Arquitecturas de Computadoras

Trabajo Práctico N°1
ALU

Profesores:

- Ing. Martin Pereyra
- Ing. Santiago Rodriguez
- Ing. Martin Alonso (Teórico)

Alumna:

- Robles Karen Yesica

2024

ÍNDICE

1. Introducción

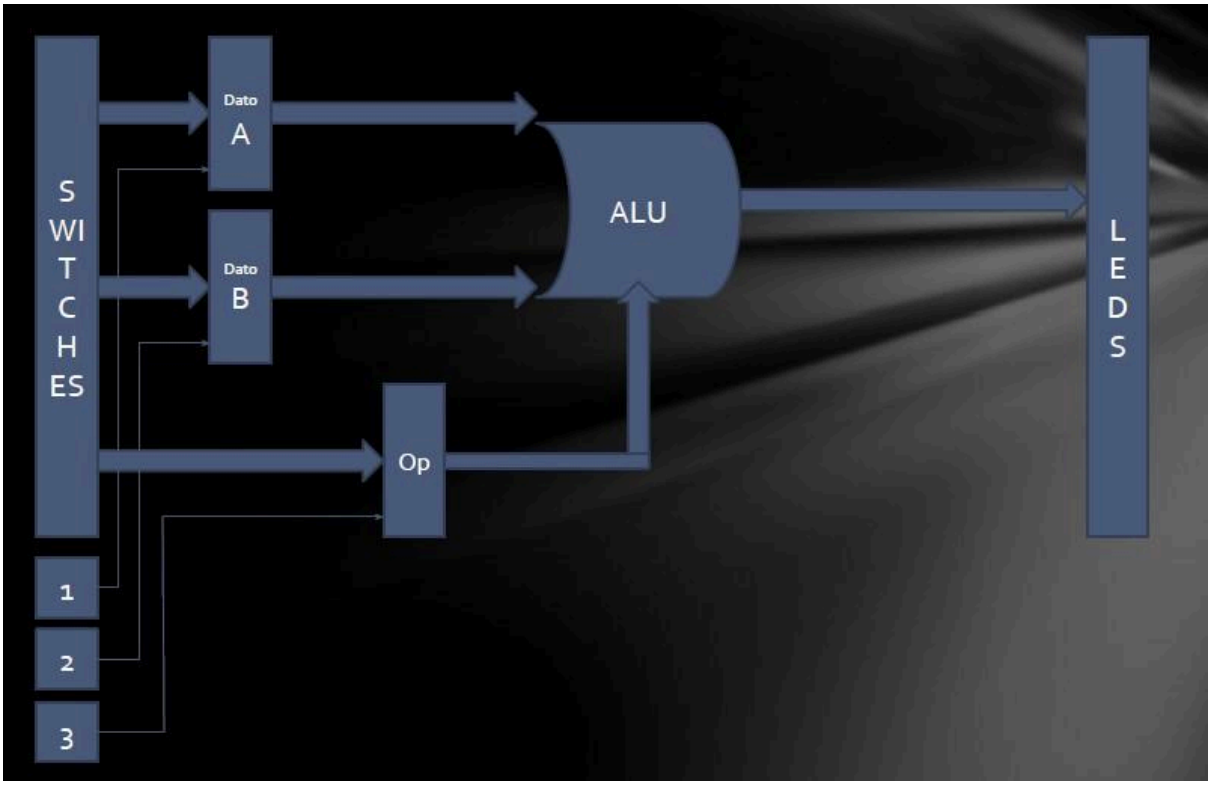
En el diseño digital, una Unidad Aritmético Lógica (ALU) es uno de los componentes más importantes dentro de un procesador, ya que permite realizar operaciones matemáticas y lógicas sobre los datos. Las ALUs son bloques funcionales esenciales en una gran variedad de sistemas embebidos y circuitos integrados, incluidos los procesadores, controladores y sistemas de señal digital. El objetivo de este proyecto es implementar una ALU parametrizable en una FPGA Basys 3 utilizando el software Vivado utilizando el lenguaje Verilog, verificando su funcionamiento mediante simulación y pruebas automáticas con testbenches

2. Objetivo

El objetivo principal de este proyecto es diseñar, implementar y validar una ALU en una FPGA que sea parametrizable en términos del ancho de datos. Esto permitirá su reutilización en proyectos posteriores con diferentes tamaños de datos. Se validará la funcionalidad de la ALU mediante testbenches que generen entradas aleatorias y realicen la verificación automática de los resultados, utilizando las herramientas de simulación y análisis de tiempos de Vivado.

3. Desarrollo

3.1. Diagrama de bloques funcional



- Los datos de entrada (A y B) y el código de operación (Op) son ingresados mediante un mismo conjunto de interruptores.
- Primero, se ingresa el valor de A presionando un botón específico para confirmar su entrada.
- Luego, se ingresa el valor de B y se presiona otro botón para confirmar este segundo dato.
- A continuación, se selecciona la operación deseada mediante otros interruptores y se presiona otro botón para ejecutar la operación.
- Una vez confirmados los operandos y la operación, el resultado se muestra en los LEDs, proporcionando una salida visual
- Además, el sistema incluye un botón de reset, que permite restablecer el sistema y limpiar los valores ingresados para realizar nuevas operaciones.
- En el caso de operaciones aritméticas, además del resultado, se indica si hubo o no overflow a través de un LED adicional que se enciende si se detecta esta condición.
- El sistema también incluye un botón de reset, que permite restablecer el sistema y limpiar los valores ingresados para realizar nuevas operaciones.

3.2. Módulos Implementados

3.2.1.Módulo ALU

Este módulo implementa una Unidad Aritmético Lógica (ALU) parametrizable que puede realizar diversas operaciones aritméticas y lógicas

.

Parámetros

- `NB_DATA`: Número de bits para los datos de entrada (por defecto 8).
- `NB_OP`: Número de bits para el código de operación (por defecto 6).

Entradas

- `i_dataA`: Primer operando (bus de datos).
- `i_dataB`: Segundo operando (bus de datos).
- `i_op`: Código de operación que determina qué operación se va a realizar.

Salidas

- `o_result`: Resultado de la operación.
- `o_overflow`: Señal que indica un desbordamiento (overflow) en las operaciones aritméticas.

Operaciones Soportadas

- Aritméticas:

- Suma (`ADD`): Código 6'b100000.
- Resta (`SUB`): Código 6'b100010.

- Lógicas:

- AND (`AND`): Código 6'b100100.
- OR (`OR`): Código 6'b100101.
- XOR (`XOR`): Código 6'b100110.
- NOR (`NOR`): Código 6'b100111.

- Desplazamientos:

- Desplazamiento a la derecha aritmético (`SRA`): Código 6'b000011.
- Desplazamiento a la derecha lógico (`SRL`): Código 6'b000010.

Lógica Combinacional:

- Implementé la lógica combinacional necesaria para las operaciones de la ALU.
- Utilicé un bloque **`always @(*)`** para asegurar que la lista de sensibilidad cubriera todas las variables relevantes.
- Todas las asignaciones dentro de este bloque fueron realizadas usando asignaciones bloqueantes **(=)**, garantizando que la lógica se evaluará en función de los cambios en las entradas.

3.2.2. Módulo ADD_SUB

Este módulo implementa la lógica para realizar operaciones de suma y resta de dos operandos con signo. Está diseñado para trabajar con un tamaño de dato configurable.

Parámetros
- `NB_DATA`: Número de bits para los operandos (por defecto 8).

Entradas
- `i_dataA`: Primer operando (con signo, rango de -128 a 127 para 8 bits).
- `i_dataB`: Segundo operando (con signo).
- `ctrl`: Control de operación; `0` para suma y `1` para resta.

Salidas
- `o_result`: Resultado de la operación, también con signo.
- `o_overflow`: Señal que indica si se ha producido un desbordamiento (overflow).

3.2.3. Módulo TOP

El módulo `TOP` actúa como la interfaz principal para la ALU, permitiendo la entrada de datos y operaciones mediante interruptores y botones, así como la visualización de resultados y señales de desbordamiento en LEDs.

Parámetros
- `NB_DATA`: Número de bits para los datos (por defecto 8).
- `NB_OP`: Número de bits para el código de operación (por defecto 6).

Entradas
- `i_dataSw`: Interruptores de entrada para los datos (bus de datos).
- `i_opSw`: Interruptores de entrada para seleccionar la operación.
- `i_btnA`: Botón para cargar `dataA`.
- `i_btnB`: Botón para cargar `dataB`.
- `i_btnO`: Botón para cargar la operación.
- `i_clk`: Señal de reloj.
- `i_reset`: Señal de reinicio.

Salidas
- `o_resultLed`: Resultado de la operación, enviado a LEDs.
- `o_overflowLed`: Señal de desbordamiento, enviada a un LED.

Lógica Secuencial:

- En el módulo top, utilicé un bloque **`always @(posedge clock)`** para modelar la lógica secuencial.

- Utilice asignaciones no bloqueantes **(<=)**, asegurando que todas las asignaciones dentro del bloque se ejecutaran en paralelo y de forma adecuada al flanco de subida del reloj.

3.2.4. Módulo de Prueba ALU_TB

El módulo `ALU_TB` es un banco de pruebas diseñado para validar el correcto funcionamiento de la ALU. Este módulo realiza múltiples pruebas automatizadas para cada operación soportada por la ALU, asegurando que los resultados sean correctos.

Parámetros
- `NB_DATA`: Número de bits para los datos (por defecto 8).
- `NB_OP`: Número de bits para el código de operación (por defecto 6).
- `N_TEST`: Número de pruebas a realizar para cada operación (por defecto 4).

Entradas
- `i_dataA`: Primer operando (entrada de datos).
- `i_dataB`: Segundo operando (entrada de datos).
- `i_op`: Código de operación a ejecutar.

Salidas
- `o_result`: Resultado de la operación realizada por la ALU.
- `o_overflow`: Señal que indica si se produjo un desbordamiento.

3.2.5. Módulo de Prueba TOP_TB

El módulo `TOP_TB` es un banco de pruebas para el módulo `TOP`, que implementa una ALU parametrizable. Este módulo realiza pruebas automatizadas para asegurar que el comportamiento del diseño sea el esperado.

Parámetros
- **NB_DATA**: Número de bits para los datos (por defecto 8).
- **NB_OP**: Número de bits para el código de operación (por defecto 6).
- **N_TEST**: Número de pruebas a realizar para cada operación (por defecto 3).

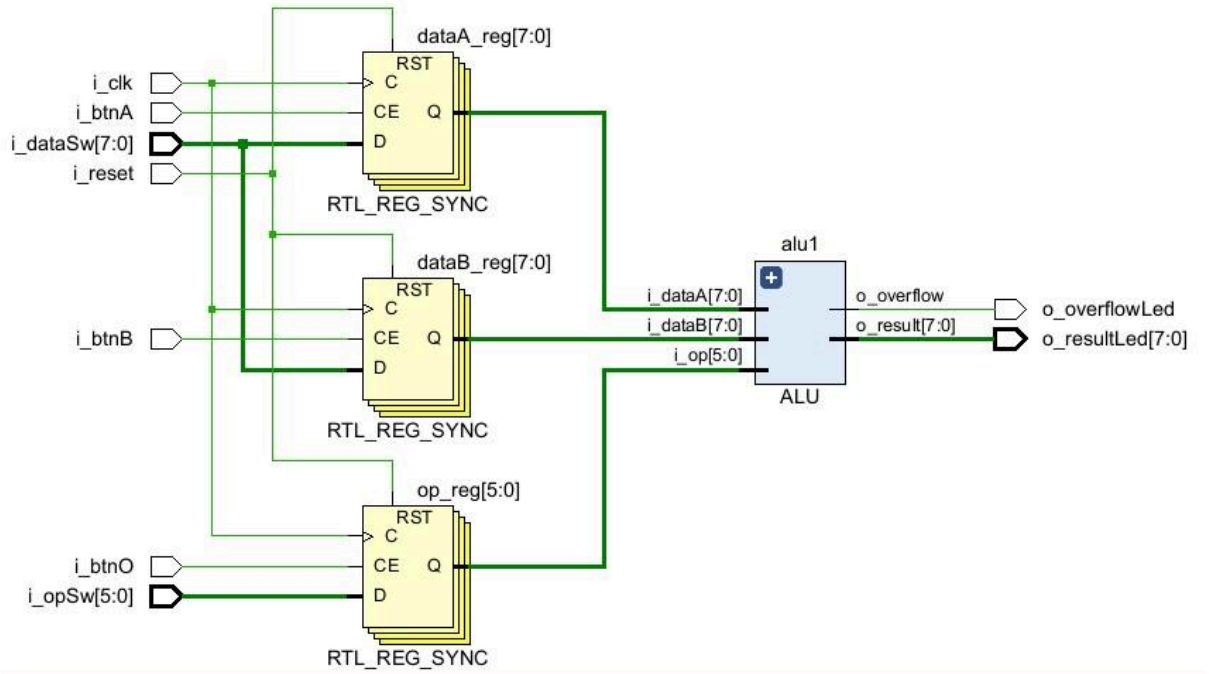
Entradas
- **i_dataSw**: Valor de entrada de los switches.
- **i_opSw**: Código de operación seleccionada desde los switches.
- **i_btnA**: Botón para cargar el primer operando.
- **i_btnB**: Botón para cargar el segundo operando.
- **i_btnO**: Botón para cargar la operación.
- **i_clk**: Señal de reloj.

- **i_reset**: Señal de reset.

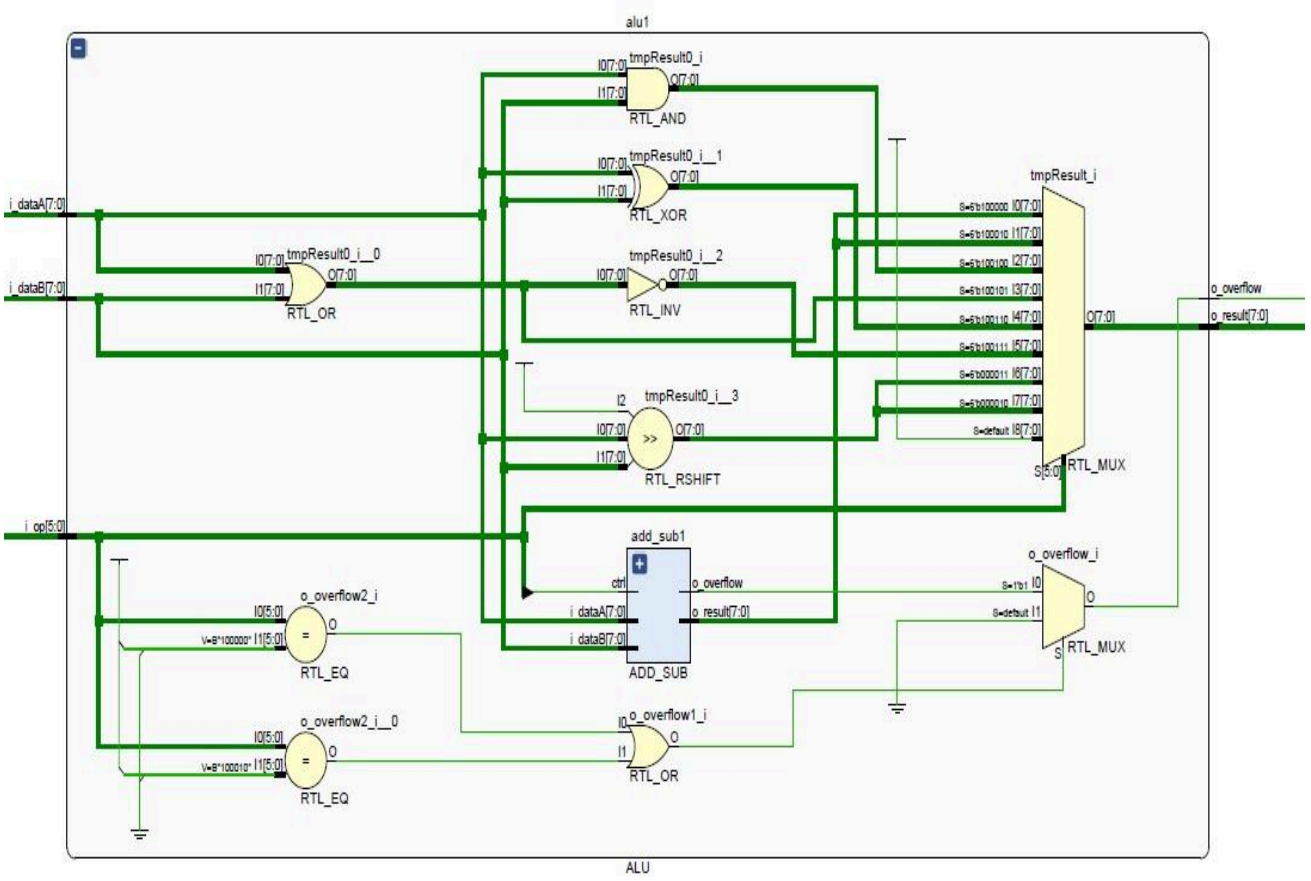
Salidas
- **o_resultLed**: Resultado de la operación mostrada en LEDs.
- **o_overflowLed**: Señal de overflow mostrada en un LED.

4. Diagrama de Bloques de la arquitectura

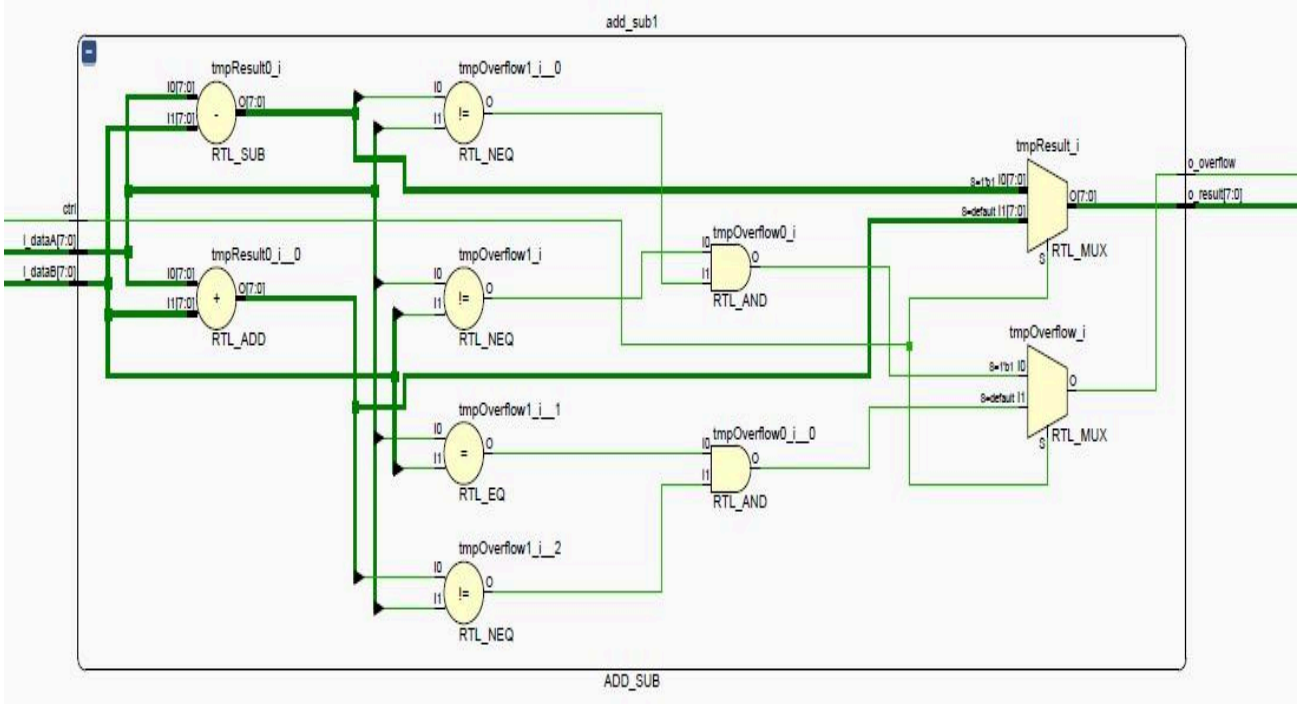
A continuación, se muestra el diagrama RTL (Register Transfer Level) generado en Vivado, que representa el diseño de la ALU con registros de entrada y salida, así como el flujo de señales entre los bloques lógicos.



ALU:


alu1
tmpResult0_i
RTL_AND
tmpResult0_i__1
RTL_XOR
tmpResult0_i__0
RTL_OR
tmpResult0_i__2
RTL_INV
tmpResult0_i__3
RTL_RSHIFT
tmpResult_i
RTL_MUX
i_dataA[7:0]
i_dataB[7:0]
i_op[5:0]
o_overflow
o_result[7:0]
add_sub1
ADD_SUB
o_overflow2_i
RTL_EQ
o_overflow2_i__0
RTL_EQ
o_overflow1_i
RTL_OR
o_overflow_i
RTL_MUX
ALU

ADD_SUB:


add_sub1
tmpResult0_i
RTL_SUB
tmpOverflow1_i__0
RTL_NEQ
tmpResult_i
RTL_MUX
ctrl
i_dataA[7:0]
i_dataB[7:0]
tmpResult0_i__0
RTL_ADD
tmpOverflow1_i
RTL_NEQ
tmpOverflow0_i
RTL_AND
tmpOverflow_i
RTL_MUX
tmpOverflow1_i__1
RTL_EQ
tmpOverflow0_i__0
RTL_AND
tmpOverflow1_i__2
RTL_NEQ
o_overflow
o_result[7:0]
ADD_SUB

5. Simulación

En la simulación del diseño digital que se presenta, se ha llevado a cabo un análisis exhaustivo de las señales involucradas con el objetivo de verificar el comportamiento esperado del circuito bajo prueba. A continuación, se muestra una imagen generada por el software Vivado, correspondiente a la salida del módulo TOP del diseño. Este módulo representa la jerarquía más alta en el circuito, integrando diversos componentes que trabajan conjuntamente para ejecutar las operaciones requeridas.

- A lo largo del tiempo (medido en nanosegundos), se pueden observar los cambios en las señales cuando se ingresan los operandos, se selecciona la operación y se generan los resultados.
- En el momento en que las señales de los botones se activan, los datos de los switches son almacenados en los registros correspondientes y se realiza la operación.
- La salida de los LEDs y la indicación de **overflow** se actualizan en respuesta a los valores de entrada y la operación realizada por la ALU.

i_dataSw[7:0]: Muestra los valores de los switches de entrada de datos en el bus de 8 bits. Estos valores representan los operandos **A** y **B**.

i_opSw[5:0]: Es el bus de switches que selecciona la operación a realizar en la ALU. Los valores cambian cuando se selecciona una operación específica.

i_btnA, i_btnB, i_btnO: Estas señales representan los botones para confirmar las entradas **A**, **B**, y la operación seleccionada. Se observa cómo los botones se activan (de 0 a 1) en momentos específicos para registrar los valores.

i_clk: Señal de reloj que sincroniza las operaciones de los registros y la ALU. Es un ciclo de reloj estable que controla el momento de actualización de los registros.

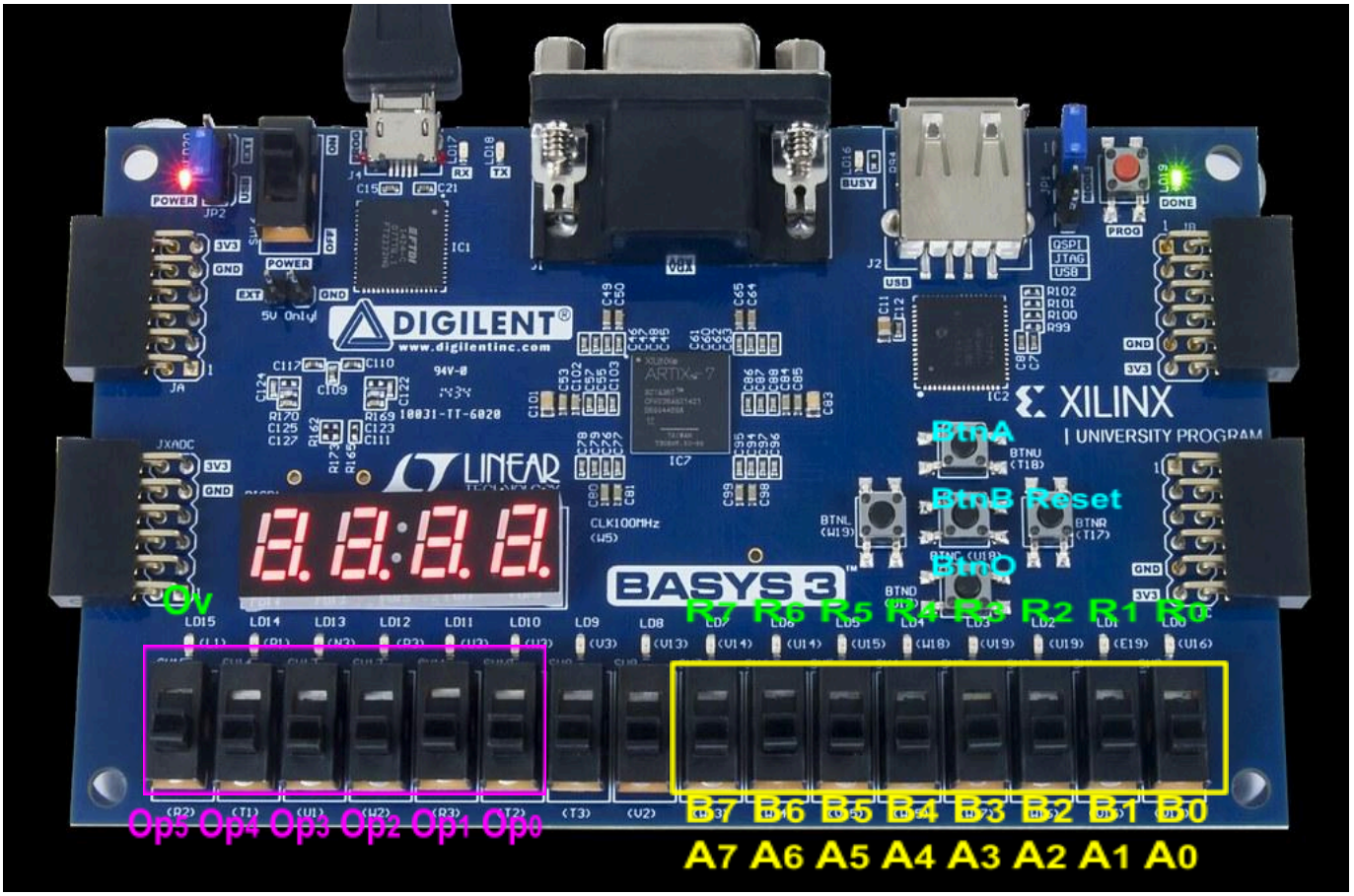
i_reset: Señal de reinicio. Aquí se ve en estado bajo, lo que indica que el sistema no está en proceso de reinicio y está permitiendo el flujo normal de datos.

o_resultLed[7:0]: Señal de salida que refleja el resultado de la operación en los LEDs. En la simulación, se puede observar cómo los valores del resultado cambian en función de los operandos y la operación seleccionada.

o_overflowLed: Indica si ocurrió un desbordamiento (overflow) durante una operación aritmética. El valor cambia de acuerdo con las operaciones realizadas, mostrando un "1" cuando ocurre overflow.

Esta simulación permite validar que el diseño de la ALU funciona correctamente en cuanto a tiempos de reloj, entradas y salidas, y el manejo de condiciones especiales como el desbordamiento.

6. Anexo


DIGILENT
www.digilentinc.com
XILINX
UNIVERSITY PROGRAM
LINEAR
BASYS 3
BtnA
BtnB
Reset
BtnC
0V
R7 R6 R5 R4 R3 R2 R1 R0
B7 B6 B5 B4 B3 B2 B1 B0
A7 A6 A5 A4 A3 A2 A1 A0
Op5 Op4 Op3 Op2 Op1 Op0