Universitat Politècnica de Catalunya

Arquitectura de computadors d'altes prestacions

Práctica 2

Sumador de 4 bits y Sumador BCD

Carlota Catot
Miguel Antunez

Grupo 7

Quatrimestre primavera 2020-2021

Índice

1. Pregunta 1

Suponga que antes del instante de tiempo t0 todas las señales del sumador son estables y que en este instante solo cambia una las 9 señales de entrada. Suponga también que la última señal de salida que se estabiliza es s2 en el instante t1. Indique el retardo del circuito (caso peor), los valores de las entradas antes del instante t0 y en el instante t0. Modifique el programa de prueba (prueba_S4bits.vhd) añadiendo un proceso que genere las señales de entrada correspondientes. Considere t0=200 ns. Compruebe el resultado con el simulador. Tenga en cuenta que los parámetros asociados a los retardos de las puertas se establecen al instanciar el componente S4bits en el programa de prueba.

1.1. Resolución

Para resolver esta pregunta, hemos modificado el código del sumador4 bits con tal de poder generar una señal y poder ver el retardo máximo de esta y de esta manera poder llenar los datos de la tabla siguiente indicando el peor de los escenarios en cuanto a retardo del circuito.

Entradas (t >t0)			Entradas (t = t0)			retardo (t1 - t0)
A	B	Cen	A	B	Cen	Ret1 (ns)
0011	0000	0	0011	0001	0	65 ns

Para poder observar esto se ha usado el código que se expone a continuación, donde se da como estímulos los valores indicados en la tabla para que se generen correctamente las ondas, una parte de dichas ondas se puede ver en la figura 1 donde se observa que hay un retardo de 65 ns con los dos cursores colocados en t0 y t1 siendo t0 200 ns y t1 265 ns.

Para poder tener el retardo correcto se han tenido que actualizar los retardos en el código para especificar que todas las puertas XOR el retardo sea de 15 ns así como también para las puertas OR, mientras que las puertas AND tienen un retardo de 10 ns.

Por último queremos destacar que este es el caso peor porque al cambiar precisamente ese bit estas provocando dos acarreos y generando así el caso peor con un retardo de **65 ns**.

1.2. Código

```
library ieee;
use ieee.std_logic_1164.all;
use work.all;

entity prueba_S4bits is

end prueba_S4bits;

architecture prueba of prueba_S4bits is

component S4bits is
generic(ret_xor: time := 15 ns; ret_and: time := 10 ns; ret_or: time := 15 ns);
port (A: in std_logic_vector(3 downto 0);
               B: in std_logic_vector(3 downto 0);
```

```
                cen: in         std_logic;
                SUM: out std_logic_vector(3 downto 0);
                csal: out std_logic);
end component;

-- senyales
signal A,B: std_logic_vector(3 downto 0);
signal cen: std_logic;
signal SUM: std_logic_vector(3 downto 0);
signal csal: std_logic;
signal c1, c2, c3, c4: std_logic;

-- instanciacion y estimulos
begin

S4bits0: S4bits port map(A=>A,B=>B,cen=>cen,csal=>csal,SUM=>SUM);

A <= "0011";
B <= "0000","0001" after 200ns, "UUUU" after 400ns;
cen <= '0';

end prueba;
```

1.3. Ventana Temporal

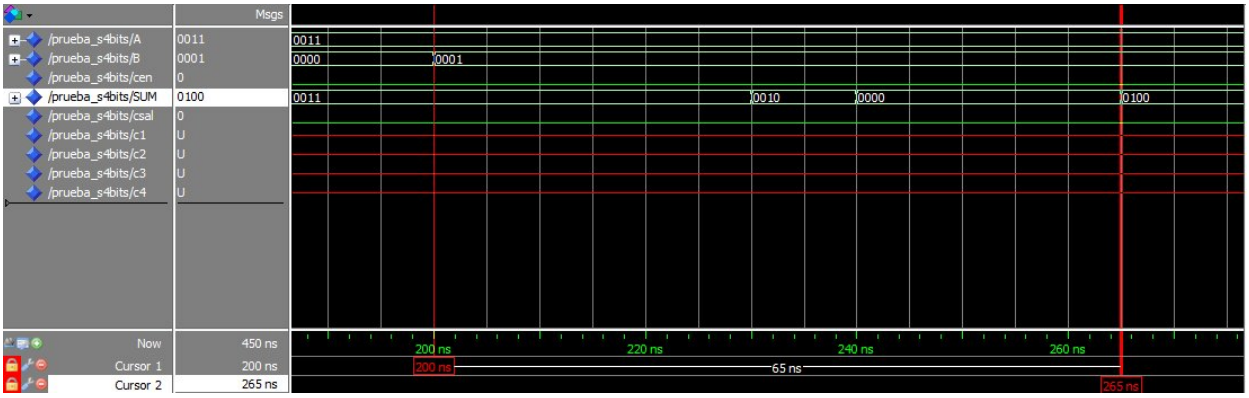

Figura 1: Ventana temporal ejecución prueba_S4bits.vhd

2. Pregunta 2

Considere el sumador de 4 bits especificado en VHDL mediante sentencias generate. Tenga en cuenta que, en este diseño, los parámetros asociados a los retardos de las puertas se establecen al instanciar los componentes s1bits en el fichero snbits.vhd. Modifique el programa de pruebas (prueba_snbits_reloj.vhd) para que imprima el primer valor de las entradas en el que se observa el retardo máximo y el mínimo.

Para contestar correctamente a esta pregunta nos centraremos en estas dos tablas, la primera corresponde al retardo mínimo del código y la segunda al retardo máximo de este.

2.1. Retardo mínimo

vector de bits		retardo mínimo
A	0000	40 ns
B	0001	
Cen	1	

Hemos detectado que el retardo mínimo es de **40ns** esto se puede explicar de una manera muy simple con el carry, como las 3 posiciones más altas son 0, no existe dependencia así que el retardo mínimo sera la suma de las puertas lógicas AND (10 ns), OR (15ns) i XOR(15ns).

2.2. Retardo máximo

vector de bits		retardo máximo
A	0000	100 ns
B	1110	
Cen	0	

Hemos detectado que el retardo máximo es de **100ns** que se produce al sumar el retardo de la conexión en serie de las puertas Cxi (25 + 25 + 25 + 25). Esto se puede ver de una manera muy clara en el dibujo de la figura 2 que tenemos a continuacion

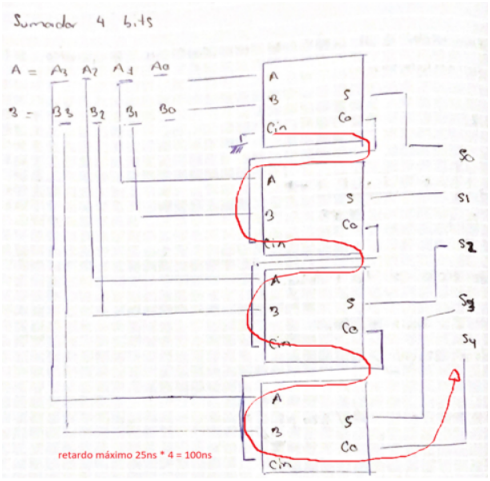

Figura 2: Esquema explicación del retardo máximo

2.3. Codigo

El fragmento de código empleado dentro del fichero prueba_snbits_reloj.vhd es el siguiente:

```
-- resto de variables
variable t_retardo_min: time;
variable t_retardo_max: time;
variable a_min: integer;
variable b_min: integer;
variable a_max: integer;
variable b_max: integer;

begin
    t_retardo_min := 9999 ns;
    t_retardo_max := 0 ns;
    a_min := 0;
    a_max := 0;
    b_min := 0;
    b_max := 0;


    for aca in 0 to 1 loop
            for i in 0 to 5 loop
            for j in 0 to 5 loop
                A <= std_logic_vector(to_unsigned(i,tam));
                B <= std_logic_vector(to_unsigned(j,tam));
                if aca = 0 then
                    cen <= '0';
                else
                    cen <= '1';
                end if;
                wait until reloj = '0' and reloj'event;
                t_retardo := periododiv2 - csal_SUM'last_event;

-- introduzca codigo para actualizar retardos maximo y minimo
                if t_retardo < t_retardo_min and t_retardo > 0 ns then
                    t_retardo_min := t_retardo;
                    a_min := i;
                    b_min := j;
                end if;
                if t_retardo > t_retardo_max then
                    t_retardo_max := t_retardo;
                    a_max := i;
                    b_max := j;
                    end if;

                assert ((csal_SUM) = (csalref & SUMref))
                report "la comprobacion falla" severity error;

                if(to_integer(unsigned(csal_SUM)) /= (i+j+aca)) then
                    errores <= errores + 1;
                end if;

-- preparacion del circuito para medir retardos
                A <= (others => 'U');
                B <= (others => 'U');
```

```
                cen <= 'U';
                wait until reloj = '1' and reloj'event;
            end loop;
        end loop;
    end loop;

    report "numero de errores: " & integer'image(errores);
    report "------ MINIMO ------";
    report "retardo_min: " & to_string(t_retardo_min, ns);
    report "a_min: " & integer'image(a_min);
    report "b_min: " & integer'image(b_min);
    report "------ MAXIMO ------";
    report "retardo_max: " & to_string(t_retardo_max, ns);
    report "a_max: " & integer'image(a_max);
    report "b_max: " & integer'image(b_max);

    final <= '1';
    wait;

end process;
```

2.4. Resultados

```
# ** Note: ------ MINIMO ------
# ** Note: retardo_min: 40 ns
# ** Note: a_min: 0
# ** Note: b_min: 1

# ** Note: ------ MAXIMO ------
# ** Note: retardo_max: 100 ns
# ** Note: a_max: 0
# ** Note: b_max: 14
```

3. Pregunta 3

Considere el circuito para sumar 2 dígitos BCD. Deduzca una expresión lógica para calcular el acarreo de salida del módulo mayor9.

3.1. Expresión lógica

Para sumar en BCD se debe codificar en grupo de 4 bits, cuando la suma es mayor que 9 se debe sumar 6 al resultado de la suma y arrastrar el acarreo generado, un ejemplo de esto lo podemos ver en la figura 3.

Pero en BCD separamos en grupo de 4 bits y cuando la suma sea mayor a 9 se **suma 6 (0110)**. Esto lo podemos ver en la figura 4

$$+\begin{array}{r}1349\\3822\end{array} = 5171$$

Figura 3: Ejemplo suma BCD decimal

$$+\begin{array}{r}0001\ 0011\ 0100\ 1001\\0011\ 1000\ 0010\ 0010\\\hline 0101\ 1011\ 0111\ 1011\\0000\ 0110\ 0000\ 0110\\\hline 0101\ 0001\ 0111\ 0001\end{array} = 5171$$

Figura 4: Ejemplo suma BCD binario

Sabiendo todo lo comentado anteriormente, se puede deducir que la expresión lógica para efectuar este acarreo es la siguiente:

```
csal <= '1' when X > "01001" else '0' after retmayor9
```

3.2. Arquitectura módulo

Muestre el cuerpo de la arquitecura del módulo sumador de 2 dígitos BCD.

Para poder ejecutar sumas con dos dígitos hemos usado la siguiente arquitectura dentro del fichero s1bcd.vhl.

```
architecture compor of s1bcd is
signal SA: st_bcd;
signal csalA: std_logic;
signal X9: st_bcd_mas_1;
signal S9: st_bcd;

begin
    sumA: snbits port map ( X => X, Y => Y, cen => cen, csal => csalA, sum => SA);
    X9 <= csalA & SA;
    m9: mayor9 port map ( X => X9, S => S9, csal => csal);
    sumB: snbits port map ( X => SA , Y => S9, cen => '0', sum => S );
end;
```

4. Pregunta 4

Considere la suma algebraica de números enteros BCD. Sean x = veinticinco e y = menos ocho. Deduzca los valores explícitos y los vectores de bits (9 bits).

4.1. Valores explícitos

	valor explícito	vector de bits
x	xe = 25 mod 200 = 25	X = (0,0,0,1,0,0,1,0,1)
y	ye = -8 mod 200 = 192	Y = (1,1,0,0,1,0,0,1,0)

Para llegar a esta conclusión,

4.2. Valor explícito de la resta

Calcule el valor explícito de la resta s = x - y. Indique el vector de bits del resultado y si es representable.

		vector de bits
xe	25 mod 200 = 25	X = (0,0,1,0,0,1,0,1)
ye	8 mod 200 = 8	Y = (0,0,0,0,1,0,0,0)
se	25 + 8 = 33 en base 10	S = (0,0,0,1,1,0,0,1,1)
representable	Es representable	

5. Pregunta 5

Considere el sumador de enteros BCD. Deduzca una expresión para evaluar el retardo de un sumador de enteros representados con 1 bit y n dígitos BCD. Utilice la nomenclatura de los retardos indicados en el package retardos_bcd_pkg.

En la figura 5 podemos ver un esquema de como es un circuito para la suma de n dígitos BCD, estos están puestos en serie y puede haber n sumadores con tal de poder sumar todos los dígitos.

Es por eso que presentamos la siguiente expresión para evaluar los retardos, donde dependiendo del número de dígitos las operaciones dentro del paréntesis siempre se ejecutará para todos los sumadores.

```
(Suma A + comprobación Mayor 9 + suma B) * NumDigitos  + S1bit =
    (retsumbin + retmayor9 + retsumbin)*NumDigitos + rets1bit
```

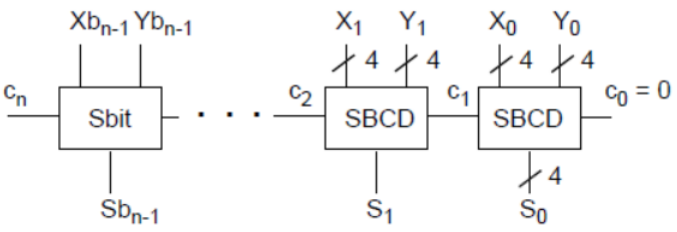

Figura 5: Esquema de un circuito para la suma de n digitos BCD

6. Pregunta 6

Considere la suma algebraica de números enteros. Deduzca las expresiones lógicas de las señales de salida del módulo compl9.

Para deducir las expresiones lógicas de las señales del módulo compl9 calculamos el complemento a 9 de todas las posibles entradas (0-15). Una vez tenemos la tabla de salida utilizamos el método de Karnaugh para obtener las expresiones lógicas utilizando la siguiente web `http://www.32x8.com/var4.html`.

6.1. Z(3)

```
z(3) = NOT X0
```

6.2. Z(2)

```
z(2) = X1
```

6.3. Z(1)

```
z(1) = X2 XOR X1
```

6.4. Z(0)

```
z(0) = NOT X3 AND NOT X2 AND NOT X1
```