Preguntas del Lab

jueves, 7 de noviembre de 2024 21:28

1) Identifique qué elementos constituyen los LEs de la FPGA Cyclone III y qué estructura tienen las LABs

Los Logic Elements de la FPGA Cyclone III están constituidos por:

- Una tabla de búsqueda (LUT) de cuatro entradas, que puede implementar cualquier función de cuatro variables.
- Un registro programable.
- Una conexión de cadena de acarreo.
- Una conexión de cadena de registros.
- La capacidad de manejar las siguientes interconexiones:
  - Local
  - Fila
  - Columna
  - Cadena de registros
  - Enlace directo
- Soporte para empaquetado de registros.
- Soporte para retroalimentación de registros.

La placa que se utilizó para la realización del Laboratorio (EP3C120F780C7) contiene 119,088 LEs.

Los Logic Array Blocks (LABs) contienen grupos de LEs, para la Cyclone III cada uno contiene:

- 16 LEs.
- Señales de control de LAB.
- Cadenas de acarreo.
- Cadenas de registros.
- Interconexiones locales, que transfieren señales entre los LEs dentro del mismo LAB, optimizando el rendimiento y el uso del área.

2) ¿De qué se trata el Nios® II?

El Nios II es una arquitectura de procesador embebido de 32 bits, diseñada específicamente para las FPGAs de Altera (ahora Intel). Es una versión mejorada del procesador Nios original, optimizada para una amplia gama de aplicaciones en sistemas embebidos, desde procesamiento digital de señales hasta control de sistemas.

3) ¿Qué diferencia existe entre IP cores y los bloques embebidos (ej multiplicador embebido) disponibles en la FPGA?

Los *IP cores* son módulos lógicos programables que se implementan en la FPGA mediante lógica configurable y pueden personalizarse para diversas funciones específicas. En cambio, los *bloques embebidos* son recursos de hardware dedicados, como multiplicadores y bloques DSP, que están físicamente integrados en la FPGA. Estos bloques embebidos no consumen lógica programable, ofrecen alto rendimiento para tareas específicas y están optimizados para eficiencia en área y

energía.

4) ¿Qué tipo de celda de programación posee el dispositivo FPGA Cyclone III?

La FPGA Cyclone III utiliza celdas de programación basadas en tecnología SRAM (Static Random Access Memory). Este tipo de celda permite que la configuración del dispositivo sea volátil, lo que significa que se pierde cuando se apaga, y requiere cargar la configuración desde una memoria externa cada vez que se enciende.

5) Realice la descripción en VHDL de un Flip Flop JK.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity ff_jk is
   Port (
      clk : in STD_LOGIC;
      j   : in STD_LOGIC;
      k   : in STD_LOGIC;
      q   : out STD_LOGIC
   );
end ff_jk;

architecture Behavioral of ff_jk is
   signal q_int : STD_LOGIC := '0';
begin
   process (clk)
   begin
      if rising_edge(clk) then
         if j = '0' and k = '1' then
            q_int <= '0';
         elsif j = '1' and k = '0' then
            q_int <= '1';
         elsif j = '1' and k = '1' then
            q_int <= not q_int;
         end if;
      end if;
   end process;
   q <= q_int;
end Behavioral;
```

6) Realice la descripción en VHDL de un sumador completo de un bit.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity sumador_completo is
   Port (
      a    : in  STD_LOGIC;
      b    : in  STD_LOGIC;
      c_in  : in  STD_LOGIC;
      suma  : out STD_LOGIC;
      c_out : out STD_LOGIC
   );
end sumador_completo;
```

```
architecture Behavioral of sumador_completo is
begin
   suma  <= a XOR b XOR c_in;
   c_out <= (a AND b) OR (c_in AND (a XOR b));
end Behavioral;
```

7) Realice la descripción en VHDL del test bench del sumador completo de un bit.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity tb_sumador_completo is
end tb_sumador_completo;

architecture test of tb_sumador_completo is
   signal a, b, c_in : STD_LOGIC := '0';
   signal suma, c_out : STD_LOGIC;

   component sumador_completo
      Port (
         a    : in  STD_LOGIC;
         b    : in  STD_LOGIC;
         c_in  : in  STD_LOGIC;
         suma  : out STD_LOGIC;
         c_out : out STD_LOGIC
      );
   end component;

begin
   uut: sumador_completo
      Port map (
         a    => a,
         b    => b,
         c_in  => c_in,
         suma  => suma,
         c_out => c_out
      );

   proceso_prueba: process
   begin
      a <= '0'; b <= '0'; c_in <= '0'; wait for 10 ns;
      a <= '0'; b <= '0'; c_in <= '1'; wait for 10 ns;
      a <= '0'; b <= '1'; c_in <= '0'; wait for 10 ns;
      a <= '0'; b <= '1'; c_in <= '1'; wait for 10 ns;
      a <= '1'; b <= '0'; c_in <= '0'; wait for 10 ns;
      a <= '1'; b <= '0'; c_in <= '1'; wait for 10 ns;
      a <= '1'; b <= '1'; c_in <= '0'; wait for 10 ns;
      a <= '1'; b <= '1'; c_in <= '1'; wait for 10 ns;
      wait;
   end process proceso_prueba;

end test;
```