

ALU (Unidad Aritmética Lógica)

Trabajo Práctico Final
Circuitos Lógicos Programables

Gustavo Campero
Especialización en Sistemas Embebidos

Alcance

- Ancho configurable (W)
- Operaciones cubiertas:
 - **ADD**: suma binaria
 - **SUB**: resta binaria
 - **MUL**: parte baja del producto (W bits menos significativos)
 - **AND, OR, XOR, NOT**: operaciones lógicas bit a bit.
 - **SLL**: shift lógico a la izquierda
 - **SRL**: shift lógico a la derecha.
 - **SRA**: shift aritmético a la derecha.
 - **ROL**: rotación circular a la izquierda.
 - **ROR**: rotación circular a la derecha.

Desarrollo

Arquitectura por bloques vs ALU monolítica

Ventajas

1. **Modularidad:** facilita lectura, depuración y evolución del diseño
2. **Reutilización:** los bloques pueden reutilizarse en otros proyectos o laboratorios, y reemplazarse por versiones alternativas.
3. **Verificación** por partes: permite testbenches unitarios por operación
4. **Escalabilidad:** es directo habilitar/deshabilitar funcionalidades

Desventajas

1. **Área potencialmente mayor:** con más cantidad de bloques puede crecer el uso de LUTs/DSPs
2. **Mayor potencia dinámica:** todos los bloques conmutan siempre, aunque no se usen, generando más consumo y temperatura

Desarrollo

Arquitectura por bloques vs ALU monolítica

Ventajas

1. **Modularidad:** facilita lectura, depuración y evolución del diseño
2. **Reutilización:** los bloques pueden reutilizarse en otros proyectos o laboratorios, y reemplazarse por versiones alternativas.
3. **Verificación** por partes: permite testbenches unitarios por operación
4. **Escalabilidad:** es directo habilitar/deshabilitar funcionalidades

Desventajas

1. **Área potencialmente mayor:** con más cantidad de bloques puede crecer el uso de LUTs/DSPs
2. **Mayor potencia dinámica:** todos los bloques conmutan siempre, aunque no se usen, generando más consumo y temperatura

Desarrollo

Arquitectura por bloques vs ALU monolítica

Desventajas

1. **Área potencialmente mayor:** con más cantidad de bloques puede crecer el uso de LUTs/DSPs



Habilitación por bloque: se agrega a cada componente una señal EN codificada a partir de la operación elegida.

2. **Mayor potencia dinámica:** todos los bloques conmutan siempre, aunque no se usen, generando más consumo y temperatura

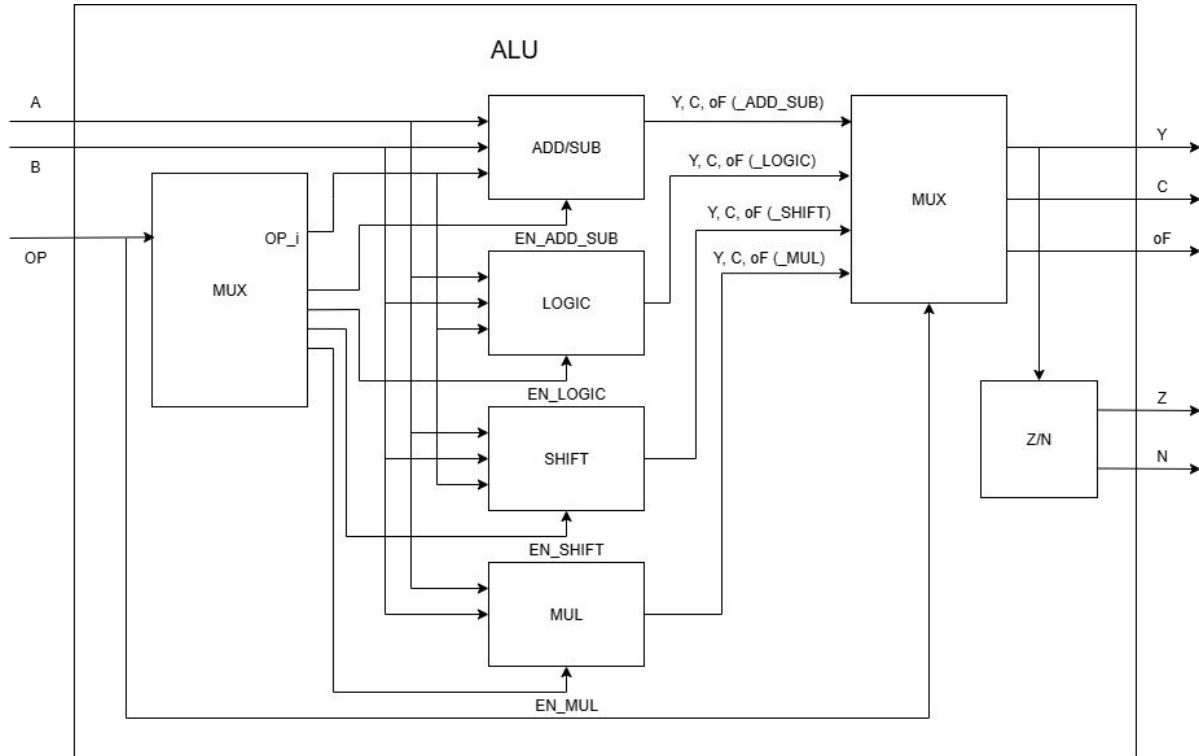


Agrupación de operaciones afines:

- Suma/resta
- Operaciones lógicas
- Desplazamientos/rotaciones

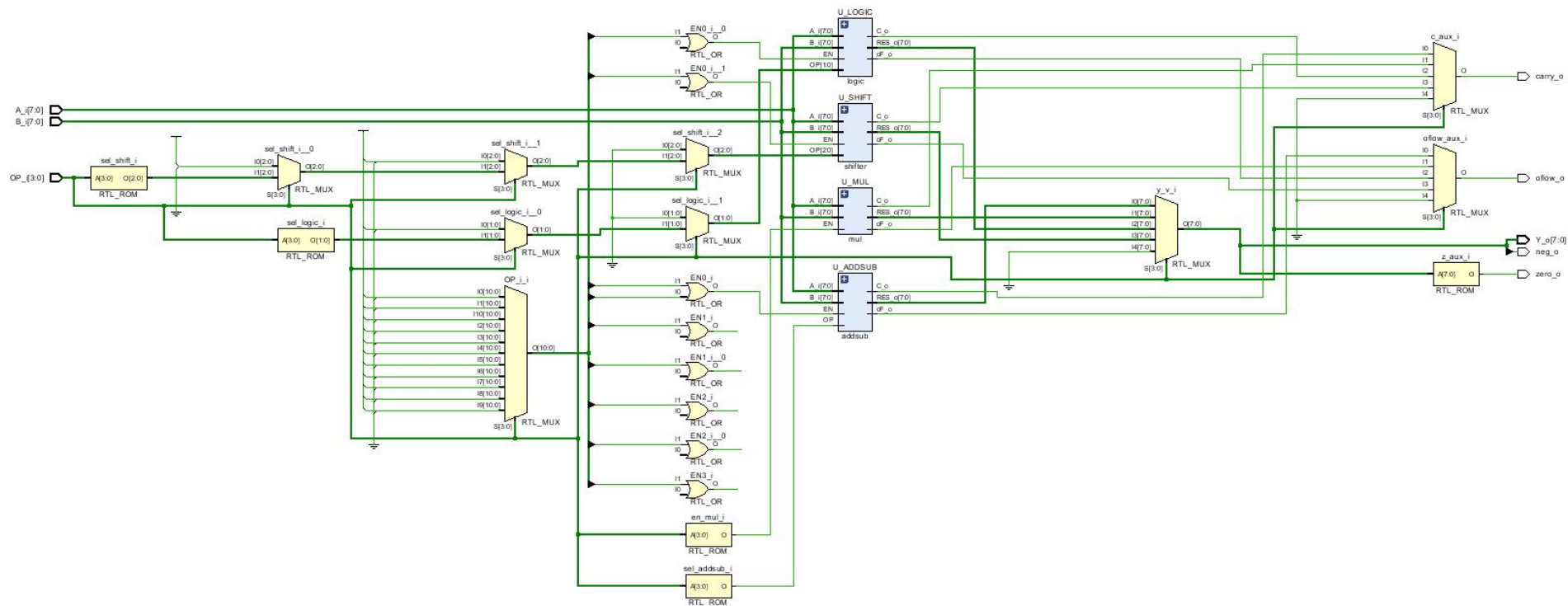
Desarrollo

Diagrama de bloques inicial



Desarrollo

Análisis RTL



Simulación

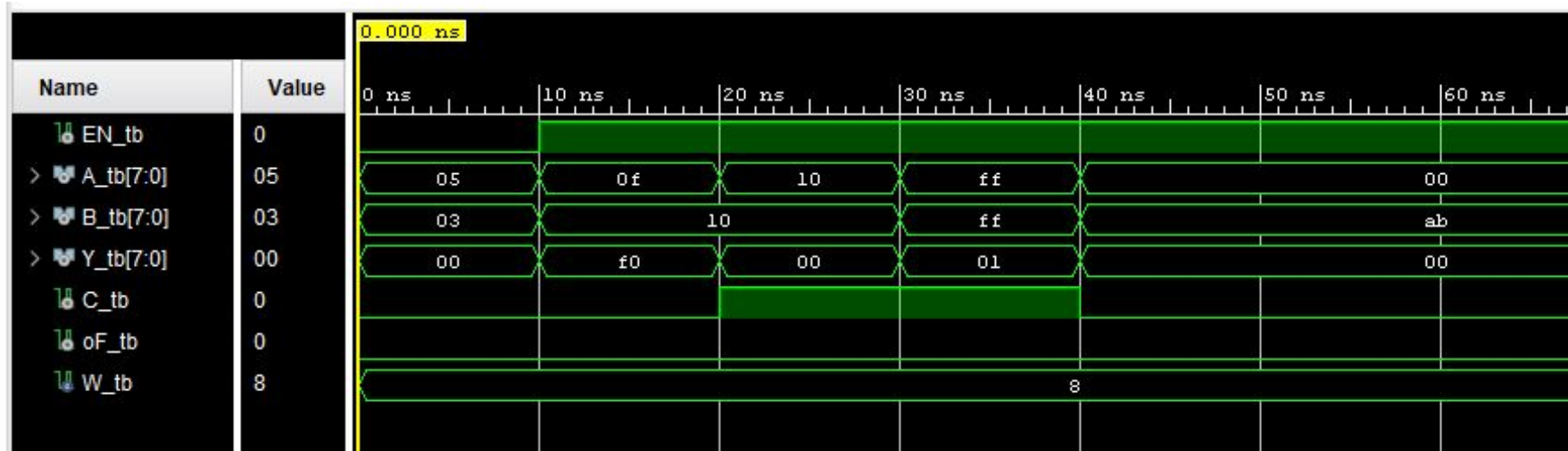
ADD/SUB

Name	Value	0 ns	10 ns	20 ns	30 ns	40 ns	50 ns	60 ns	70 ns
EN_tb	0								
OP_tb	1								
A_tb[7:0]	05	05	ff	7f	05	80	02		
B_tb[7:0]	03	03	01	03	01	03			
Y_tb[7:0]	00	00	08	00	80	02	7f	ff	
C_tb	1								
oF_tb	0								
W_tb	8	8							

Simulación

MUL

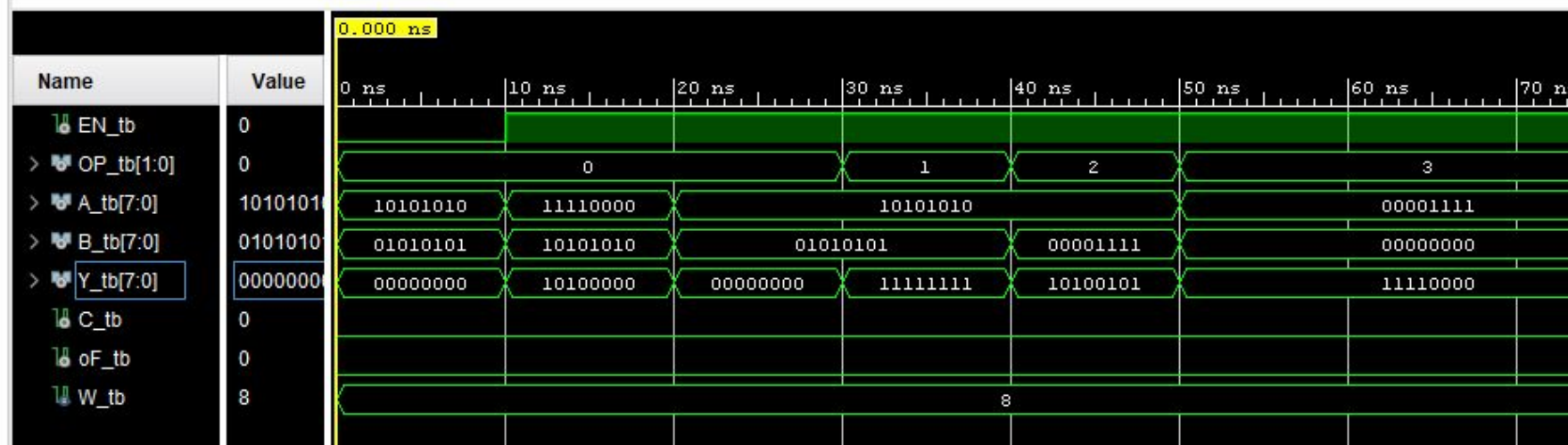
#	EN	A	B	Detalle	RES	C	oF
1	0	05	03	EN = 0	00	0	0
2	1	0F	10	$0x0F * 0x10 = 0x00F0$	F0	0	0
3	1	10	10	$0x10 * 0x10 = 0x0100$	00	1	0
4	1	FF	FF	$0xFF * 0xFF = 0xFE01$	01	1	0
5	1	00	AB	$0x00 * 0xAB = 0x0000$	00	0	0



Simulación

LOGIC

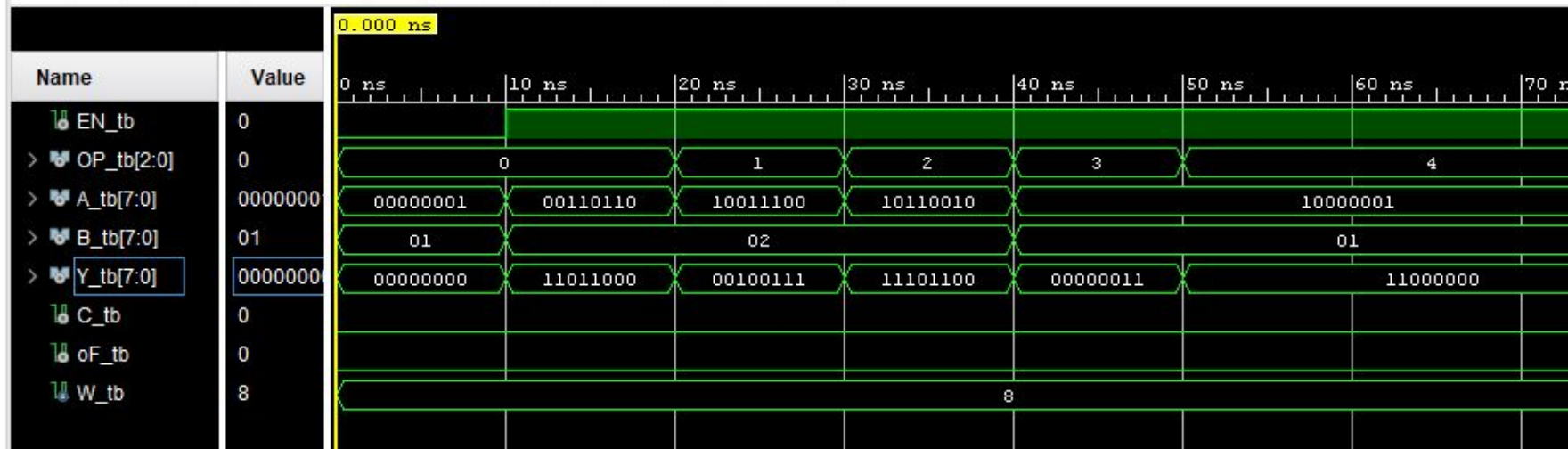
#	EN	A	B	OP	RES	C	oF
1	0	10101010	01010101	00	00000000	0	0
2	1	11110000	10101010	00	10100000	0	0
3	1	10101010	01010101	00	00000000	0	0
4	1	10101010	01010101	01	11111111	0	0
5	1	10101010	00001111	10	10100101	0	0
6	1	00001111	00000000	11	11110000	0	0



Simulación

SHIFT

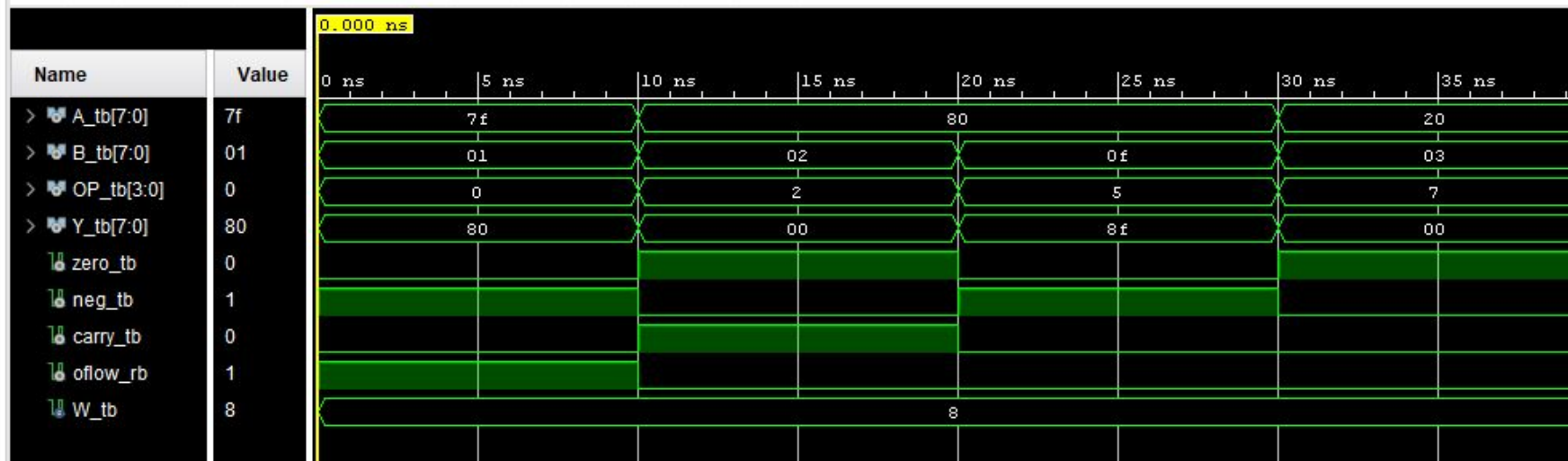
#	EN	A	B	OP	RES	C	oF
1	0	00000001	01	00	00000000	0	0
2	1	00110110	02	00	11011000	0	0
3	1	10011100	02	00	00100111	0	0
4	1	10110010	02	01	11101100	0	0
5	1	10000001	01	10	00000011	0	0
6	1	10000001	01	11	11000000	0	0



Simulación

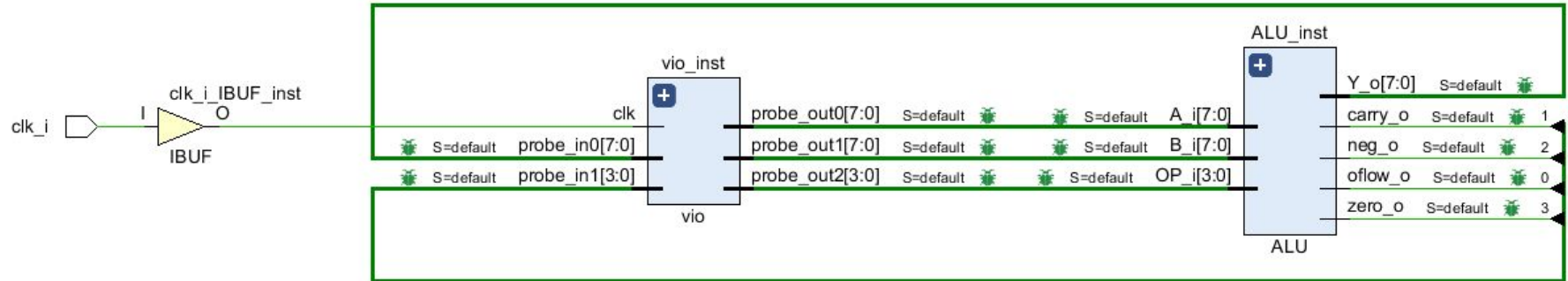
ALU

#	A	B	OP	Y	Z	N	C	oF
1	7F	01	0000	80	0	1	0	1
2	80	02	0010	00	1	0	1	0
3	80	0F	0101	8F	0	1	0	0
4	20	03	0111	00	1	0	0	0



Programación

VIO



Programación

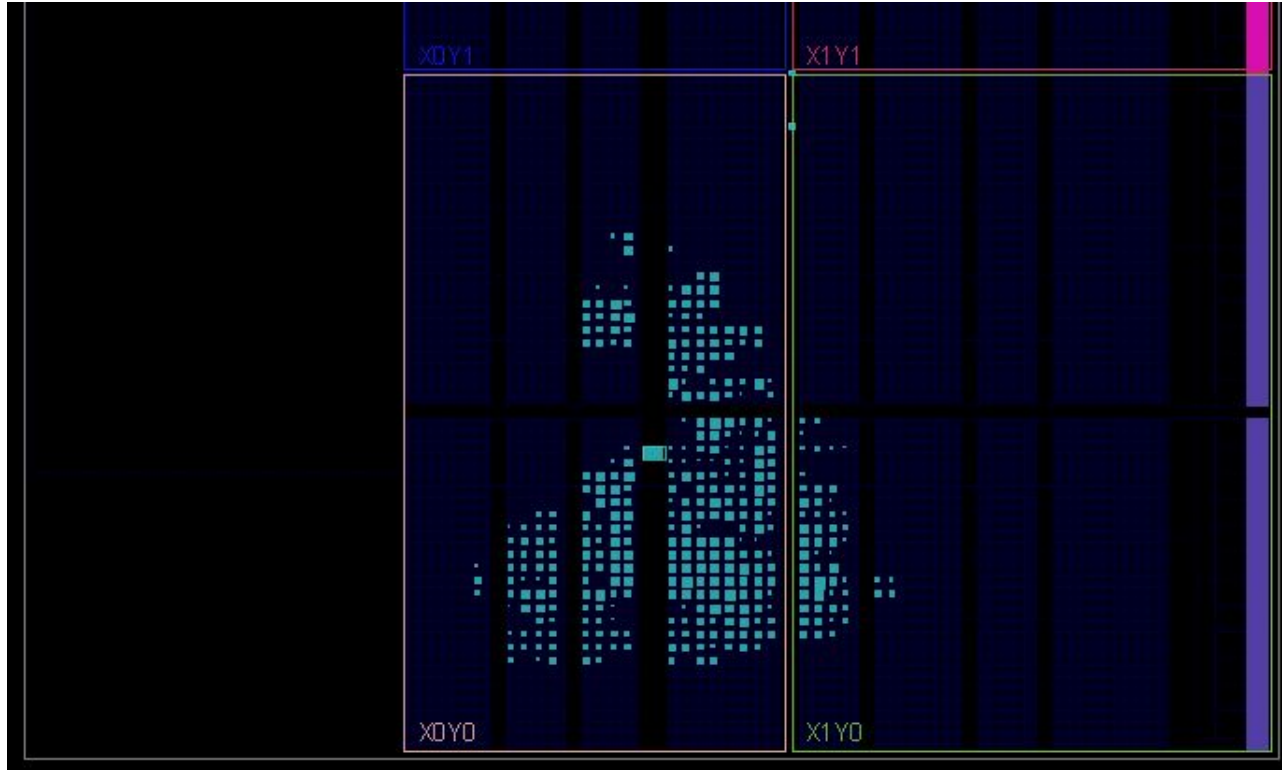
Tabla de uso de recursos de la FPGA

Name	Slice LUTs (17600)	Slice Registers (35200)	Slice (4400)	LUT as Logic (17600)	LUT as Memory (6000)	LUT Flip Flop Pairs (17600)	Bonded IOB (100)	BUFGCTRL (32)	BSCANE2 (4)
▼ N ALU_VIO	874	1086	366	850	24	451	1	2	1
> I ALU_inst (ALU)	244	0	73	244	0	0	0	0	0
> I dbg_hub (dbg_hub)	463	723	222	439	24	295	0	1	1
> I vio_inst (vio)	167	363	92	167	0	135	0	0	0

Resource	Utilization	Available	Utilization %
LUT	874	17600	4.97
LUTRAM	24	6000	0.40
FF	1086	35200	3.09
IO	1	100	1.00

Programación

Tabla de uso de recursos de la FPGA































Programación

hw_vio_1					
Name	Value	Activity	Direction	VIO	
> probe_A[7:0]	[H] 7F		Output	hw_vio_1	
> probe_B[7:0]	[H] 01		Output	hw_vio_1	
> probe_OP[3:0]	[H] 0		Output	hw_vio_1	
> probe_Y[7:0]	[H] 80		Input	hw_vio_1	
▼ probe_flags[3:0]	[H] 5		Input	hw_vio_1	
└─ probe_flags[3]	0		Input	hw_vio_1	
└─ probe_flags[2]	1		Input	hw_vio_1	
└─ probe_flags[1]	0		Input	hw_vio_1	
└─ probe_flags[0]	1		Input	hw_vio_1	

hw_vio_1					
Name	Value	Activity	Direction	VIO	
> probe_A[7:0]	[H] 80		Output	hw_vio_1	
> probe_B[7:0]	[H] 02		Output	hw_vio_1	
> probe_OP[3:0]	[B] 0010		Output	hw_vio_1	
> probe_Y[7:0]	[H] 00		Input	hw_vio_1	
▼ probe_flags[3:0]	[H] A		Input	hw_vio_1	
└─ probe_flags[3]	1		Input	hw_vio_1	
└─ probe_flags[2]	0		Input	hw_vio_1	
└─ probe_flags[1]	1		Input	hw_vio_1	
└─ probe_flags[0]	0		Input	hw_vio_1	

Programación

hw_vio_1					
    					
Name	Value	Activity	Direction	VIO	
>  probe_A[7:0]	[H] 80		Output	hw_vio_1	
>  probe_B[7:0]	[H] 0F		Output	hw_vio_1	
>  probe_OP[3:0]	[B] 0101		Output	hw_vio_1	
>  probe_Y[7:0]	[H] 8F		Input	hw_vio_1	
√  probe_flags[3:0]	[H] 4		Input	hw_vio_1	
└─  probe_flags[3]	0		Input	hw_vio_1	
└─  probe_flags[2]	1		Input	hw_vio_1	
└─  probe_flags[1]	0		Input	hw_vio_1	
└─  probe_flags[0]	0		Input	hw_vio_1	

hw_vio_1					
    					
Name	Value	Activity	Direction	VIO	
>  probe_A[7:0]	[B] 0001_0000		Output	hw_vio_1	
>  probe_B[7:0]	[H] 03		Output	hw_vio_1	
>  probe_OP[3:0]	[B] 0111		Output	hw_vio_1	
>  probe_Y[7:0]	[B] 1000_0000	↑	Input	hw_vio_1	
√  probe_flags[3:0]	[H] 4	↕	Input	hw_vio_1	
└─  probe_flags[3]	0	↓	Input	hw_vio_1	
└─  probe_flags[2]	1	↑	Input	hw_vio_1	
└─  probe_flags[1]	0		Input	hw_vio_1	
└─  probe_flags[0]	0		Input	hw_vio_1	

Gracias