Rubén Comerón Galán

Práctica 5-Sumador CSLA

El modelo VHDL del bloque mencionado en el enunciado de la sesión es:

```
ENTITY CSL IS
	GENERIC (k: INTEGER:=4);
	PORT(x_in, y_in: IN BIT_VECTOR(k-1 downto 0); c_in: IN BIT; sal: OUT
BIT_VECTOR(k-1 downto 0); c_out: OUT BIT);
END ENTITY;

ARCHITECTURE estructural OF CSL IS

	COMPONENT RCA
		GENERIC (c: INTEGER:=4);
		PORT(x_in, y_in: IN BIT_VECTOR(c-1 downto 0); c_in: IN BIT; sal: OUT
BIT_VECTOR(c-1 downto 0); c_out: OUT BIT);
	END COMPONENT;
	COMPONENT mux21
		PORT (e1, e2, control: IN BIT; s: OUT BIT);
	END COMPONENT;
	COMPONENT and2
		GENERIC (retardo: TIME:= 1 ns);
		PORT (e1, e2: IN BIT; sal: OUT BIT);
	END COMPONENT;
	COMPONENT or2
		GENERIC (retardo: TIME:= 1 ns);
		PORT (e1, e2: IN BIT; sal: OUT BIT);
	END COMPONENT;
SIGNAL RCA_0, RCA_1: BIT_VECTOR(k-1 DOWNTO 0);
SIGNAL RCA_0_Cout, RCA_1_Cout, Cin_and_Cout: BIT;
BEGIN

	RCA_carry_0: RCA PORT MAP(x_in, y_in, '0', RCA_0, RCA_0_Cout);
	RCA_carry_1: RCA PORT MAP(x_in, y_in, '1', RCA_1, RCA_1_Cout);
	G: FOR i IN 0 TO k-1 GENERATE
		multiplexor: mux21 PORT MAP(RCA_0(i), RCA_1(i), c_in, sal(i));
	END GENERATE G;
	Cout_and: and2 PORT MAP (c_in, RCA_1_Cout, Cin_and_Cout);
	Cout_or: or2 PORT MAP (RCA_0_Cout, Cin_and_Cout, c_out);

END estructural;
```

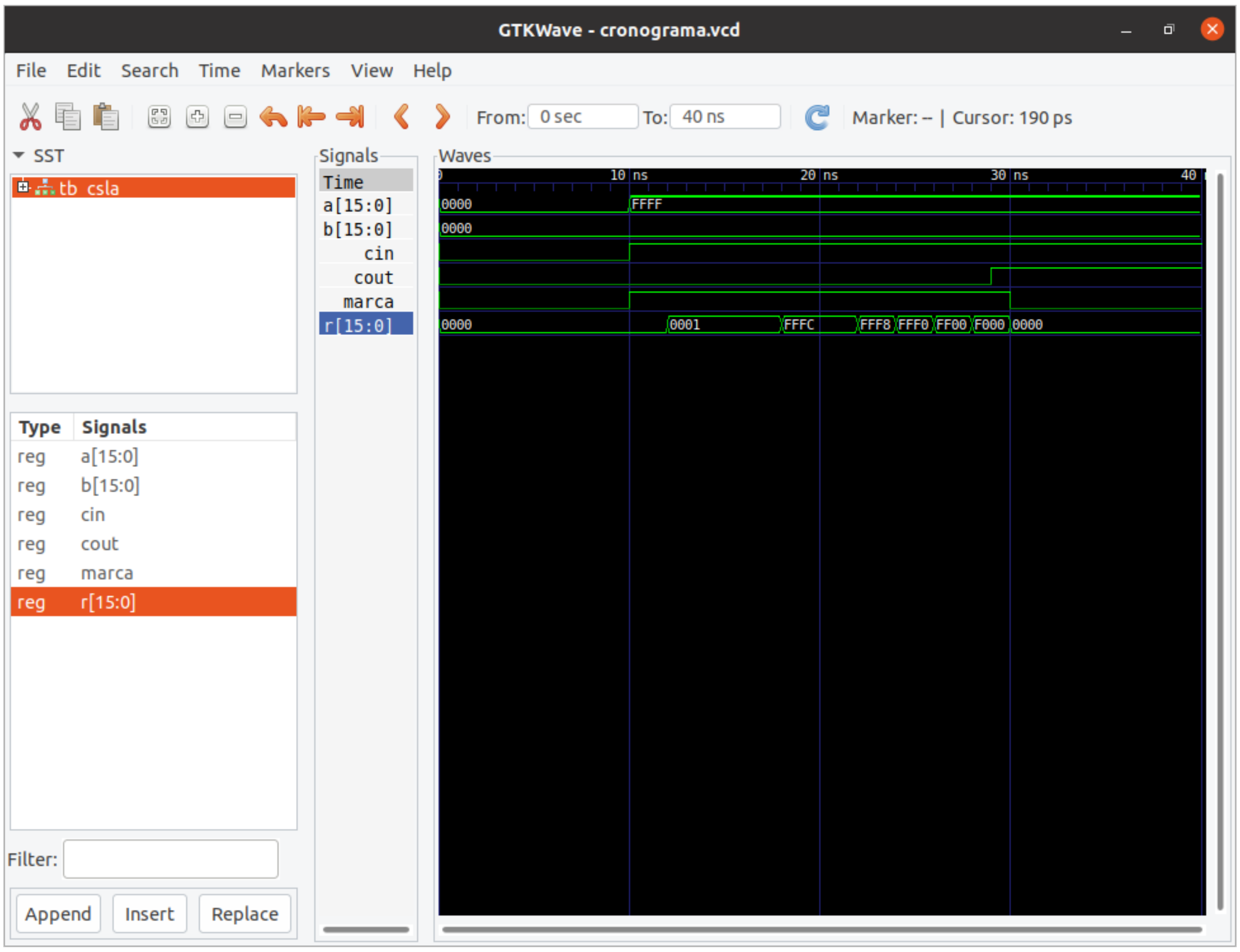
El modelo VHDL del selector de acarreo (CSLA) es:

```
ENTITY CSLA IS
	GENERIC (n: INTEGER:=16);
	PORT(x_in, y_in: IN BIT_VECTOR(n-1 downto 0); c_in: IN BIT; sal: OUT BIT_VECTOR(n-1 downto 0); c_out: OUT BIT);
END ENTITY;

ARCHITECTURE estructural OF CSLA IS

COMPONENT CSL
	GENERIC (k: INTEGER:=4);
	PORT(x_in, y_in: IN BIT_VECTOR(k-1 downto 0); c_in: IN BIT; sal: OUT BIT_VECTOR(k-1 downto 0); c_out: OUT BIT);
END COMPONENT;
SIGNAL Cout_aux: BIT_VECTOR(3 DOWNTO 0);
CONSTANT k: INTEGER:=4;
BEGIN

	G: FOR i IN 0 TO (n/4)-1 GENERATE
	lsb: IF i = 0 GENERATE
			primero: CSL PORT MAP(x_in((i+1)*k-1 DOWNTO i*k), y_in((i+1)*k-1 DOWNTO i*k), c_in, sal((i+1)*k-1 DOWNTO i*k), Cout_aux(i));
		END GENERATE lsb;

		siguientes: IF i > 0 AND i < (n/4)-1 GENERATE
			intermedios: CSL PORT MAP(x_in((i+1)*k-1 DOWNTO i*k), y_in((i+1)*k-1 DOWNTO i*k), Cout_aux(i-1), sal((i+1)*k-1 DOWNTO i*k), Cout_aux(i));
		END GENERATE siguientes;

		msb: IF i = (n/4)-1 GENERATE
			ultimos: CSL PORT MAP(x_in((i+1)*k-1 DOWNTO (i*k)), y_in((i+1)*k-1 DOWNTO i*k), Cout_aux(i-1), sal((i+1)*k-1 DOWNTO i*k), c_out);
		END GENERATE msb;
	END GENERATE G;

END estructural;
```

El test-bench utilizado es:

```
ENTITY tb_CSLA IS
END tb_CSLA;

ARCHITECTURE estructural OF tb_CSLA IS
```

```
        COMPONENT CSLA
                GENERIC (n: INTEGER:=4; k: INTEGER:=2);
                PORT(x_in, y_in: IN BIT_VECTOR(n-1 downto 0);
                    c_in: IN BIT;
                    sal: OUT BIT_VECTOR(n-1 downto 0);
                    c_out: OUT BIT);
        END COMPONENT;

        FOR componente: CSLA USE ENTITY WORK.CSLA(estructural);

        CONSTANT TAMANO_PALABRA: INTEGER := 4;
        CONSTANT TAMANO_PALABRA_RCA: INTEGER := 2;

        SIGNAL A, B, R: BIT_VECTOR(TAMANO_PALABRA-1 downto 0);
        SIGNAL cin, cout, marca: BIT;

BEGIN

        componente: CSLA GENERIC MAP (TAMANO_PALABRA) PORT MAP (A, B, Cin, R,
Cout);

        PROCESS
        VARIABLE valor, limite, cuenta_carry, sol: INTEGER;
        BEGIN

                limite:= 2**TAMANO_PALABRA-1;

                FOR i IN 0 TO limite LOOP
                        valor:=i;

                        FOR k IN B'REVERSE_RANGE LOOP
                                A(k)<=BIT'VAL(valor REM 2);--cambiamos A
                                valor:=valor/2;
                        END LOOP;

                        FOR j IN 0 TO limite LOOP
                                valor:=j;

                                FOR k IN B'REVERSE_RANGE LOOP
                                        B(k)<=BIT'VAL(valor REM 2);--cambiamos B
                                        valor:=valor/2;
                                END LOOP;
                                FOR carry IN 0 TO 1 LOOP
                                        cin<=BIT'VAL(carry REM 2);
                                        WAIT FOR 20 ns;
                                END LOOP;
                        END LOOP;
                END LOOP;
        WAIT;
        END PROCESS;
END estructural;
```


GTKWave - cronograma.vcd
File Edit Search Time Markers View Help
From: 0 sec To: 40 ns
Marker: -- | Cursor: 190 ps
SST
tb_csla
Signals
Waves
Time
a[15:0]
b[15:0]
cin
cout
marca
r[15:0]
10 ns
20 ns
30 ns
40
0000
FFFF
0000
0000
0001
FFFC
FFF8
FFF0
FF00
F000
0000
Type Signals
reg a[15:0]
reg b[15:0]
reg cin
reg cout
reg marca
reg r[15:0]
Filter:
Append
Insert
Replace