Laboratorio 1
Tecnológico de Costa Rica

Bryan Esquivel Flores
2020035806
zayus@estudiantec.cr
Alajuela, Costa Rica

Freddy Mora Bolaños
2021040443
freddy.mora@estudiantec.cr
Alajuela, Costa Rica

Andrés Vargas Arce
2018151379
anbo80@estudiantec.cr
Alajuela, Costa Rica

August 23, 2024

1 Ejercicio 1

Para este primer ejercicio se diseñarán los bloques marcados en verde del siguiente diagrama de bloques:

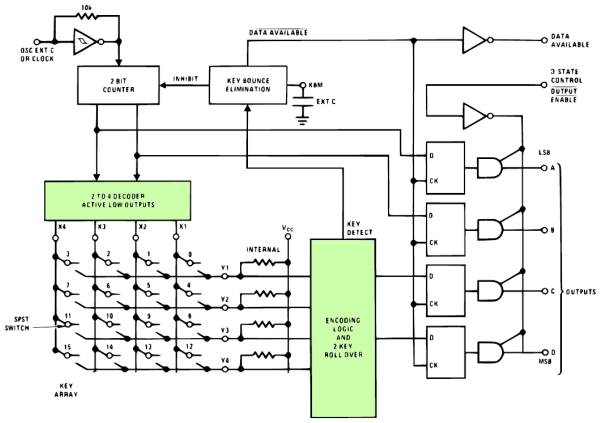

Figura 1: Diagrama de bloques codificador de matrices de 16 teclas MM74C922

Para el bloque llamado "Encoding Logic And 2key Roll Over" se realizó el siguiente análisis:

V1	V2	V3	V4	S1	S2
0	0	0	0	X	X
0	0	0	1	1	1
0	0	1	0	1	0
0	1	0	0	0	1
1	0	0	0	0	0
X	X	X	X	X	X

Tabla 1: Tabla de verdad del bloque "Encoding Logic And 2key Roll Over".

La simplificación por medio de mapas de Karnaugh es la siguiente:

V3V4 \ V1V2	00	01	11	10
00	X	X	X	X
01	1	X	X	X
11	X	X	X	X
10	1	X	X	X

Figura 2: Mapa de Karnaugh S1.

V3V4 \ V1V2	00	01	11	10
00	X	1	X	X
01	1	X	X	X
11	X	X	X	X
10	X	X	X	X

Figura 3: Mapa de Karnaugh S2.

Dando como resultado:

$$S1 = \overline{V1} \cdot \overline{V2} \qquad\qquad S2 = \overline{V1} \cdot \overline{V3} \tag{1}$$

S1 y S2 se comportan de la misma forma que una compuerta NOR por lo que podemos utilizar el siguiente circuito:

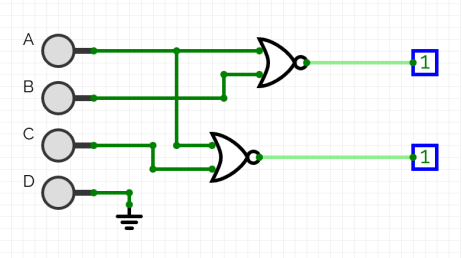

Figura 4: Circuito para el bloque "Encoding Logic And 2key Roll Over".

Donde A=V1,B=V2,C=V3 y D=V4

1.1 Decodificador de 2 a 4 activo en bajo

J1	J2	A	B	C	D
0	0	0	1	1	1
0	1	1	0	1	1
1	0	1	1	0	1
1	1	1	1	1	0

Tabla 2: Tabla de verdad del bloque "Decodificador de 2 a 4".

La simplificación por medio de mapas de Karnaugh es la siguiente:

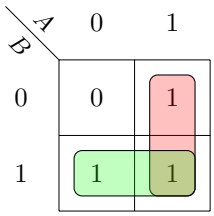

Figura 5: Mapa de Karnaugh Q_0.

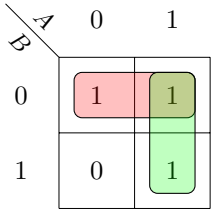

Figura 6: Mapa de Karnaugh Q_1.

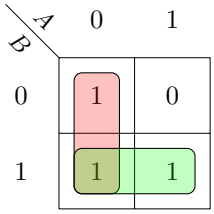

Figura 7: Mapa de Karnaugh Q_2.

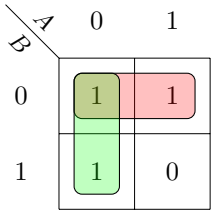

Figura 8: Mapa de Karnaugh Q_3.

Dando como resultado:

$$Q_0 = A + B \qquad Q_1 = \overline{B} + A \tag{2}$$

$$Q_2 = \overline{A} + B \qquad Q_3 = \overline{A} + \overline{B} \tag{3}$$

Tomando en cuanta las ecuaciones anteriores, se puede tomar dos NOR y ponerlas consecutivas para representar Q_0 por ejemplo a su vez usando el mismo procedimiento con Q_2 y Q_3 solo que negando una entrada de una de ellas para poder después negarlas. En cambio Q_3. Por lo tanto, quedaría el siguiente circuito:

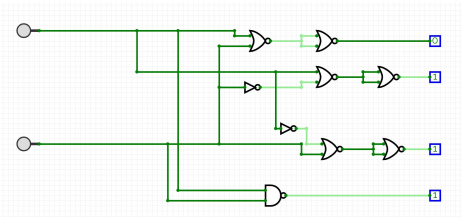

Figura 9: Circuito para el bloque "Decodificador de 2 a 4 activo en bajo".

2 Ejercicio 2

En este apartado se muestran los resultados obtenidos en la simulación del módulo complemento a 2 y su respectiva tabla de verdad. Además, el código utilizado para el módulo y el TB se pueden encontrar en Repositorio.

Interruptor3	Interruptor2	Interruptor1	Interruptor0	LED3	LED2	LED1	LED0
0	0	0	0	0	0	0	0
0	0	0	1	1	1	1	1
0	0	1	0	1	1	1	0
0	0	1	1	1	1	0	1
0	1	0	0	1	1	0	0
0	1	0	1	1	0	1	1
0	1	1	0	1	0	1	0
0	1	1	1	1	0	0	1
1	0	0	0	1	0	0	0
1	0	0	1	0	1	1	1
1	0	1	0	0	1	1	0
1	0	1	1	0	1	0	1
1	1	0	0	0	1	0	0
1	1	0	1	0	0	1	1
1	1	1	0	0	0	1	0
1	1	1	1	0	0	0	1

Tabla 3: Tabla de verdad del bloque "Complemento a 2".

2.1 Resultados del funcionamiento

Debido a la configuración de los LEDs de la FPGA, donde encienden en bajo por tener el ánodo fijo en Vcc, se tuvo que invertir la salida final del complemento a 2.

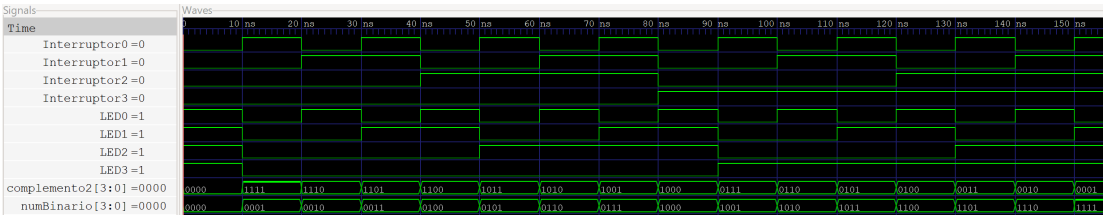

Figura 10: Simulación módulo complemento a 2.

3 Ejercicio 3

En este apartado se muestran los resultados obtenidos en la simulación del multiplexor 4:1, el código utilizado para el multiplexor y el TB se pueden encontrar en Repositorio la carpeta Ejercicio 3, dentro del Laboratorio 1.

3.1 Resultados buses de 4 bits

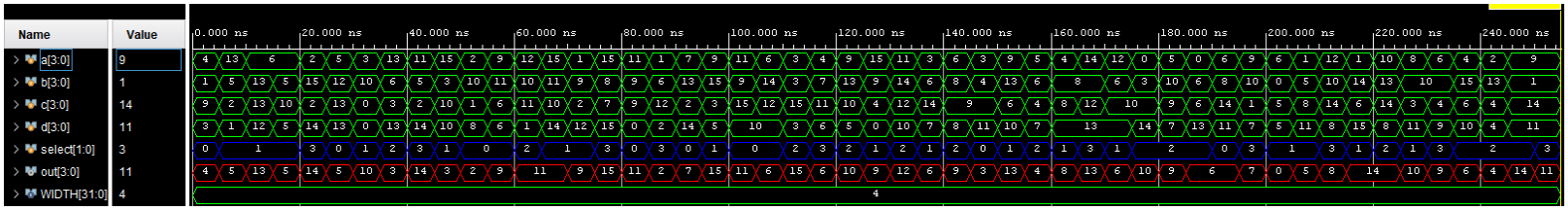

Figura 11: Simulación multiplexor 4:1, buses de 4 bits.

3.2 Resultados buses de 8 bits

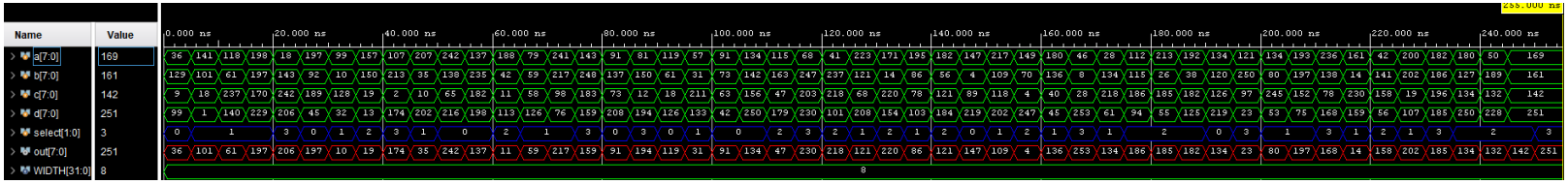

Figura 12: Simulación multiplexor 4:1, buses de 8 bits.

3.3 Resultados buses de 16 bits

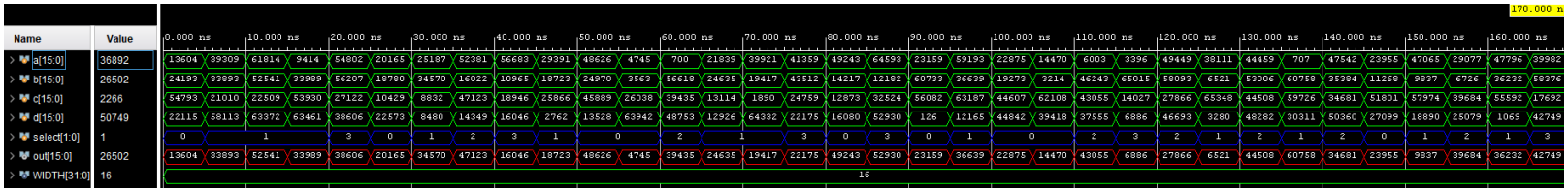

Figura 13: Simulación multiplexor 4:1, buses de 16 bits parte 1.

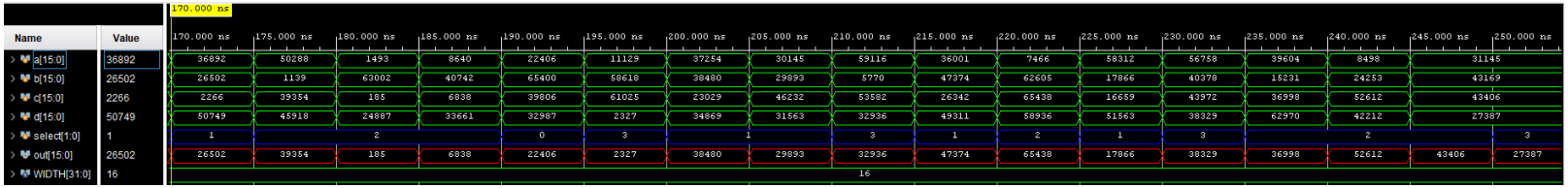

Figura 14: Simulación multiplexor 4:1, buses de 16 bits parte 2.

En la simulación se tienen 4 entradas principales A,B,C, y D. La entrada select se encarga de dejar pasar el valor de una de las cuatro entradas, estas se seleccionan de la siguiente manera:

- select=00_2 deja pasar A.
- select=01_2 deja pasar B.
- select=10_2 deja pasar C.

- select=11_2 deja pasar D.

Las pruenas se realizaron con entradas aleatorias. La entrada select también cambia de manera aleatoria.

4 Ejercicio 4

En este apartado se muestran los resultados obtenidos en la simulación del módulo de modulación de PWM y su respectivo diagrama de flujo. Además, el código utilizado para el módulo y el TB se pueden encontrar en Repositorio.

4.1 Diagrama de flujo

Se realizó un diagrama de flujo para una mayor claridad del problema.

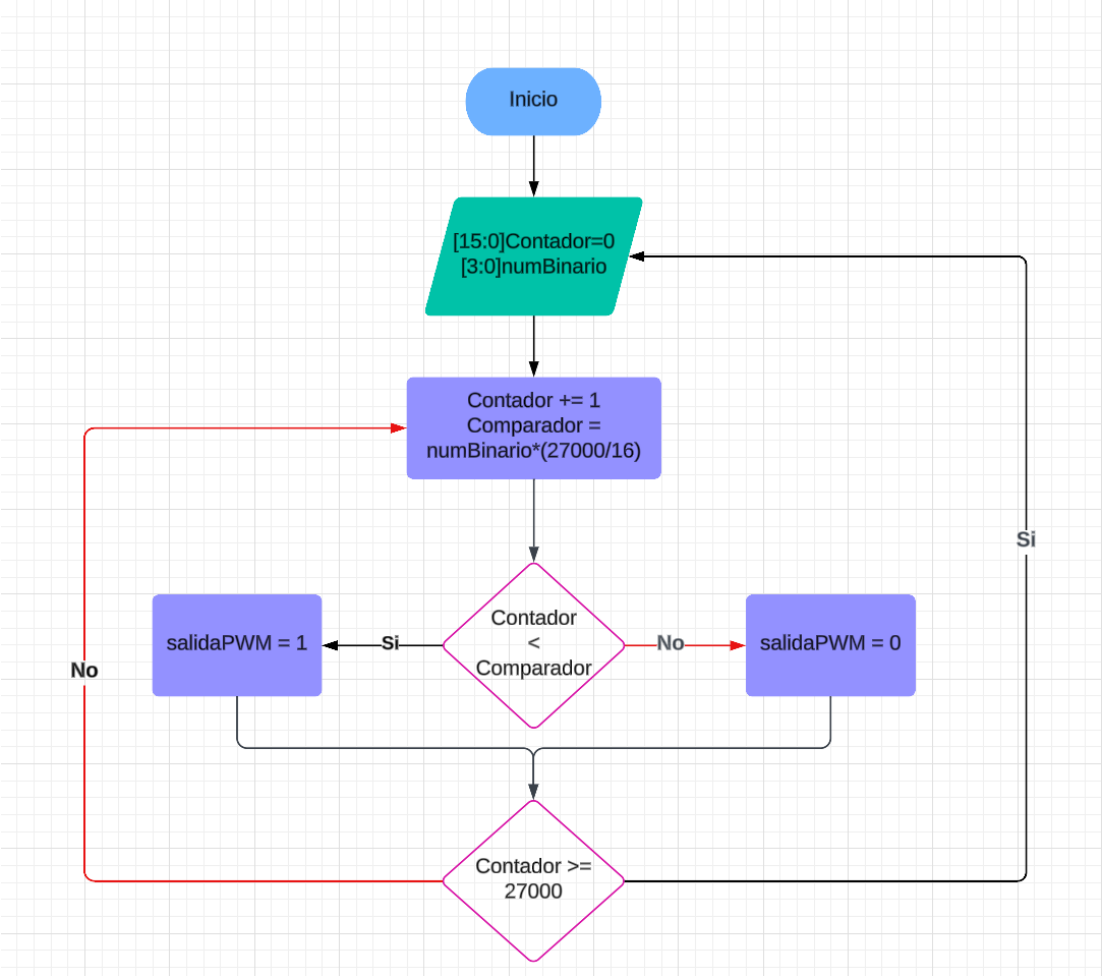

Figura 15: diagrama de flujo módulo ancho de pulso.

4.2 Resultados del funcionamiento

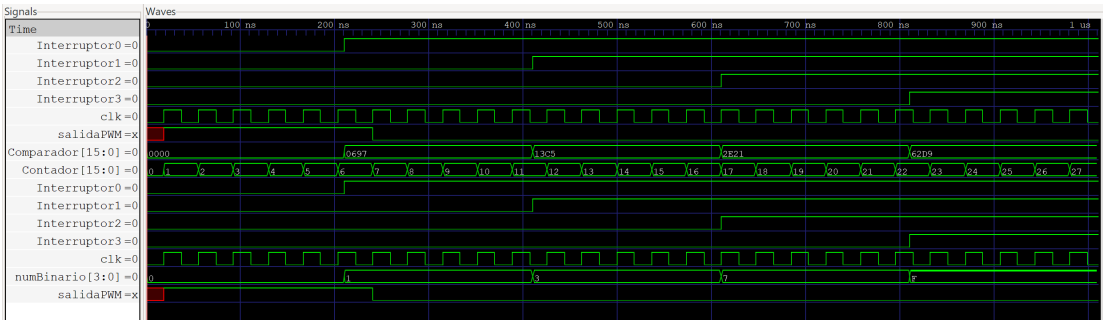

Figura 16: Simulación módulo ancho de pulso.

5 Ejercicio 5

Para este ejercicio se construyó una ALU parametrizable de n bits, todo se realizó en el lenguaje de systemverilog, la simulación realizada se trabajó con una ALU de 4bits, el código se puede encontrar en Repositorio dentro de la Carpeta Ejercicio 5, dentro del Laboratorio 1.

5.1 Resultados de la simulación

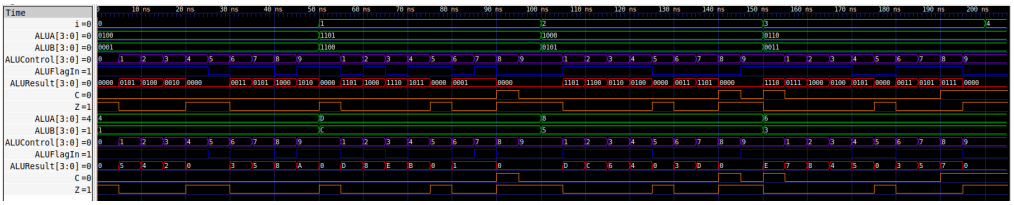

Figura 17: Simulación ALU de 4 bits.

Para entender la imagen se debe saber que el ALUResult depende de la opción elegida por el ALUControl, este elige sus funciones por medio de un número y el código para cada función es el siguiente:

- 0h - and
- 1h - or
- 2h - suma
- 3h - +1
- 4h - -1

- 5h - not
- 6h - resta
- 7h - xor
- 8h - desplazamiento a la derecha
- 9h - desplazamiento a la izquierda

5.2 Entradas

- ALUA : operando 1.
- ALUB : operando 2.
- ALUControl : selecciona la operación a realizar entre operandos.
- ALUFlagIn: acarreo de entrada para suma o resta — bit de entrada para los desplazamientos de 8h y 9h.

5.3 Salidas

La ALU además de la salida principal ALUResult posee dos sálidas más, una llamada Z y otra llamada C.

- Z : es 0 si ALUResult es 1 y 1 si ALUResult es 0.
- C : Es el último bit desplazado para las operaciones 8h y 9h.

5.4 Simulación

En 17 se puede ver que ALUA y ALUB se mantienen constantes y cambian cada 50ns, esto es para crear un orden y que cada vez que se cambien sus valores se puedan realizar todas las operaciones de la ALU. ALUFlaIn varía al inicio de cada operación para darle un componente aleatorio y poder verificar más casos.

Para ser más conveniente en una fila de la simulación se presentan los valores de ALUA, ALUB y ALUResult de manera binaria para comprobar las operaciones de desplazamiento o las compuertas lógicas de una manera más cómoda.

6 Conclusión