En el caso de la Cyclone III, los elementos que constituyen los LE’s (Logic Elements) son:

* Una Look Up Table (LUT) de 4 entradas
* Un registro programable
* una “carry chain connection” (Por lo que entiendo, permite interconectar LE’s)
* una “register chain connection” (Permite interconectar los registros de las LE’s)
* Cada LE puede interconectar sus señales a distintas redes de conexión dentro del chip:

Local: conecta con LEs cercanos en la misma área

Row: conecta con otros LEs en la misma fila

Column: conecta con LEs en la misma columna

Register chain: conecta a través de la cadena de registros antes mencionada

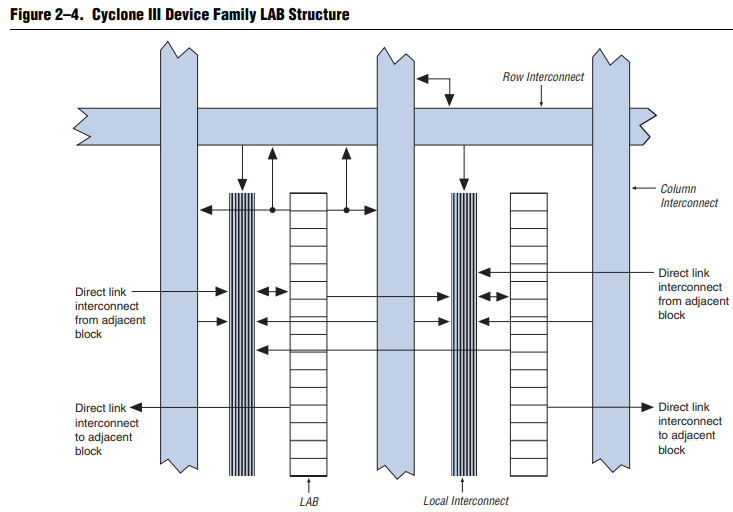
Direct link: permite una conexión directa y rápida con otro LE cercano sin pasar por la red general

* “Register packing support” (Permite que el registro y la lógica (LUT) se empaquen dentro del mismo LE)
* “Register feedback support” (Permite que la salida del registro dentro del LE se use como entrada de la LUT en el mismo LE)

Los LABs (Logic Array Blocks) tienen la siguiente topología:

* 16 LE’s
* LAB control signals
* LE carry chains
* Register chains
* Interconexiones locales

y la siguiente estructura:



2)

Nios® II hace referencia a una familia de procesadores Embebidos en la Cyclone III, se sintetiza dentro del FPGA, usando los Logic Elements (LEs), memorias y otros recursos internos del dispositivo.

3)

La principal diferencia es que los bloques embebidos son componentes fijos que vienen integrados en el fpga, mientras que los IP Cores (núcleos con Propiedad Intelectual) son módulos lógicos Programados (usan los recursos del fpga) diseñado por el fabricante o por terceros. Es necesario pagar para usar algunos.

4)Se utilizan celdas de programacion SRAM. La programación consiste en escribir valores (0 o 1) en las celdas SRAM, lo que define la lógica de cada celda y las conexiones entre ellas.

5) FFJK (con set, reset, sin salida negada)

library ieee;

use ieee.std\_logic\_1164.all;

entity FFJK is

port(

clk : in std\_logic;

J : in std\_logic;

K : in std\_logic;

reset : in std\_logic;

set : in std\_logic;

Q : out std\_logic

);

end entity;

architecture ff of FFJK is

SIGNAL Qa: std\_logic := '0';

begin

process (clk, set, reset)

begin

if reset = '1' then

Qa <= '0';

elsif set = '1' then

Qa <= '1';

elsif (rising\_edge(clk)) then

if (J = '0' and K = '0') then

Qa <= Qa;

elsif (J = '0' and K = '1') then

Qa <= '0';

elsif (J = '1' and K = '0') then

Qa <= '1';

elsif (J = '1' and K = '1') then

Qa <= not Qa;

end if;

end if;

end process;

Q <= Qa;

end architecture;

6) VHDL Full Adder

library ieee;

use IEEE.STD\_LOGIC\_1164.ALL;

entity FullAdder is

port( A: in std\_logic;

B: in std\_logic;

Cin: in std\_logic;

S: out std\_logic;

Cout: out std\_logic);

End FullAdder;

architecture behavioral of FullAdder is

signal BC1, BC2, AB, AC : std\_logic;

BEGIN

S <= A xor BC1;

BC1 <= B xor Cin;

BC2 <= B and Cin;

AB <= A and B;

AC <= A and Cin;

Cout <= BC2 or AB or AC;

End behavioral;

7) VHDL Full Adder testbench

library ieee;

use IEEE.STD\_LOGIC\_1164.ALL;

-- TESTBENCH NO TIENE ENTRADAS NI SALIDAS

entity FullAdder\_testbench is

End FullAdder\_testbench;

architecture pepe of FullAdder\_testbench is

-- DECLARO COMPONENTES DE TESTBENCH

component FullAdder

port(

A: in std\_logic;

B: in std\_logic;

Cin: in std\_logic;

S: out std\_logic;

Cout: out std\_logic

);

end component;

-- DECLARO SEÑALES

signal BC1, BC2, AB, AC : std\_logic;

-- Inputs

signal A : std\_logic := '0';

signal B : std\_logic := '0';

signal Cin : std\_logic := '0';

-- Outputs

signal S : std\_logic;

signal Cout : std\_logic;

BEGIN

-- INICIALIZO UUT (Unit under Test)

uut: FullAdder port map (

A => A,

B => B,

Cin => Cin,

S => S,

Cout => Cout

);

stim\_proc: process

BEGIN

A <= '0'; B <= '0'; Cin <= '0'; wait for 10ns;

A <= '0'; B <= '0'; Cin <= '1'; wait for 10ns;

A <= '0'; B <= '1'; Cin <= '0'; wait for 10ns;

A <= '0'; B <= '1'; Cin <= '1'; wait for 10ns;

A <= '1'; B <= '0'; Cin <= '0'; wait for 10ns;

A <= '1'; B <= '0'; Cin <= '1'; wait for 10ns;

A <= '1'; B <= '1'; Cin <= '0'; wait for 10ns;

A <= '1'; B <= '1'; Cin <= '1'; wait for 10ns;

wait;

END process;

End pepe;