TRABAJO PRÁCTICO N °1

ALU

Docentes:
Rodriguez Santiago
Pereyra Martin

Alumnos:
Saul Emilio Muñoz
CRISTIAN DAVID VELAZQUEZ

INTRODUCCIÓN

Este trabajo práctico tiene por objeto implementar en una FPGA una ALU. La Alu es parametrizable (bus de datos) para poder ser utilizada posteriormente en el trabajo final. Además se valida el funcionamiento por medio de un Test Bench y se simula el diseño utilizando herramientas de simulación incluidas en vivado. En el test bench se realiza un conjunto de operaciones, se chequean los resultados con condiciones las cuales imprimen el resultado e indican si este es correcto o no.

repositorio: https://github.com/CristianVelazquez/arqui-2022.git

DESARROLLO

Descripción General del trabajo

Se tiene una ALU como único módulo, la cual tiene un funcionamiento combinacional. En esta se realizan las operaciones aritméticas solicitadas, las cuales son: ADD, SUB, AND, OR, XOR, SRA, SRL Y NOR. En la imagen se puede observar el código correspondiente a cada operación.

Operación	Código
ADD	100000
SUB	100010
AND	100100
OR	100101
XOR	100110
SRA	000011
SRL	000010
NOR	100111

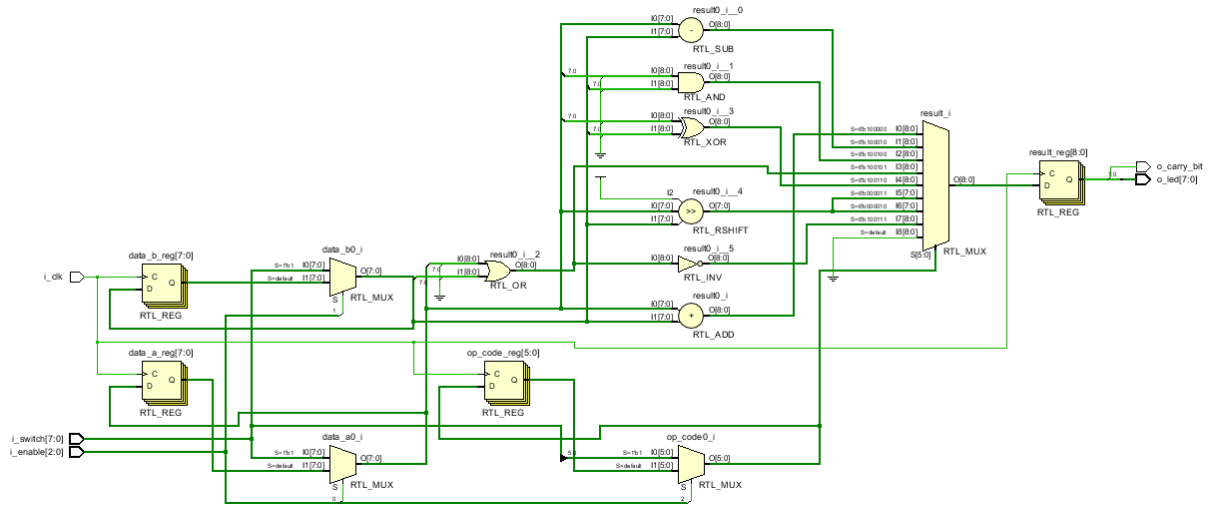
Las entradas del módulo:

- clock (reloj).
- enable, bus de tres bits, controla las acciones
- switch, bus de ocho bits, especifica una operación o el valor del dato_a o del dato_b

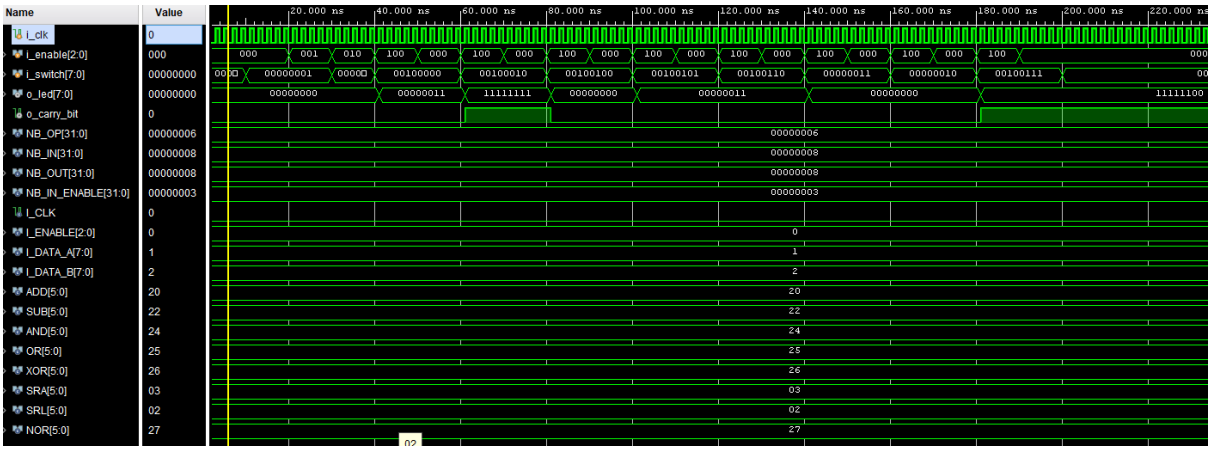
La salida del módulo:

- leds, un bus de 9 bits (el noveno bit indica el carry), especifican el resultado de la operación o el valor de las variables dato_a o dato_b.

Esquema del análisis RTL



Simulación de la Alu



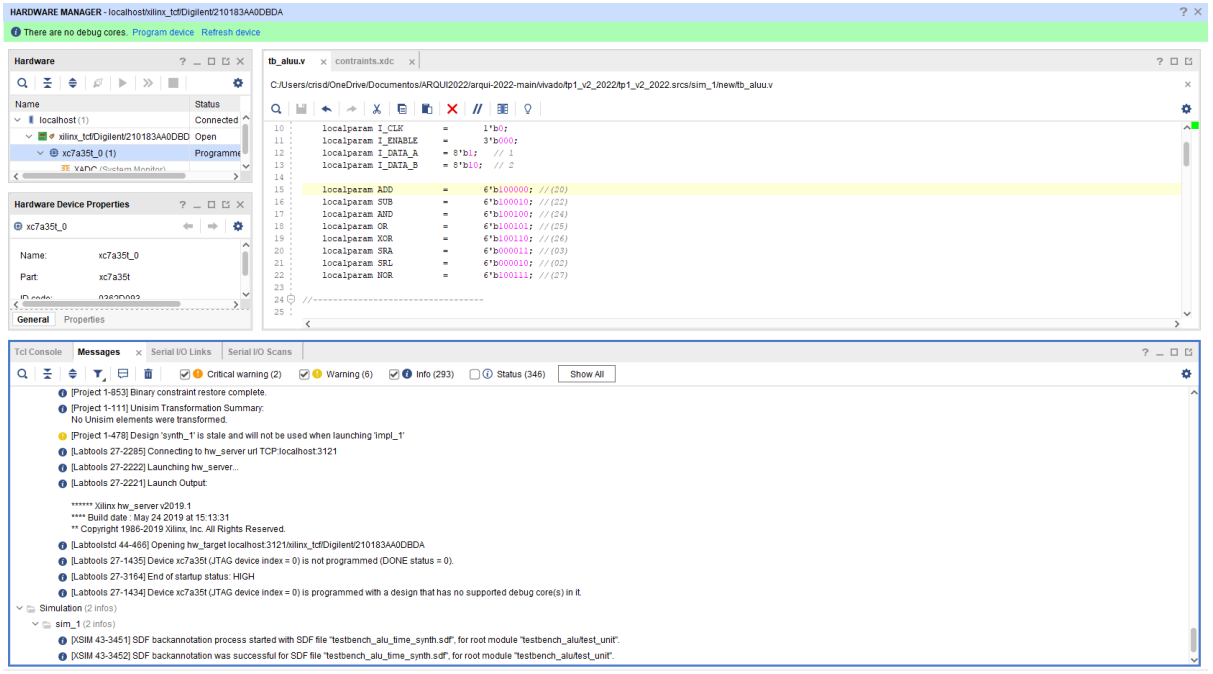
Se asigna al dato_a el valor de 1, al dato_b el valor de 2

Esta es la salida en consola:

```
############################################################
ADD code operation = 00100000 20 (hexa)
ADD result = 00000011
test operation ADD correct!
############################################################
SUB code operation = 00100010 22 (hexa)
SUB result = 11111111
test operation SUB correct!
############################################################
AND code operation = 00100100 24 (hexa)
AND result = 00000000
```

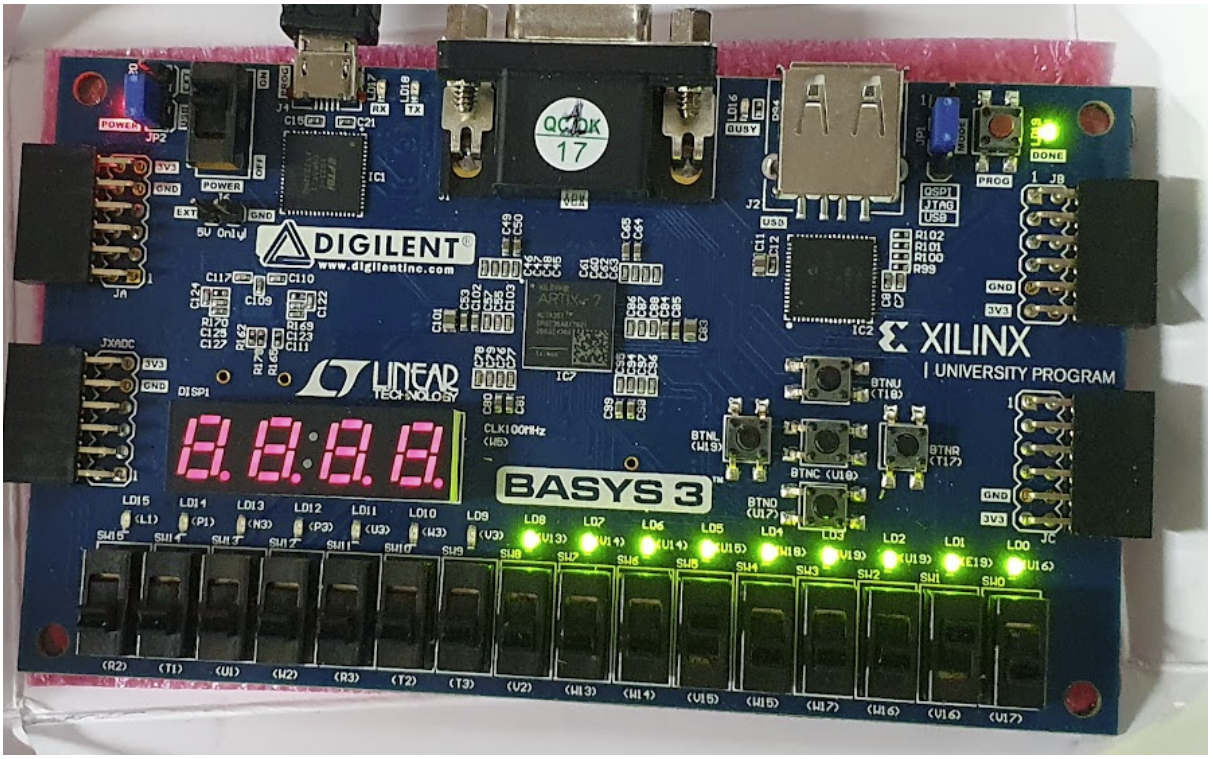
test operation AND correct!
##
OR code operation = 00100101 25 (hexa)
OR result = 00000011 (decimal)
test operation OR correct!
##
XOR code operation = 00100110 26 (hexa)
XOR result = 00000011 (decimal)
test operation XOR correct!
##
SRA code operation = 00000011 03 (hexa)
SRA result = 00000000
test operation SRA correct!
##
SRL code operation = 00000010 02 (hexa)
SRL result = 00000000 (decimal)
test operation SRL correct!
##
NOR code operation = 00100111 27 (hexa)
NOR result = 11111100 (decimal)
test operation SRL correct!
##

Implementación y debug



Se puede observar en la captura que el programa se escribió con éxito en la placa y que no hubo bugs.

Captura de operación de resta, se puede ver que todos los leds prendidos porque el resultado es un número negativo y el led LD8 representa el carry
La resta realizada es 1-2 = ff (en complemento a2)



CONCLUSIÓN

Durante el desarrollo del trabajo práctico se fue comprendiendo lo visto en clase ya que además de realizar los módulos, se hicieron los test. Con los test se pudieron corregir algunos problemas al simular porque podíamos ver en forma detallada con los tiempos que valor lógico tienen los botones, el valor en hexadecimal en los registros, el código de la operación y la salida resultante. El análisis RTL también nos sirvió para ver de manera gráfica cómo iba tomando forma nuestro sistema y pudimos identificar visualmente cómo está integrado el sistema.
Al momento de realizar la escritura del código en la placa no resultó ser tan complicado como parecía. En internet hay mucha información sobre el mapeo de los pines y en la en la Basys 3 están todos los pines indicados por lo que solo tuvimos que revisar que los pines sean los que queremos usar en los constraints.