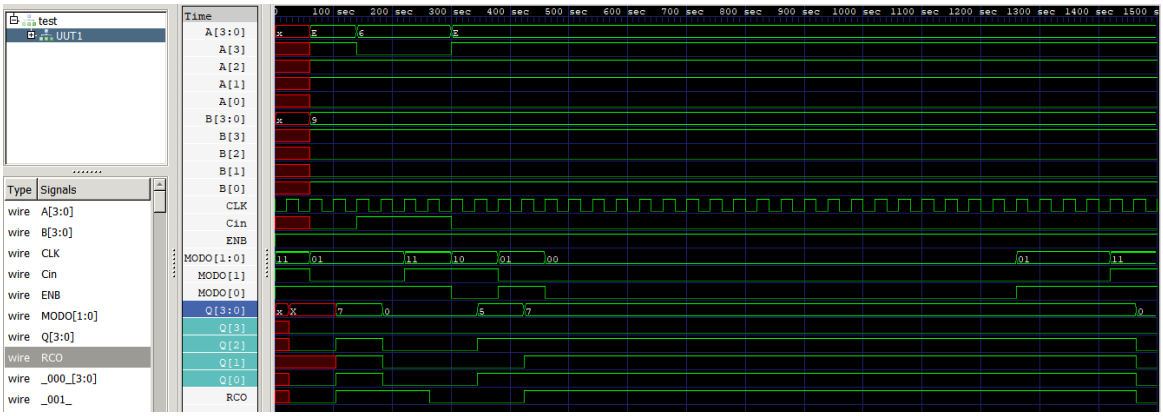

Figura 13: Prueba 9. Diagramas de tiempo utilizando **CLK** alto.

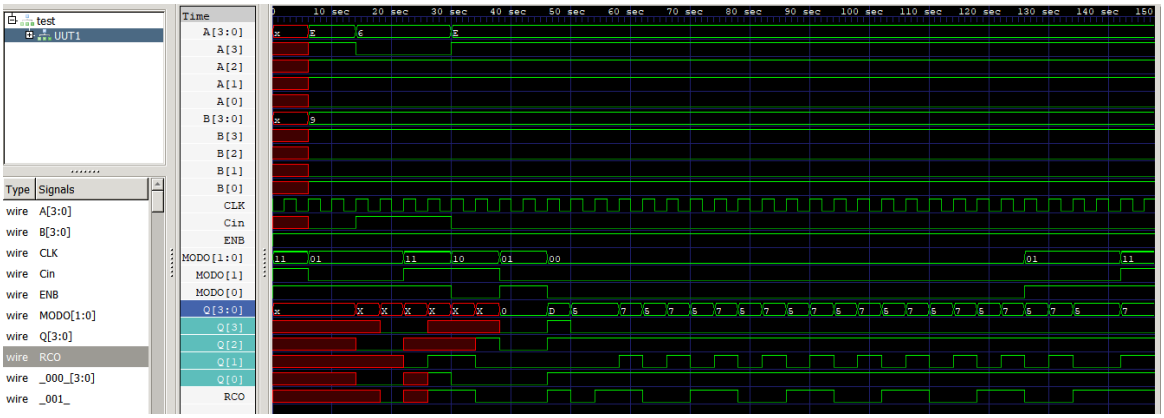

Figura 14: Prueba 9. Diagramas de tiempo utilizando **CLK** bajo.

- **Prueba 10**: Número de componentes usados: NAND, NOR, NOT y FFs.

```
12.1.2. Re-integrating ABC results.
ABC RESULTS:              NAND cells:       58
ABC RESULTS:               NOR cells:       63
ABC RESULTS:               NOT cells:       27
ABC RESULTS:        internal signals:      124
ABC RESULTS:           input signals:       17
ABC RESULTS:          output signals:        5
Removing temp directory.
```

Figura 15: Prueba 10. Cantidad de compuertas usadas.

```
Mapping DFF cells in module `\Sumador':
  mapped 5 $_DFF_P_ cells to \DFF cells.
```

Figura 16: Prueba 10. Cantidad de FFs usados.

6. Conclusiones

- Con respecto a las primeras pruebas de operaciones del sumador, se tuvieron buenos resultados, ya que se pudo tener prácticamente todos los resultados esperados, sin embargo en la primera prueba se tiene un fallo al realizar la limpieza, donde se limpian los resultados para poner el contador en cero, pero se ponen en cero 3 bit de los 4 bits, en el caso del bit Q[1] se desconoce la causa de este problema al utilizar el achico generado en síntesis, ya que se deberían tener los mismos resultados que con el archivo de diseño, pero también es curioso que esto pasa solo en esta prueba, ya que en otras pruebas se realiza la limpieza y esta ejecuta de manera correcta.

- Con la síntesis utilizando componentes comerciales, se puede ver que se vuelve muy grande el diagrama, esto se puede deber a que dentro de las librerías utilizadas no se contemplan el uso de MUX, como si se utiliza cuando de realiza el proceso de manera genérica.

- Al realizar la comparación entre realizar el testbench con el archivo resultado de síntesis o el de diseño, se puede ver el que se tiene el mismo problema que en la prueba 1, se refleja en los diagramas de tiempo el fallo al realizar esa primera limpieza en el bit Q[1].

- En la prueba 9 se puede ver la importancia de un tiempo de reloj correcto, ya que si este es muy bajo no se le da tiempo a los componentes de realizar las operaciones y se pierden los resultados, incluso se empieza a oscilar.

Referencias

[1] J. A. G. Garza, *Sistemas digitales y electrónica digital, prácticas de laboratorio.* Pearson Educación, 2006.