Practica 7

Operaciones Aritméticas

Profesor: Barrón Vera José Emanuel
Materia: Fundamentos de diseño digital
Grupo: 3CV6
Alumno: Cazares Cruz Jeremy Sajid
Boleta: 2021630179

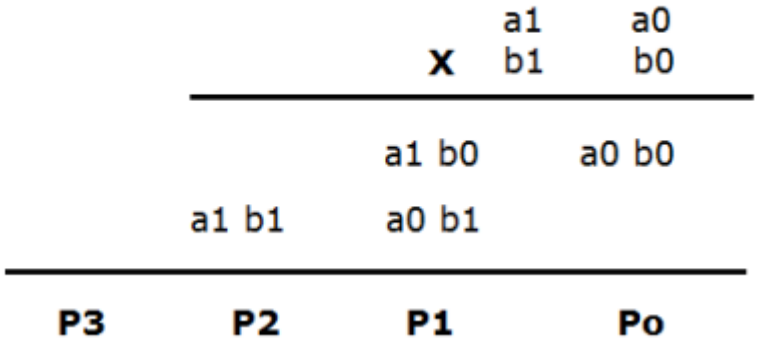
Multiplicador binario

x	0	1
0	0	0
1	0	1

Desarrollo experimental

Mediante el desarrollo de la práctica se tuvo la implementación de un código propio de tal manera que las funciones son enteramente realizadas por nosotros, de esta manera se comprendió de mejor manera la implementación del código VHDL para diferentes utilidades.

En este caso se tiene el uso de un multiplicador binario de dos bits, por lo tanto, se tiene 4 entradas y posibles salidas las cuales son el resultado de las sumas realizadas por la multiplicación



Como se puede observar P1 y P2 son el resultado de la suma del producto de a0, a1, b0 y b1, sin embargo, se tiene el uso de P3 debido en el caso que exista un acarreo desde P1, siendo esto solo posible en el caso donde se multiplique tres por tres o bien el número binario 11 por sí mismo.

Para la realización en código de la práctica se tiene que pueden usarse las salidas de los resultados de la multiplicaciones de tal manera que se realizaría la forma de un medio sumador entre sus componentes, sin embargo esto sería de una manera más complicada y quizás no la manera óptima de realizarlo ya que entre sus componentes de la multiplicación estaría el acarreo de una suma anterior así como la implementación de 3 sumadores completos donde el primero de estos no tendría un acarreo de entrada debido a que P0 jamás pasara a ser más de 1 mientras que P1 y P2 está la posibilidad de tener un acarreo de salida a la siguiente suma de los productos.

Se hizo el análisis de la tabla de verdad para así realizar el correcto funcionamiento de la multiplicación, de igual manera se realizaron los cálculos teóricos de las posibles multiplicaciones, corroborando que lo presentado dentro de la tabla de verdad sea verdadero.

La tabla de verdad correspondiente es la siguiente:

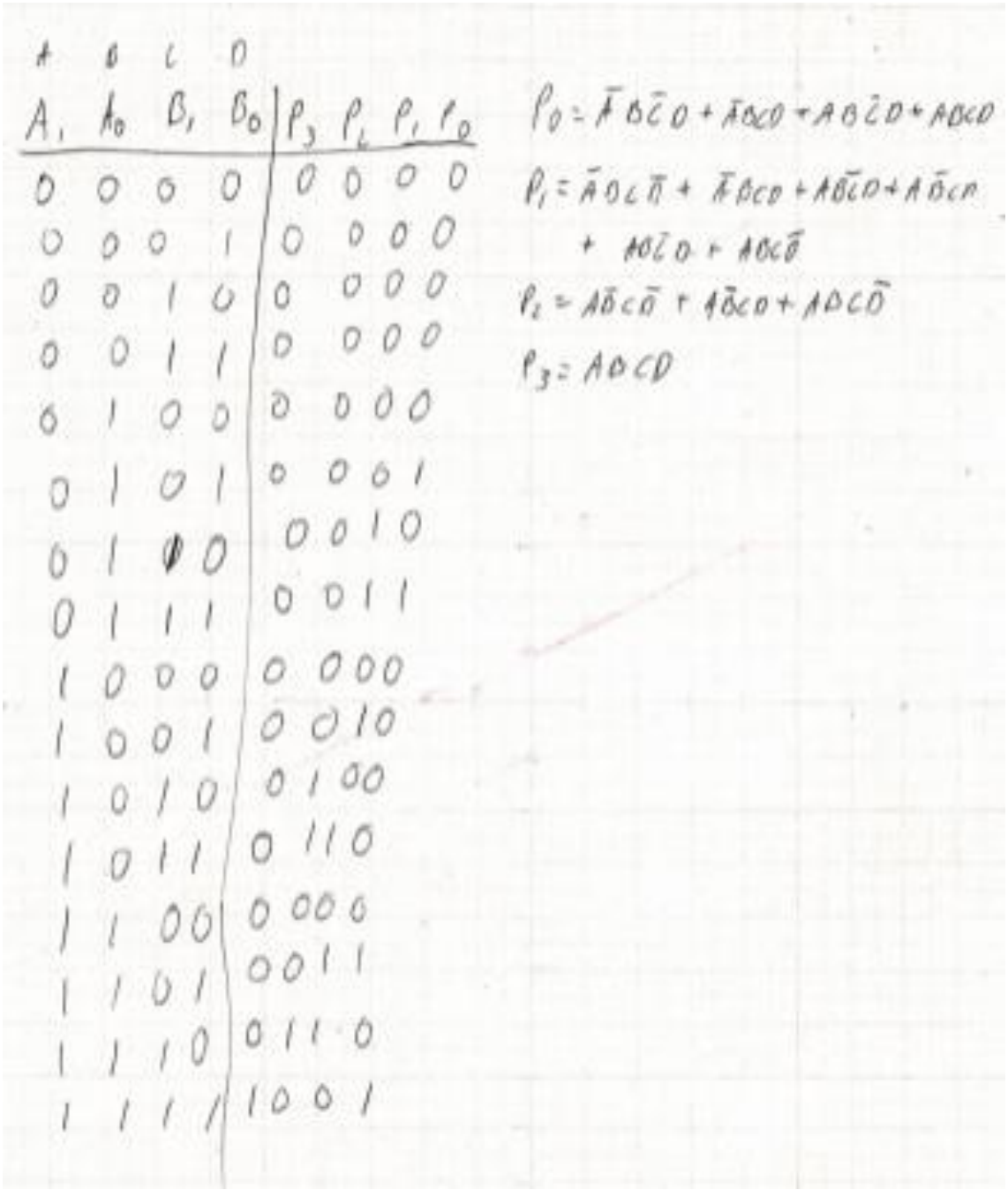


figura 1 tabla de verdad y minterminos

De tal manera que se puede ver los minterminos resultantes de la tabla, al ser estos “grandes” se les aplicara mapas de Karnaugh para la simplificación óptima para el problema

Sim embargo, para el ultimo digito o bit (pP3) no se tiene una simplificación al solo tener una función única la cual es cuando todo sea 1 de caso contrario no existe condición para que esta función sea igual a 1

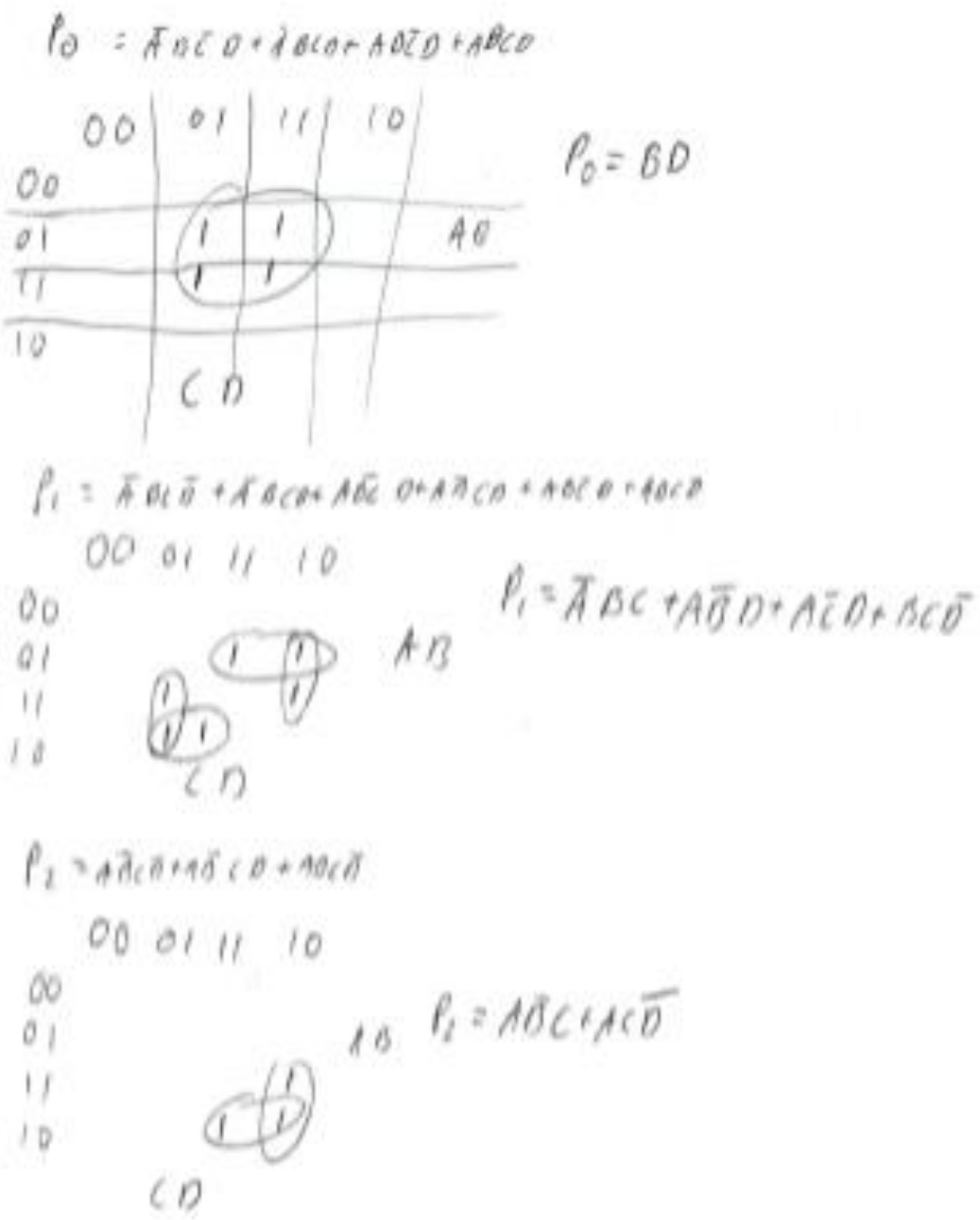


figura 2 Simplificación por mapa de Karnaugh

En la figura 2 se puede observar la simplificación de las variables de la tabla de verdad siendo así una implementación más sencilla en VHDL

De igual forma en la siguiente figura se presentará el circuito resultante el cual se hizo mediante la simplificación de las salidas

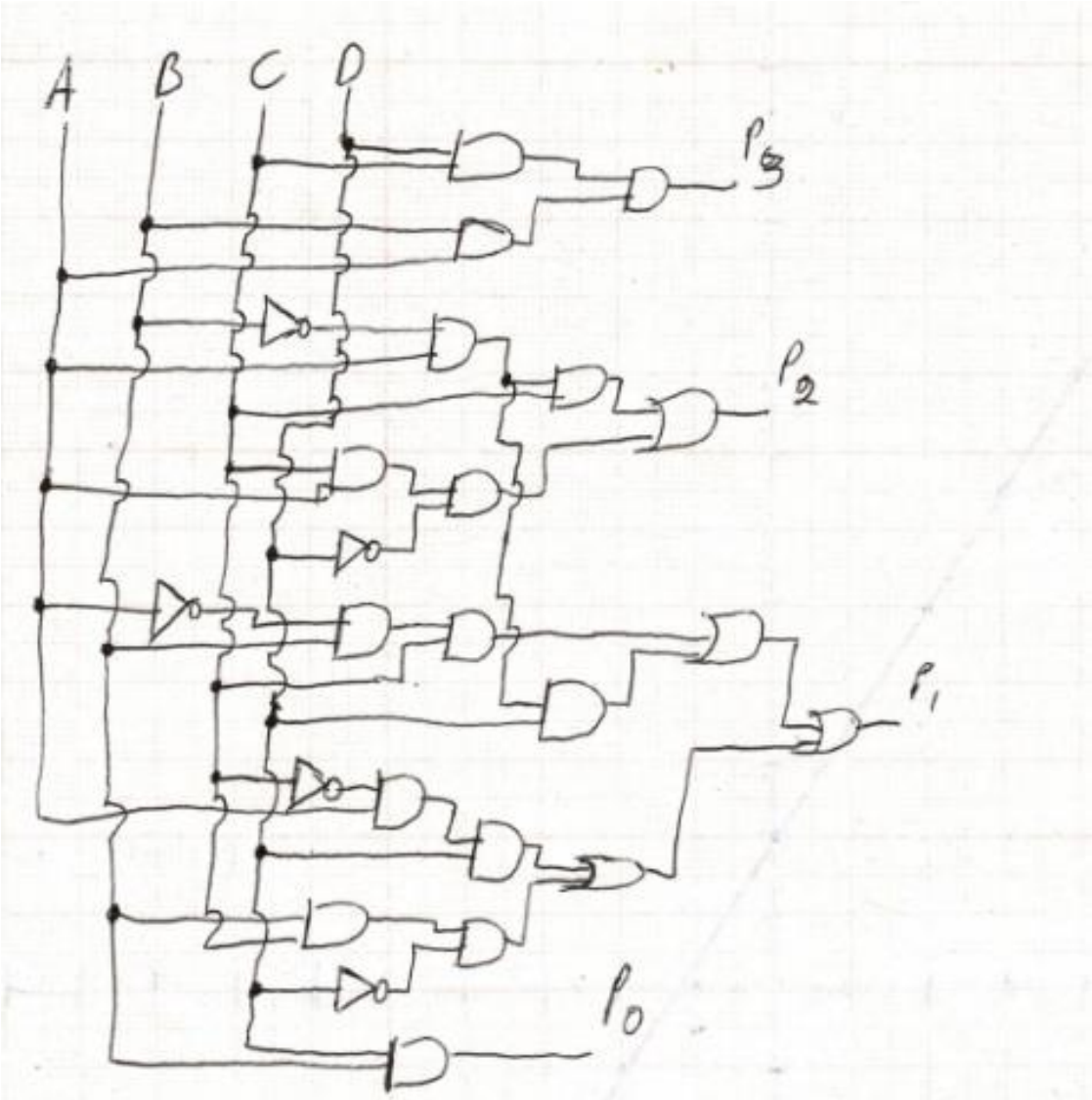


figura 3 Circuito resultante

De igual forma para el circuito resultante se tomó en cuenta las variables de entrada como “ABCD”, sin embargo, se tiene como a1, a0, b1 y b0 correspondientemente donde A es a1 y C es b1

```
1 LIBRARY ieee;
2 USE ieee.std_logic_1164.all;
3
4 ENTITY mult IS PORT(
5     a0,a1,b0,b1:IN  STD_LOGIC;
6     p0,p1,p2,p3: OUT STD_LOGIC
7     );
8
9 END mult;
```

figura 4 Código parte 1

Se declara la librería a utilizar, así como la entidad donde de igual manera se declaran los pines de entrada y salida asignándoles nombres, en este caso se utiliza los nombres de a0, a1, b0 y b1 en referencia a la multiplicación que se realizara posteriormente de estos bits. Mientras que los pines de salida se les llama como p0, p1, p2 y p3 siendo p0 el bit de menos significativo y p3 el bit más significativo

```
11 architecture funcion of mult is
12 begin
13         p3<=(a1 and a0) and (b1 and b0);
14         p2<=((a1 and (not a0)) and b1) or ((a1 and b1) and (not b0));
15         p1<=(((not a1) and a0) and b1) or ((a1 and (not a0)) and b0) or ((a1 and (not b1)) and b0) or ((a0 and b1) and (not b0));
16         p0<=(a0 and b0);
17
18 end funcion;
```

figura 5 Código parte 2

Se inicializa la arquitectura donde se realizará la función del gal, donde como ya se había mencionado se utiliza las funciones simplificadas de los minterminos de esta manera se cumple la tabla de verdad que a su vez es la multiplicación de 2 bits tal como se indica a llegar en la practica

```
11 architecture funcion of mult is
12 begin
13         p3<=(a1 and a0) and (b1 and b0);
14         p2<=((a1 and (not a0)) and b1) or
15         p1<=(((not a1) and a0) and b1) or
16         p0<=(a0 and b0);
17
18 end funcion;
```

figura 6 código

Simulaciones

Finalmente, para la parte de la simulación del código se tiene que los casos de mayor interés son aquellos donde en A y B exista por lo menos un bit en 1, de caso contrario donde solo en A o solo en B se tiene un bit en 1 pero en el otro lado no existe ningún bit en 1 el resultado será cero en todas sus salidas

Una vez teniendo en cuenta esto se tiene lo siguiente

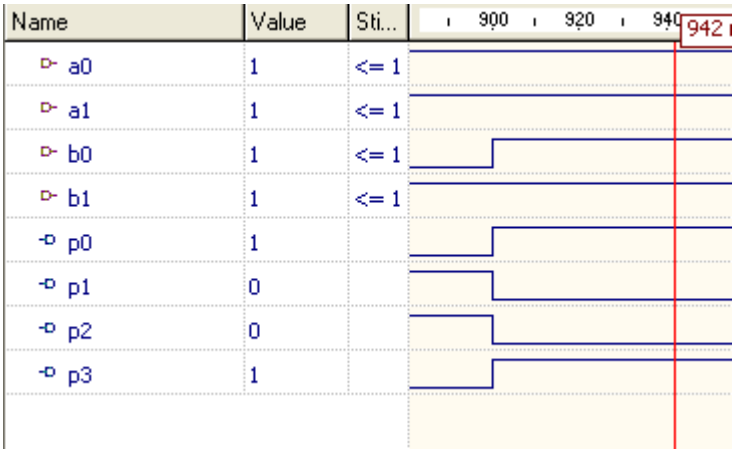


figura 7 3x3

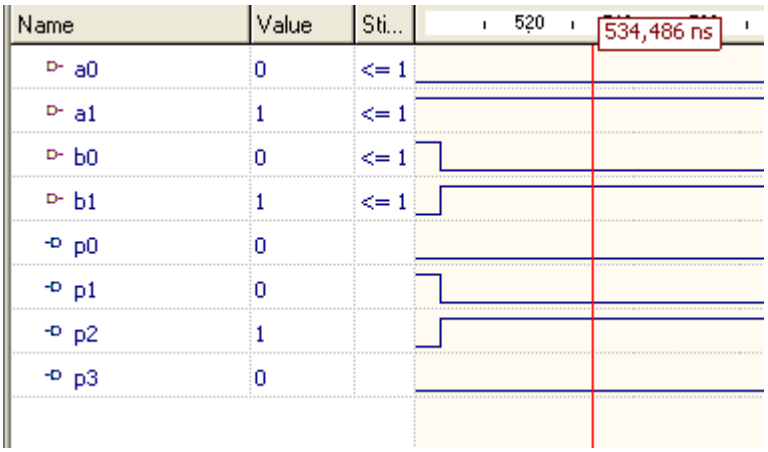


figura 8 2x2

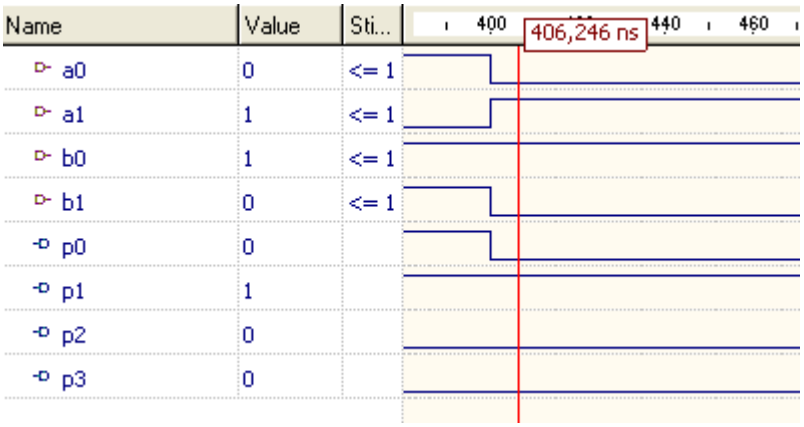


figura 9 2x1

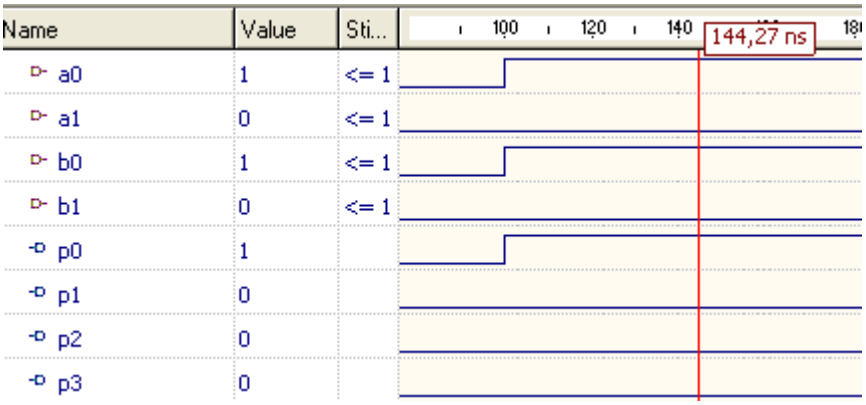


figura 10 1x1

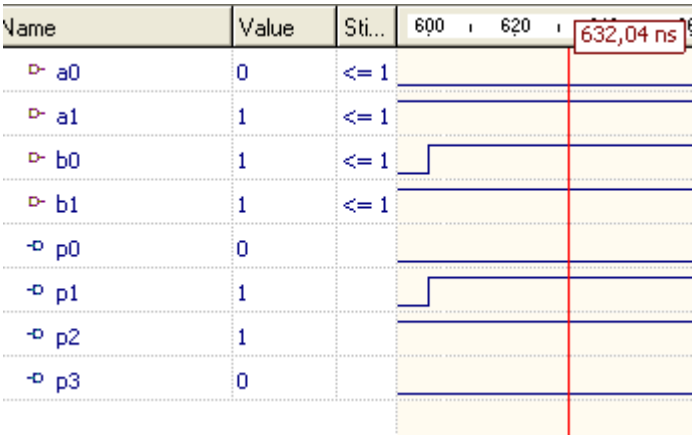


figura 11 3x2

Y se tiene una prueba de escritorio donde se puede observar lo simulado de tal manera que los resultados son del todo correctos

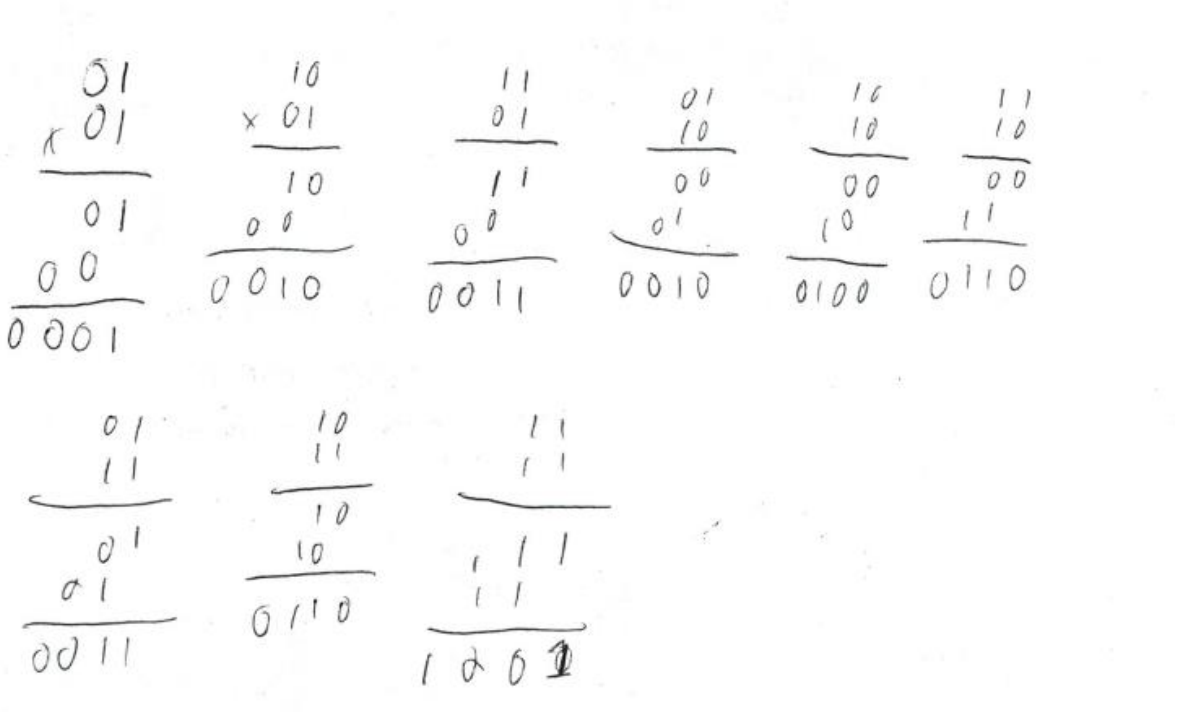


figura 12 prueba de escritorio

Conclusiones

Con ayuda de la práctica se pudo diseñar un circuito combinatorio a partir de un enunciado de tal manera de a futuro tener la habilidad de diseñar, probar y comprobar circuitos combinatorios mediante un solo enunciado

Bibliografía

VHDL: uniendo circuitos, un sumador completo de 4 bits •. (2019, 8 agosto). JnjSite.com. https://jnjsite.com/vhdl-uniendo-circuitos-un-sumador-completo-de-4-bits/

U. (2021, 17 noviembre). *Multiplicador 2x2 Bits, salida 4 Bits*. RoboBox. http://robobox101.blogspot.com/2013/07/multiplicador-2x2-bits-salida-4-bits.html