INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE COMPUTO

Mult4bit

Practica 9

Materia:

Arquitectura de computadoras

Profesor:

Castillo Cabrera Gelacio

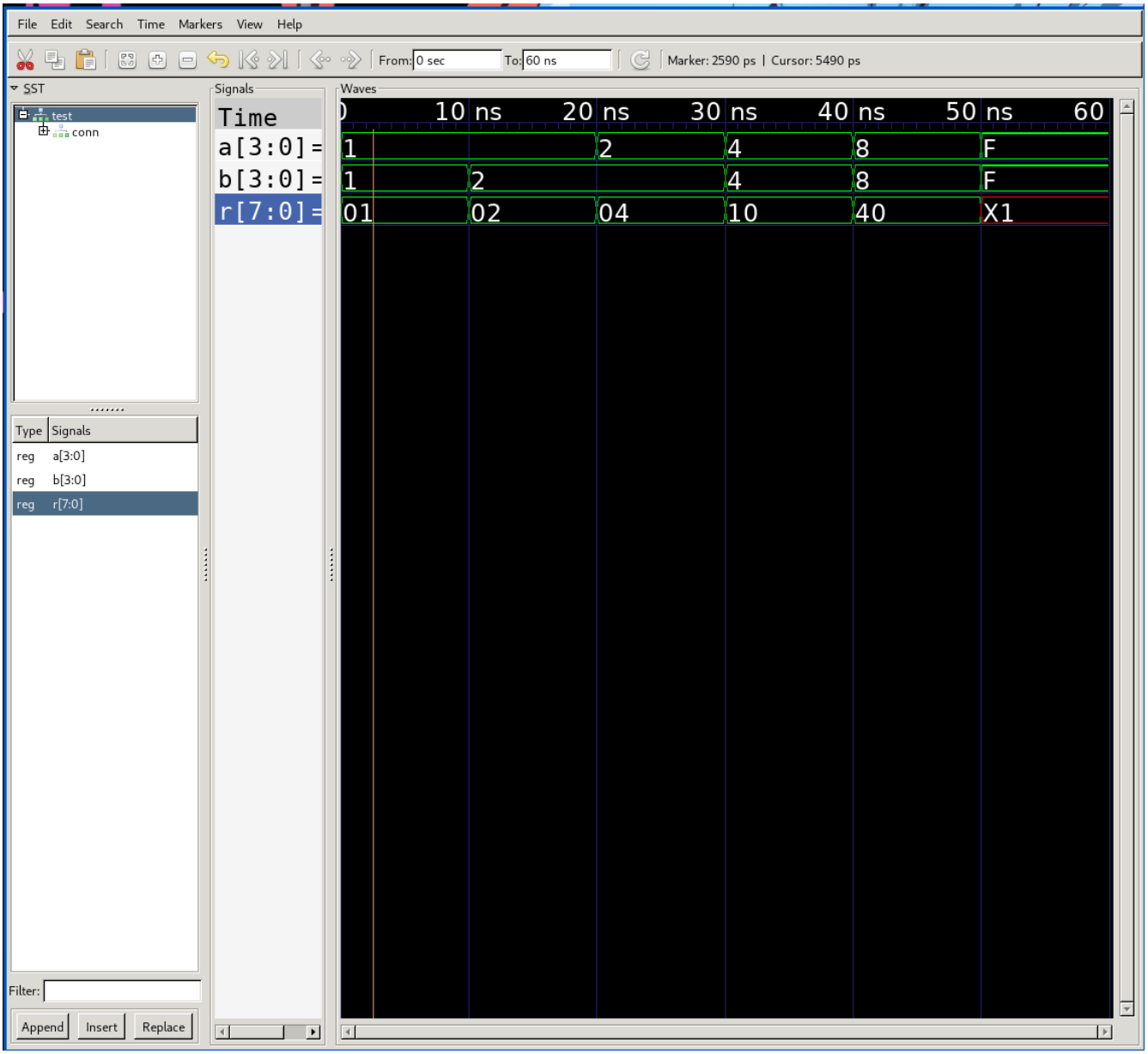
Alumno:

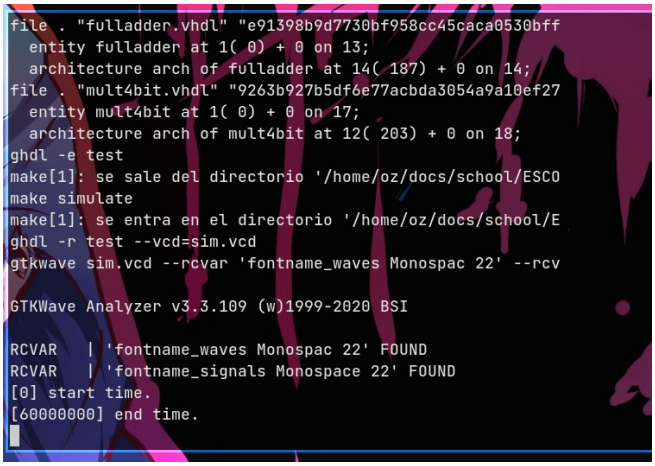
Cortés Piña Oziel

Grupo:

3CM12

Simulación en GHDL y GTLWAVE





Código VHDL

Halfadder

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

entity halfadder is
      port(
            a:    in    std_logic;
            b:    in    std_logic;
            s:    out   std_logic;
            c:    out   std_logic
      );
end halfadder;

architecture arch of halfadder is
begin
      s <= a xor b;
      c <= a and b;
end arch;
```

Fulladder

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

entity fulladder is
      port(
            a:    in    std_logic;
            b:    in    std_logic;
            e:    in    std_logic;
            s:    out   std_logic;
            c:    out   std_logic
      );
end fulladder;

architecture arch of fulladder is
      component halfadder is
            port(
                  a:    in    std_logic;
                  b:    in    std_logic;
                  s:    out   std_logic;
                  c:    out   std_logic
            );
      end component;
      signal sum, carryA, carryB:     std_logic;
begin
      first: halfadder    port map( a => a, b => b, s => sum,    c => carryA );
      second:      halfadder    port map( a => e, b => sum,     s => s,            c =>
carryB );
      c <= carryA or carryB;
end arch;
```

Adder8bit

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

entity adder8bit is
	port(
		a:	in	std_logic_vector(7 downto 0);
		b:	in	std_logic_vector(7 downto 0);
		e:	in	std_logic;
		s:	out	std_logic_vector(7 downto 0);
		c:	out	std_logic
	);
end adder8bit;

architecture arch of adder8bit is
	component fulladder  is
		port(
			a:	in	std_logic;
			b:	in	std_logic;
			e:	in	std_logic;
			s:	out	std_logic;
			c:	out	std_logic
		);
	end component;
	signal carry: std_logic_vector(6 downto 0);
begin
	s0:	fulladder port map( a => a(0), b => b(0), e => e,           s => s(0), c => carry(0) );
	s1:	fulladder port map( a => a(1), b => b(1), e => carry(0), s => s(1), c => carry(1) );
	s2:	fulladder port map( a => a(2), b => b(2), e => carry(1), s => s(2), c => carry(2) );
	s3:	fulladder port map( a => a(3), b => b(3), e => carry(2), s => s(3), c => carry(3) );
	s4:	fulladder port map( a => a(4), b => b(4), e => carry(3), s => s(4), c => carry(4) );
	s5:	fulladder port map( a => a(5), b => b(5), e => carry(4), s => s(5), c => carry(5) );
	s6:	fulladder port map( a => a(6), b => b(6), e => carry(5), s => s(6), c => carry(6) );
	s7:	fulladder port map( a => a(7), b => b(7), e => carry(6), s => s(7), c => c );
end arch;
```

Mult4bit

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

entity mult4bit is
	port(
		a:	in	std_logic_vector(3 downto 0);
		b:	in	std_logic_vector(3 downto 0);
		r:	out	std_logic_vector(7 downto 0)
	);
end mult4bit;

architecture arch of mult4bit is
	component adder8bit is
		port(
			a:	in	std_logic_vector(7 downto 0);
			b:	in	std_logic_vector(7 downto 0);
			e:	in	std_logic;
			s:	out	std_logic_vector(7 downto 0);
			c:	out	std_logic
		);
	end component;
	signal p0, p1, p2, p3:std_logic_vector(7 downto 0);
	signal s0, s1:		std_logic_vector(7 downto 0);
	signal carry:		std_logic_vector(1 downto 0);
begin

	p0(0) <= b(0) and a(0);
	p0(1) <= b(1) and a(0);
	p0(2) <= b(2) and a(0);
	p0(3) <= b(3) and a(0);
	p0(4) <= '0';
```

```
	p0(5) <= '0';
	p0(6) <= '0';
	p0(7) <= '0';

	p1(0) <= '0';
	p1(1) <= b(0) and a(1);
	p1(2) <= b(1) and a(1);
	p1(3) <= b(2) and a(1);
	p1(4) <= b(3) and a(1);
	p1(5) <= '0';
	p1(6) <= '0';
	p1(7) <= '0';

	p2(0) <= '0';
	p2(1) <= '0';
	p2(2) <= b(0) and a(2);
	p2(3) <= b(1) and a(2);
	p2(4) <= b(2) and a(2);
	p2(5) <= b(3) and a(2);
	p2(6) <= '0';
	p2(7) <= '0';

	p3(0) <= '0';
	p3(1) <= '0';
	p3(2) <= '0';
	p3(3) <= b(0) and a(3);
	p3(4) <= b(1) and a(3);
	p3(5) <= b(2) and a(3);
	p3(6) <= b(3) and a(3);
	p3(7) <= '0';

	r0: adder8bit port map( a => p0,     b => p1,       e => '0', s => s0, c => carry(0) );
	r1: adder8bit port map( a => s0,     b => p2,       e => carry(0), s => s1, c => carry(1) );
	s2: adder8bit port map( a => s1,     b => p3,       e => carry(1), s => r, c => r(7) );
end arch;
```

Test

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

entity test is
end test;

architecture arch of test is
	component mult4bit is
		port(
			a:	in	std_logic_vector(3 downto 0);
			b:	in	std_logic_vector(3 downto 0);
			r:	out	std_logic_vector(7 downto 0)
		);
	end component;
	signal a: std_logic_vector(3 downto 0);
	signal b: std_logic_vector(3 downto 0);
	signal r: std_logic_vector(7 downto 0);
begin
	conn: mult4bit port map( a => a, b => b, r => r );
	process begin
		a <= "0001";
		b <= "0001";
		wait for 10 ns;
		a <= "0001";
		b <= "0010";
		wait for 10 ns;
		a <= "0010";
```

```
            b <= "0010";
            wait for 10 ns;
            a <= "0100";
            b <= "0100";
            wait for 10 ns;
            a <= "1000";
            b <= "1000";
            wait for 10 ns;
            a <= "1111";
            b <= "1111";
            wait for 10 ns;
            wait;
      end process;
end arch;
```

Análisis de vectores

Vectores de entrada

A		B	
Binario	Hexadecimal	Binario	Hexadecimal
0001	1	0001	1
0001	1	0010	2
0010	2	0010	2
0100	4	0100	4
1000	8	1000	8
1111	F	1111	F

Valores de salida

Binario	Hexadecimal
00000001	01
00000010	02
00000100	04
00010000	10
00100000	40
x1100001	X1

Conclusión

A pesar de que de cierta manera el diseño del circuito es elegante el inocente tamaño lo vuelve complejo por que usar circuitos auxiliares ayudan pero al usar un sumador positivo corrompe los resultados a escalas grandes por lo que no es conveniente este diseño en el cual me apoyo de circuitos anteriores aunque más rápido de elaborar.