Ingeniería de Computadores III

**Curso 2020-2021**

Entrega de Abril

*María Esther Ramos Iglesias. mramos1238*

*email:* [*mramos1238@alumno.uned.es*](mailto:mramos1238@alumno.uned.es)

*Centro asociado: UNED Aranjuez*

Ejercicio 1

En el ejercicio 1 se va a trabajar en el diseño de un circuito digital que implemente las funciones F1 y F2 a partir de la siguiente tabla de verdad.

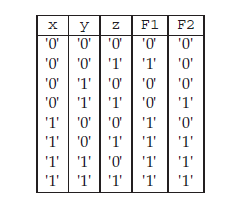


Imagen 1: Tabla de verdad de las funciones F1 y F2

## Apartado a)

En el primer apartado se pide implementar la ***entity*** del circuito. Declaramos los puertos que serán visibles desde el exterior, que en este caso se corresponden con las entradas y salidas de la tabla de verdad.

library IEEE;

use IEEE.std\_logic\_1164.all;

Entity *ejercicio1\_a* is

    port(

        x,y,z: in *std\_logic*;

        F1: out *std\_logic*;

        F2: out *std\_logic*);

end entity *ejercicio1\_a*;

## Apartado b)

Para este apartado desarrollamos una ***architecture*** que va a definir el comportamiento del circuito para la ***entity*** creada previamente.

Nuestro primer paso será crear las funciones booleanas que definen el circuito utilizando para ello [Karnaugh](https://es.wikipedia.org/wiki/Mapa_de_Karnaugh).

F1 = x + ӯz

F2 = x(y+z) + yz

library IEEE;

use IEEE.std\_logic\_1164.all;

*architecture ejercicio1\_b of ejercicio1\_a is*

*begin*

*--F1: x + ((not y)z)*

*--F2: x( y + z ) + yz*

*F1 <= x or ((not y) and z);*

*F2 <= (x and (y or z)) or (y and z);*

*end architecture ejercicio1\_b*;

## Apartado c)

Si diseñamos el circuito obtenido en el apartado b) obtenemos la siguiente figura

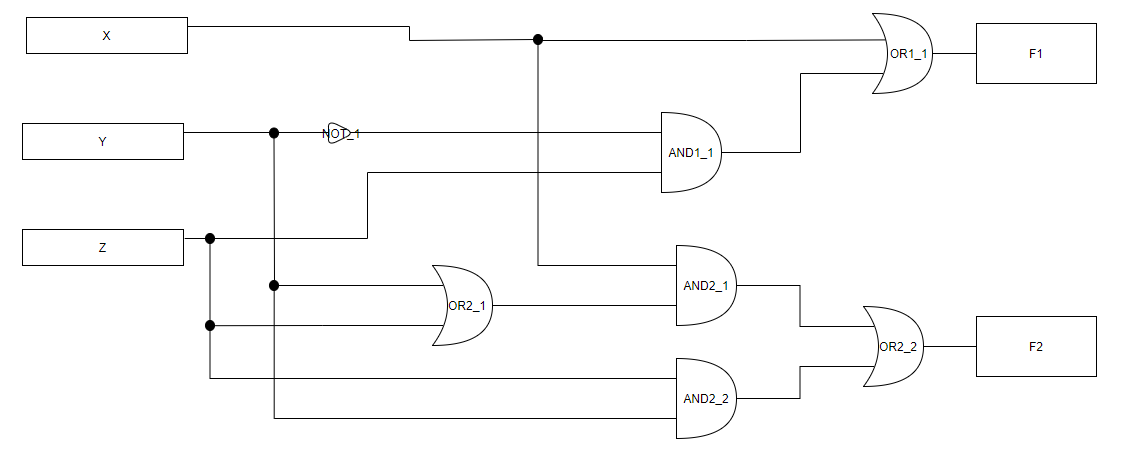


Imagen 2: Diseño del Circuito

Una vez tenemos el circuito terminado y sabemos el tipo de puertas lógicas necesitamos en el proyecto, procedemos a crear su ***entity*** y ***architecture***

library IEEE;

use IEEE.std\_logic\_1164.all;

entity *not1* is    -- Puerta not: 1 entrada.

    port(

        x: in *std\_logic*;

        F: out *std\_logic* );

end entity *not1*;

*architecture not1 of not1 is*

*begin*

*F <= not x;*

*end architecture not1*;

library IEEE;

use IEEE.std\_logic\_1164.all;

entity *and1* is  -- Puerta lógica and: 2 entradas.

    port(

        x,y: in *std\_logic*;

        F: out *std\_logic*);

end entity *and1*;

*architecture and1 of and1 is*

*begin*

*F <= x and y;*

*end architecture and1*;

library IEEE;

use IEEE.std\_logic\_1164.all;

entity *or1* is  -- Puerta lógica or: 2 entradas.

    port(

        x,y: in *std\_logic*;

        F: out *std\_logic* );

end entity *or1*;

*architecture or1 of or1 is*

*begin*

*F <= x or y;*

*end architecture or1*;

## Apartado d)

Escribimos en código una ***architecture*** que describa la estructura del circuito utilizando las puertas lógicas creadas en el apartado c). Para ello utilizaremos señales auxiliares

Para las señales se ha seguido la estructura sP1\_X o sP2\_X :

* P : Inicial de la puerta lógica a utilizar
* X : Identificador numérico (Para diferenciar sí se usa la misma puerta lógica ).

library IEEE;

use IEEE.std\_logic\_1164.all;

*architecture ejercicio1\_d of ejercicio1\_a is*

*component not1 is*

*port(*

*x: in std\_logic;*

*F: out std\_logic);*

*end component not1;*

*component and1 is*

*port(*

*x,y: in std\_logic;*

*F: out std\_logic*

*);*

*end component and1;*

*component or1 is*

*port(*

*x,y: in std\_logic;*

*F: out std\_logic);*

*end component or1;*

*signal sO2\_1, sA2\_1, sA2\_2, sO2\_2, sN1\_1, sA1\_1, sO1\_1 : std\_logic;*

*begin*

*N1\_1: component not1 port map (y, sN1\_1);*

*A1\_1: component and1 port map (sN1\_1, z, sA1\_1);*

*O1\_1: component or1 port map (x, sA1\_1, F1);*

*O2\_1: component or1 port map (y, z, sO2\_1);*

*A2\_1: component and1 port map (x, sO2\_1, sA2\_1);*

*A2\_2: component and1 port map (y, z, sA2\_2);*

*O2\_2: component or1 port map (sA2\_1, sA2\_2, F2);*

*end architecture ejercicio1\_d*;

## Apartado e)

Vamos a comprobar el correcto funcionamiento de las ***arquitecture*** creadas y para ello hemos creado una batería de pruebas.

library IEEE;

use IEEE.std\_logic\_1164.all;

entity *ejercicio1\_e* is

end entity *ejercicio1\_e*;

*architecture ejercicio1\_e of ejercicio1\_e is*

*signal x,y,z : std\_logic;*

*signal F1,F2: std\_logic;*

*component ejercicio1\_a is*

*port (*

*x,y,z: in std\_logic;*

*F1,F2: out std\_logic);*

*end component ejercicio1\_a;*

*begin*

*uut: component ejercicio1\_a port map (x,y,z,F1,F2);*

*test : process*

*begin*

*x <= '0', '1' after 40 ns ;*

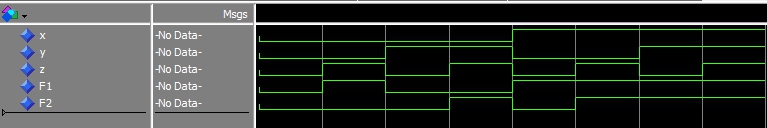
*y <= '0', '1' after 20 ns, '0' after 40 ns, '1' after 60 ns;*

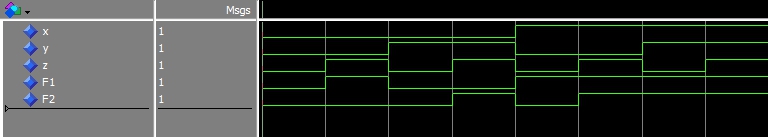
*z <= '0', '1' after 10 ns, '0' after 20 ns, '1' after 30 ns, '0' after 40 ns, '1' after 50 ns, '0' after 60 ns, '1' after 70 ns ;*

*wait;*

*end process test;*

*end architecture ejercicio1\_e*;

Con las pruebas ya creadas nos salen los siguientes cronogramas:



Ejercicio 2