TRABAJO PRÁCTICO N °1

ALU

Docentes:

Rodriguez Santiago

Pereyra Martin

Alumnos:

Sosa Octavio

Velazquez Cristian

INTRODUCCIÓN

Se realizó este trabajo práctico para implementar en 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 Test Bench y se simula el diseño utilizando herramientas de simulación de vivado incluyendo análisis de tiempo. El test bench incluye la generación de entradas aleatorias y código de chequeo automático.

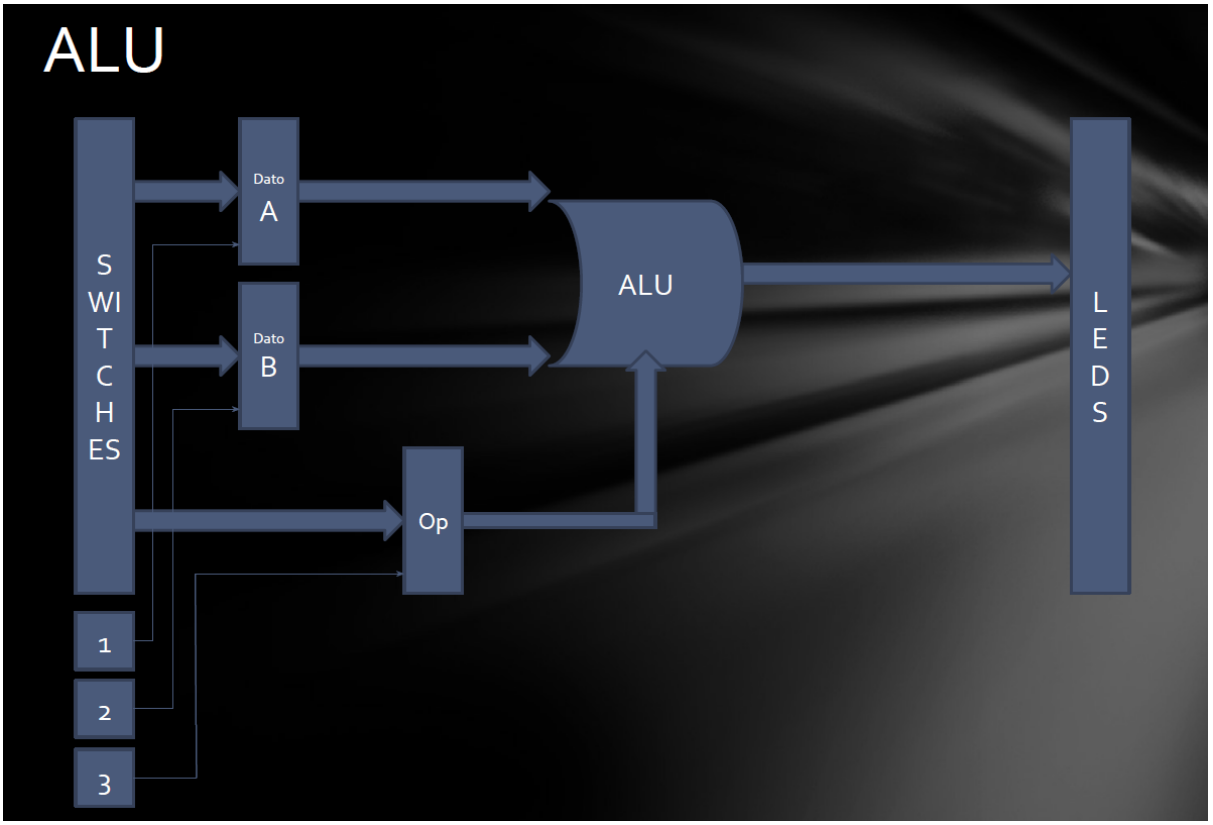
DESARROLLO

Descripción General del trabajo

Se tiene una ALU como módulo principal 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.

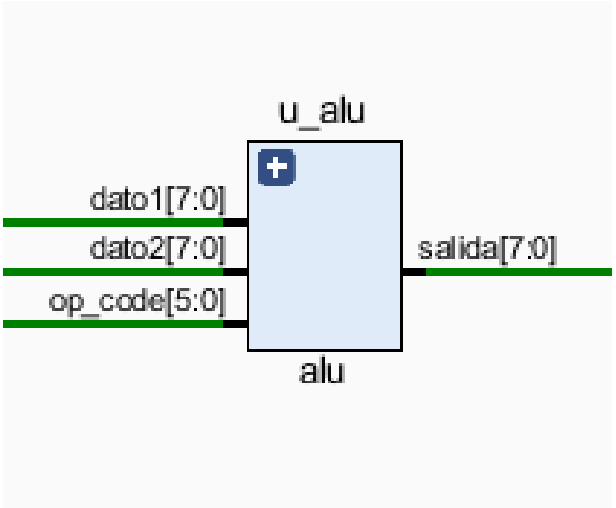
Operación	Código
ADD	100000
SUB	100010
AND	100100
OR	100101
XOR	100110
SRA	000011
SRL	000010
NOR	100111

El sistema cuenta con un switch de 8 bits y 3 botones de un bit cada uno. El botón 1, cuando se presiona, sirve para ingresar el valor que se encuentra en el switch en el dato A. El botón 2, realiza la misma función pero el valor se envía al dato B. El botón 3 es para ingresar el valor que está en el switch al código de operación, el cual identifica la operación que se desea realizar.
Entonces con los botones se ingresan los valores con los que deseo trabajar y el resultado de la operación con los datos se puede observar en la salida donde se encuentran los leds.

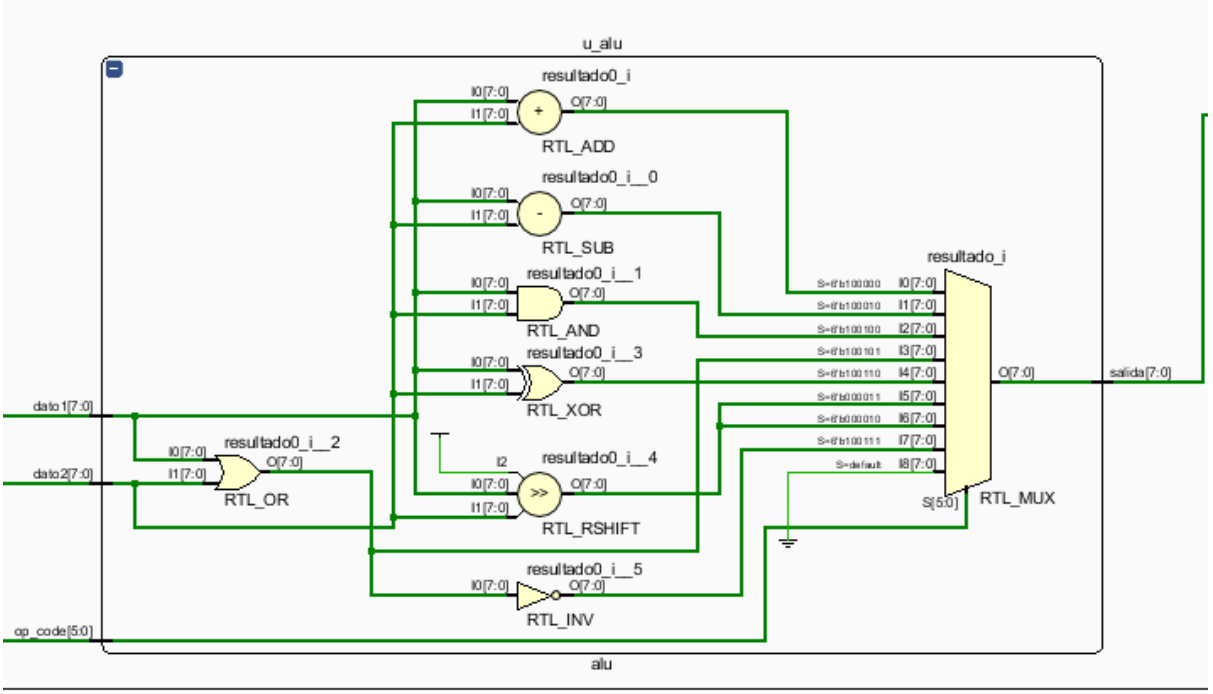


Solución

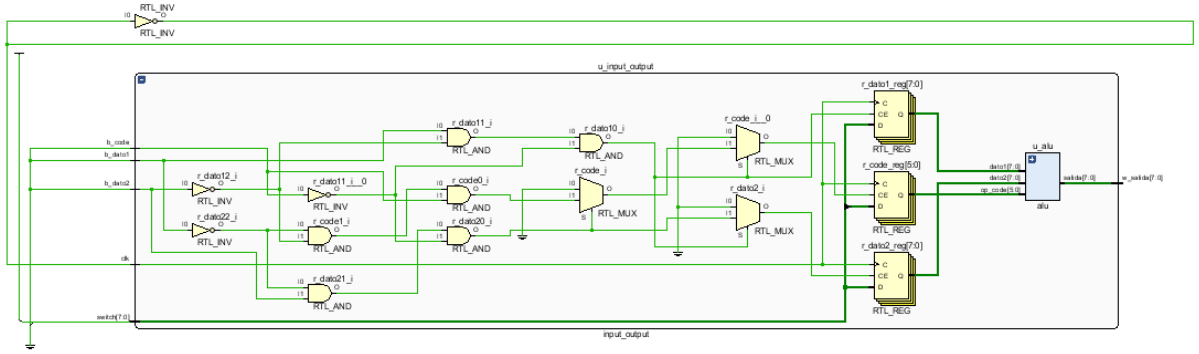
Como se mencionó anteriormente la ALU sólo realiza el cálculo de las operaciones, la cual tiene como entrada el dato1 de 8 bits, dato2 de 8 bits y el op_code de 6 bits. El código de operación indica cuál operación se realizará con los datos. El resultado final se lo asigna a la salida que es de 8 bits.



Como se puede observar en la siguiente imagen la ALU es un combinacional donde se pueden ver que cada operador lógico tiene como entrada dato1 y dato2 y el resultado de cada operador va a un multiplexor en común.



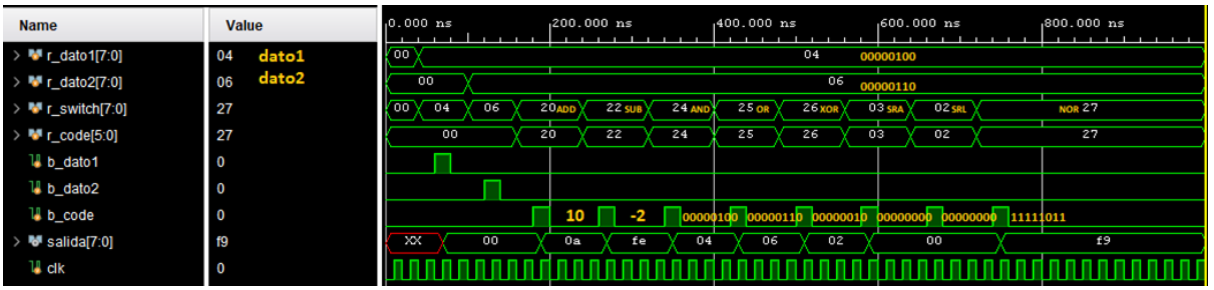
Esquema del análisis RTL



En la imagen se puede observar los cables de entrada que corresponden al switch (8 btis) y los botones (1 bit) y el cable de salida (8 btis). Dentro del sistema se encuentra el módulo de entrada, el registro de dato1 (8 btis), registro de dato2 (8 btis), registro del código de operación (6 btis) y la ALU.
Como observación vemos que además de tener como entrada los botones y el switch, está el clock. El clock nos permite sincronizar. Con este podemos sincronizar los botones que se presionaron y el valor que se envía del switch a los registros y para el test de la ALU.

Simulación de Alu

En el apartado anterior se explicaron los módulos que se hicieron, para estos dos módulos, hicimos 2 test bench.
A continuación se ve el resultado del primero:

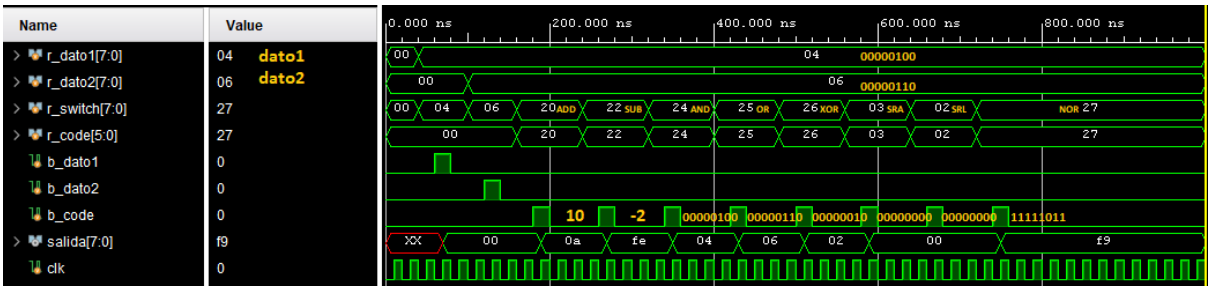



```
# run 1000ns
  ADD CORRECTO
  SUB CORRECTO
  AND CORRECTO
  OR CORRECTO
  XOR CORRECTO
  SRA CORRECTO
  SRL CORRECTO
  NOR CORRECTO
```

Aquí se realizó la simulación únicamente del módulo de la ALU, se puede ver que los resultados son correctos y son muy similares a los que observaremos en el apartado siguiente. Por lo que, no se detallarán, en mayor medida, las operaciones resueltas en esta simulación, pero si se lo hará en el testeo de todo el módulo completo.

Simulación del módulo completo:

Captura 1:




```
# run 1000ns
 ADD CORRECTO
 SUB CORRECTO
 AND CORRECTO
 OR CORRECTO
 XOR CORRECTO
 SRA CORRECTO
 SRL CORRECTO
 NOR CORRECTO
```

En este apartado se muestra el testeo del módulo completo: switch, botones de entrada, ALU y salida. Se puede observar, como en el caso anterior, que todas las operaciones dan correctamente.
Para analizar, de una manera más ordenada, el gráfico de tiempo, se tradujeron a binario o a decimal los números que se utilizan en los cálculos. Podemos ver que todas las operaciones se realizan correctamente. A continuación vamos a ver estos mismos cálculos en mayor detalle:

Datos utilizados:

Dato1: 04_D, 04_H, $0000\ 0100_B$
Dato2: 06_D, 06_H, $0000\ 0110_B$

Operaciones realizadas:

ADD: $04_D + 06_D = 10_D = 0A_H = 0000\ 1010_B$

SUB: $04_D - 06_D = -02_D = FE_H = 1111\ 1110_B$

AND: $0100_B \& 0110_B = 0100_B = 4_D = 4_H$

OR: $0100_B \& 0110_B = 0110_B = 6_D = 6_H$

XOR: $0100_B \& 0110_B = 0010_B = 2_D = 2_H$

SRA: $0000\ 0100_B >>> 6_D = 0000\ 0000B = 0_D = 0_H$

SRL: $0000\ 0100_B >> 6_D = 0000\ 0000_B = 0_D = 0_H$

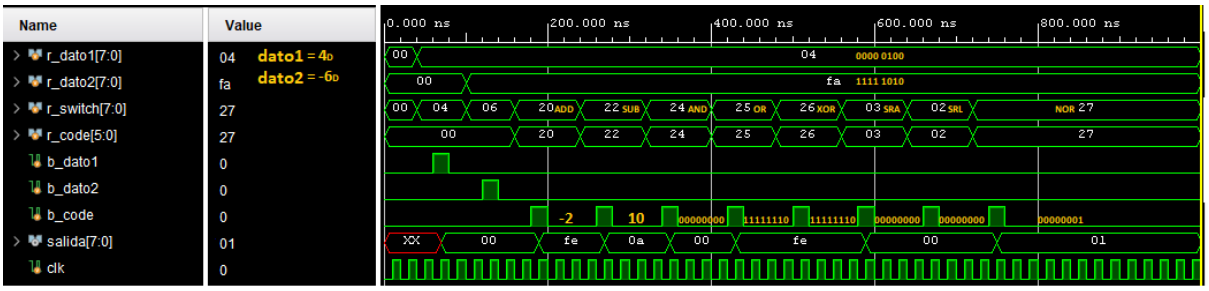
NOR: $0000\ 0100_B$ & $0000\ 0110_B$ = $1111\ 1001_B$ = $F9_H$ = -7_D

Como se puede apreciar, todas las operaciones se realizan correctamente.
Para demostrar que se comporta de la misma manera con números negativos, agregaremos dos gráficos más a continuación, pero no se detallarán los cálculos:

Captura 2:

Datos utilizados:

Dato1: 04_D, 04_H, $0000\ 0100_B$
Dato2: -6_D, FA_H, $1111\ 1010_B$

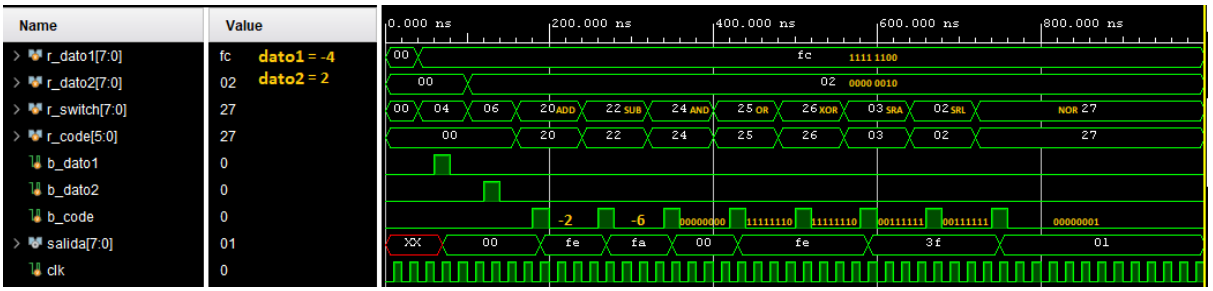



```
# run 1000ns
 ADD CORRECTO
 SUB CORRECTO
 AND CORRECTO
 OR CORRECTO
 XOR CORRECTO
 SRA CORRECTO
 SRL CORRECTO
 NOR CORRECTO
```

Captura 3:

Datos utilizados:

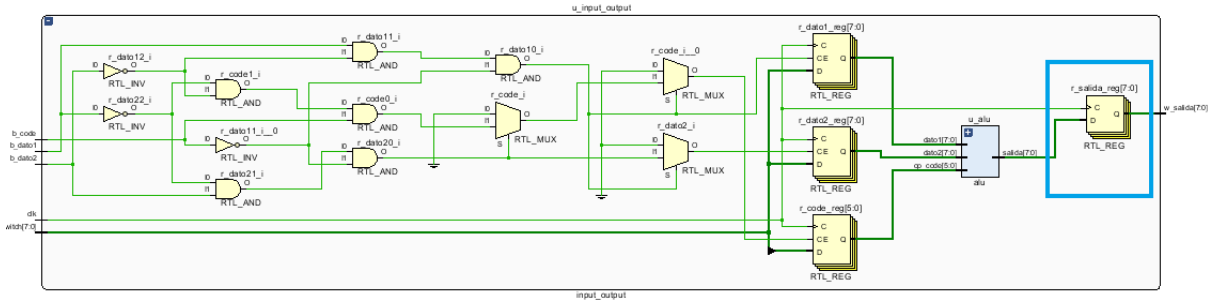
Dato1: -4_D, FC_H, $1111\ 1100_B$
Dato2: 02_D, 02_H, $0000\ 0010_B$



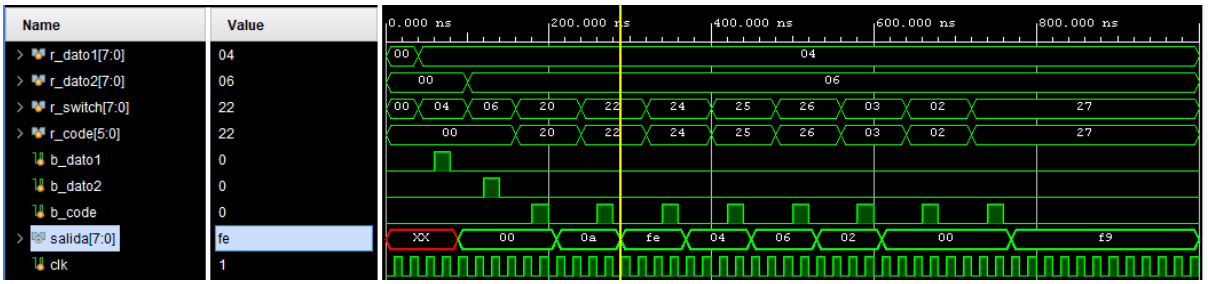
```
# run 1000ns
 ADD CORRECTO
 SUB CORRECTO
 AND CORRECTO
 OR CORRECTO
 XOR CORRECTO
 SRA CORRECTO
 SRL CORRECTO
 NOR CORRECTO
```

Error corregido en la salida

Luego de analizar los gráficos en profundidad para realizar el informe, detectamos que había un error en la salida. Esta no estaba sincronizada, actuando como combinacional, por lo que introdujimos un registro que actúa como un Flip-Flop para actualizar el valor de salida con el clock.



Ahora podemos ver como el gráfico de tiempo cambió levemente, pero se observa claramente que la modificación en la salida se realiza luego del flanco de subida del clock.



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.

No solo usamos las herramientas de vivado, nos fue de gran ayuda usar la plataforma en edaplayground donde pudimos identificar específicamente en cuál línea del código había un error de sintaxis o de instancia.

Debido a que no contamos con la FPGA físicamente, no pudimos realizar la implementación.