Práctica 00: Introducción a VHDL

Por:
Ian Gabriel Cañas Fernández, 1092228

.
.

Objetivos generales:

- Aprender los conceptos básicos en VHDL
- Familiarizarse con el software Vivado

Experimento *Hola Mundo*:

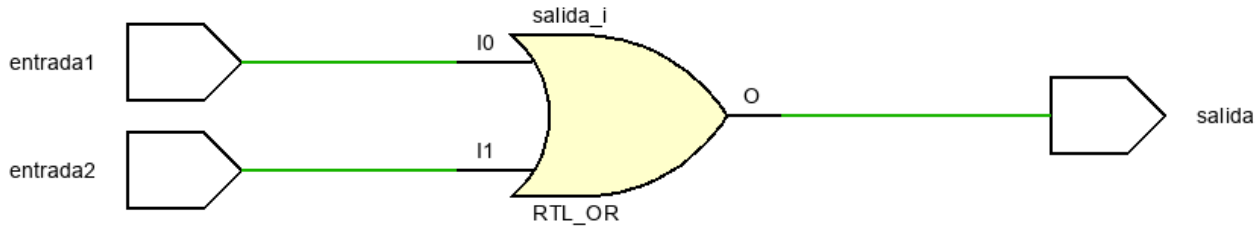
Reconociendo el comportamiento de una compuerta lógica *OR*.

Objetivos:

- Familiarizarnos con el comportamiento de la compuerta lógica OR
- Reconocer los inputs y outputs en una FPGA

Procedimiento:

Nos presentamos ante una compuerta OR, representada mediante el siguiente diagrama lógico.



Es decir, la salida será verdadera cuando al menos una de las dos entradas de la compuerta lo sea. Se transcribe la compuerta a VHDL y se carga a la tarjeta Nexys A7, asignándole las entradas y salidas como corresponde.

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

entity HolaMundo is
    Port (  entradal : in STD_LOGIC;
            entrada2 : in STD_LOGIC;
            salida : out STD_LOGIC);
end HolaMundo;

architecture Behavioral of HolaMundo is
begin

salida <= entradal or entrada2;

end Behavioral;
```

Resultados:
Habiendo probado el sistema, obtenemos los siguientes resultados en base a las entradas de la compuerta. Estas predicciones se confirman fielmente con los resultados obtenidos mediante la FPGA.

Entrada1	Entrada2	Salida (entrada1 OR entrada2)
0	0	0
0	1	1
1	0	1
1	1	1*

Tabla 1 – Compuerta lógica de *Hola Mundo*.

* En teoría la compuerta anteriormente vista representa una suma entre los valores, pero tomando en cuenta que 1 representa realmente alto voltaje, el resultado de 1+1 se sigue presentando como 1, pues sigue siendo un alto voltaje.

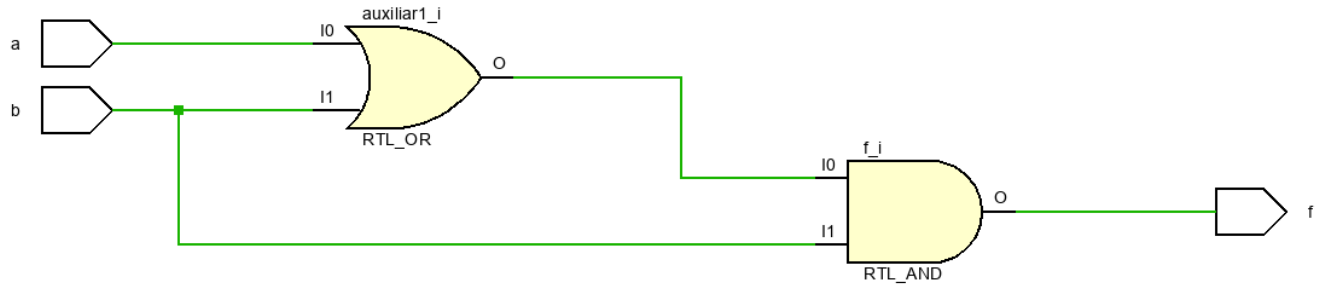
Análisis:
Se comprueba experimentalmente que los valores esperados mediante la teoría son totalmente acordes entre sí y que, la compuerta OR representa una suma entre las entradas, viendo, mediante Boole, que 1 corresponde a un valor verdadero o alto (High) voltaje.

Experimento *Uso de Señales*:
Combinando los comportamientos de las compuestas AND y OR de la forma $(a\ OR\ b)\ AND\ b$.

Objetivos:

- Reconocer cuando nos presentamos ante una combinación de compuertas lógicas
- Introducir el uso de señales

Procedimiento:
Nos presentamos ante una compuerta AND, en la que uno de los valores de entrada depende del valor de salida de otra compuerta, a este "valor intermedio" se le llama una señal. A continuación, se presenta el diagrama lógico de la arquitectura del circuito.



Viendo la anterior estructura, se puede predecir que, para que el valor de salida sea verdadero, ambas entradas han de serlo, por lo que necesariamente b ha de contener un valor verdadero.

Se escribe la arquitectura en el archivo y se pone a prueba con la FPGA.

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

entity UsoDeSenales is
  Port (    a: in STD_LOGIC;
            b: in STD_LOGIC;
            f: out STD_LOGIC);
end UsoDeSenales;

architecture Behavioral of UsoDeSenales is
signal auxiliar1 : std_logic;

begin

auxiliar1 <= a or b;
f <= auxiliar1 and b;

end Behavioral;
```

Resultados:

Habiendo probado el sistema, obtenemos los siguientes resultados en base a las entradas de la compuerta. Estas predicciones se confirman fielmente con los resultados obtenidos mediante la FPGA.

a	b	Auxiliar_1 (a OR b)	F (auxiliar_1 AND b)
0	0	0	0
0	1	1	1
1	0	1	0
1	1	1	1

Tabla 2 – Compuerta lógica de *Uso de señales*.

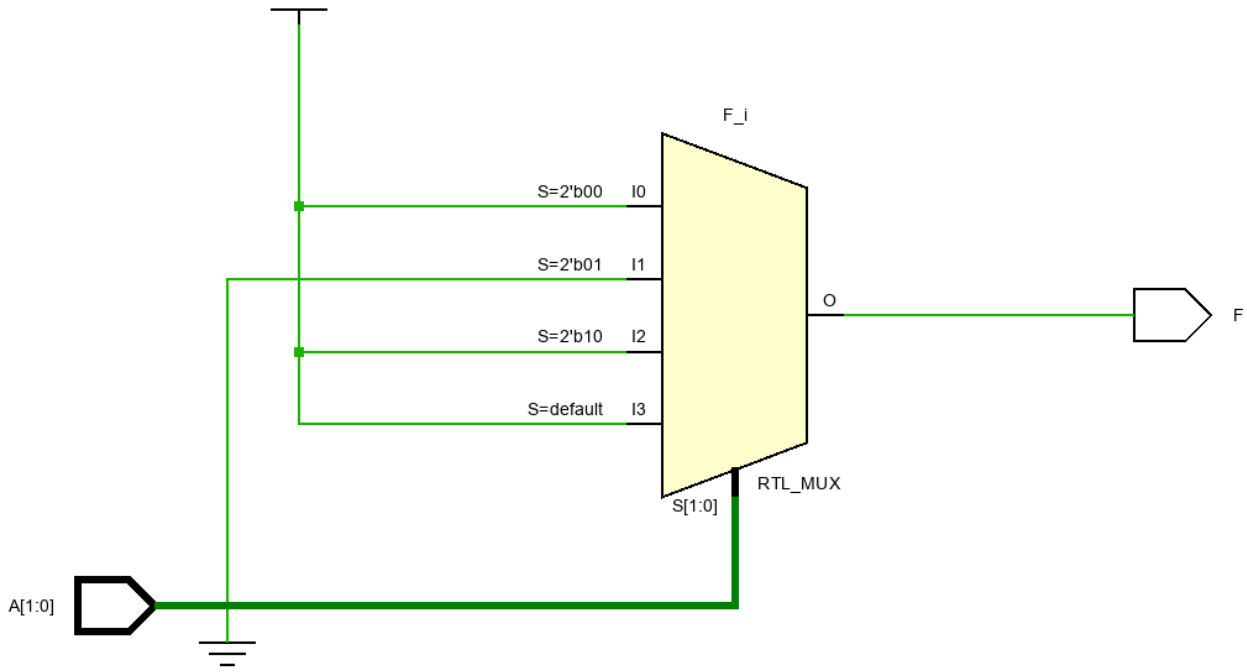
Análisis:
Se comprueba experimentalmente que los valores esperados mediante la teoría son totalmente acordes entre sí y que, la compuerta AND podría ser vista como un producto entre las entradas. Además, se percibe la facilidad y fluidez con la que las señales son trabajadas en VHDL.

Experimento *Tabla de verdad*:
Responder a una única combinación de estímulos.

Objetivos:
- Introducir y reconocer los vectores de bits
- Involucrar las sentencias concurrentes

Procedimiento:
En este programa reconocemos los vectores lógicos, que no son más que un conjunto de valores almacenados bajo un solo nombre. También puede ser visto como un número con varios dígitos. Como el vector es del tipo lógico, estaríamos viendo una representación de los números binarios.

Nos presentamos ante una sentencia concurrente, que son sentencias condicionales que tienen al menos un valor por defecto que ha de cumplirse para asignar uno u otro valor. Esta puede verse como el clásico *if-else*, con la obligación de que exista el *else*. La sentencia que presentamos en esta configuración es WITH-SELECT-ELSE. Se representa mediante el siguiente diagrama.



Viendo la anterior estructura, veremos, entonces, que obtendremos el valor falso en la única condición en que el valor de A sea 01

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

entity TablaDeVerdad is
  Port (    A: in STD_LOGIC_VECTOR(1 DOWNTO 0);
            F: out STD_LOGIC);
end TablaDeVerdad;

architecture Behavioral of TablaDeVerdad is
begin

    with A select
        F <=    '1' when "00",
                '0' when "01",
                '1' when "10",
                '1' when others;

end Behavioral;
```

Resultados:

Habiendo probado el sistema, obtenemos los siguientes resultados dependientes del valor del vector A.

A	F

00	1
01	0
10	1
11	1

Tabla 3 – Valores de la sentencia concurrente.

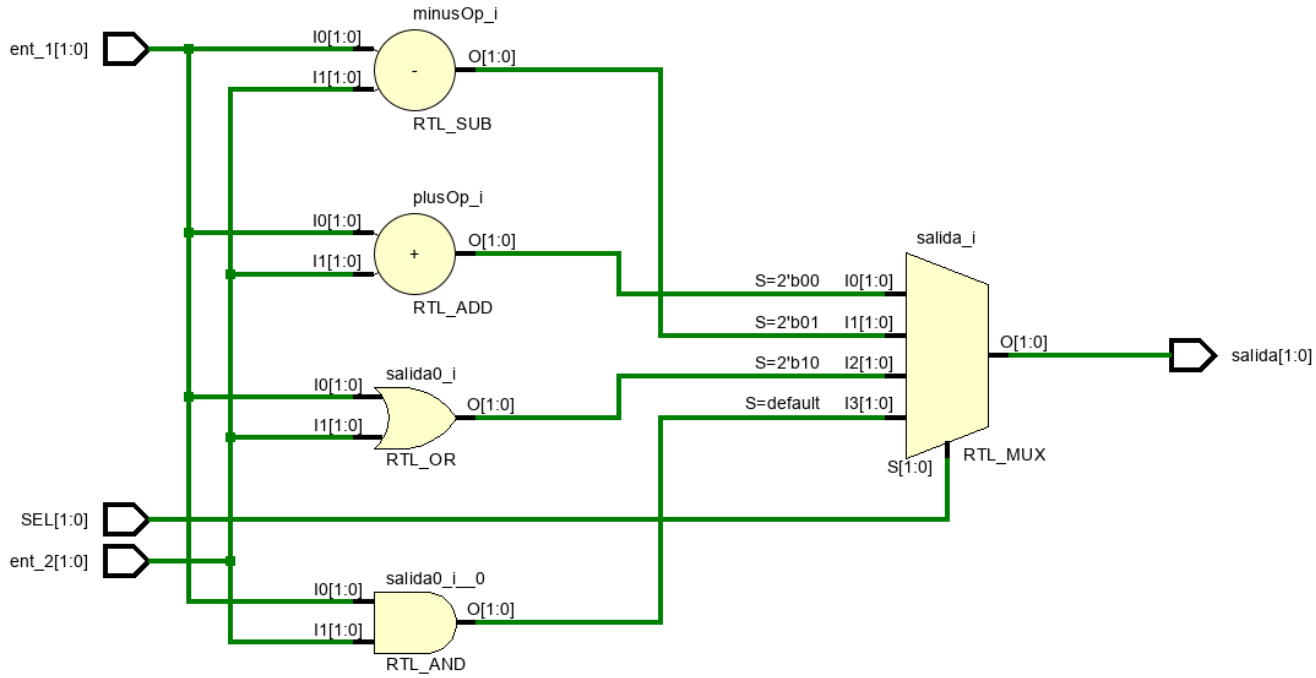
Análisis:
Se comprueba experimentalmente que los valores esperados mediante la teoría son totalmente acordes entre sí y que se va a cumplir con fluidez la veracidad de la condición que se presente.

<u>Experimento *Unidad Aritmético-Lógica*</u>:
Acudiendo a un cálculo u otro bajo ciertas condiciones.

Objetivos:
- Realizar cálculos simples con números binarios de dos bits
- Involucrar las sentencias concurrentes

Procedimiento:
En este programa reconocemos tratamos los vectores lógicos como números binarios y vemos las sentencias bajo un caso más complejo.

Podemos ver que el vector SEL indicará cuál de las operaciones hechas anteriormente será aquella a ser presentada.



El código VHDL de el anterior esquema se presenta a continuación:

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

entity ALU is
Port (SEL : in std_logic_vector(1 downto 0);
      ent_1 : in std_logic_vector (1 downto 0);
      ent_2 : in std_logic_vector (1 downto 0);
      salida : out std_logic_vector (1 downto 0));
end ALU;

architecture Behavioral of ALU is

begin

with SEL select
    salida <= ent_1 + ent_2 when "00",
              ent_1 - ent_2 when "01",
              ent_1 or ent_2 when "10",
              ent_1 and ent_2 when others;


end Behavioral;
```

Resultados:

Habiendo probado el sistema, obtenemos los siguientes resultados dependientes para todas las operaciones que se puedan seleccionar mediante SEL

Ent_1	Ent_2	Ent_1 + Ent_2	Ent_1 - Ent_2	Ent_1 OR Ent_2	Ent_1 AND Ent_2
00	00	00	00	00	00
00	01	01	11**	01	00
00	10	10	10**	10	00
00	11	11	01**	11	00
01	00	01	01	01	00
01	01	10	00	01	01
01	10	11	11**	11	00
01	11	00*	10**	11	01
10	00	10	10	10	00
10	01	11	01	11	00
10	10	00*	00	10	10
10	11	01*	11**	11	10
11	00	11	11	11	00
11	01	00*	10	11	01
11	10	01*	01	11	10
11	11	10*	00	11	11

Tabla 2 – Valores de la sentencia concurrente.

* Estos valores ocupan realmente más dígitos por la naturaleza del resultado (ejemplo: 101 mostrará 01)
** Como las computadoras no trabajan con el valor negativo como tal, el resultado lo presentan mediante el complemento a dos del correspondiente resultado negativo (ejemplo: $01 - 11 = -10 \rightarrow 01 + 1 = 10$)

Análisis:

Se comprueba experimentalmente que los valores esperados mediante la teoría son acordes entre sí, igual podemos observar que la FPGA, para no presentar valores negativos, devuelve el complemento A2 de dicho negativo, pues se le es más familiar y natural para trabajar. A parte, podemos observar, que las comparativas OR y AND se analizan dígito por dígito. Y, aunque OR sea equivalente a una suma entre ambas entradas, cuando las entradas son vectores lógicos, la equivalencia se pierde, pues OR seguirá trabajando dígito por dígito.