UNIVERSIDAD
COMPLUTENSE
MADRID

DISEÑO AUTOMÁTICO DE SISTEMAS

LABORATORIO 1: ALU COMBINACIONAL

Daniel Portuondo Rodríguez

Declaro que este trabajo es resultado de mi propio esfuerzo, que todas las fuentes consultadas han sido citadas adecuadamente y que no he incurrido en plagio ni en ninguna otra forma de deshonestidad académica.

2. Diseño

2.1. Especificación del diseño

• Descripción funcional clara del sistema.
El sistema se trata de una ALU combinacional de 2 operandos de 8 bits cada uno, capaz de sumar, restar, multiplicar y calcular el opuesto, generando una salida de 16 bits que es mostrada en formato binario por los leds de la placa de prototipado Xilinx Basys 3.

• Entradas y salidas (tabla con nombre, ancho y descripción).

Nombre	Tipo	Ancho (bits)	Descripción
sws	Entrada	16b	Señal de entrada de lógica directa que se corresponde a los valores de los switches. Los 8 bits más significativos pertenecen al operando izquierdo y los 8 menos significativos al operando derecho.
btnL	Entrada	1b	Señales de entrada que provienen de los botones izquierdo y derecho de la placa respectivamente. Cada botón define un bit, y juntos conforman la señal opCode de 2 bits, que selecciona la operación que se aplica. 00: Suma 01: Resta 10: Negación del operando derecho 11: Multiplicación
btnR	Entrada	1b	
leds	Salida	16b	Señal de salida que proporciona el resultado de la operación realizada, con lógica directa. Se llama así porque en este caso va conectada a los leds de la placa.
an_n	Salida	4b	Señal de salida fijada a ‘1110’, que define cuáles de los 7 segmentos se actualiza (con lógica inversa). En este caso siempre y únicamente se actualiza el 7 segmentos de la derecha.
segs_n	Salida	8b	Señal de salida que contiene en sus 8 bits la cifra menos significativa del resultado, en formato para ser mostrado por los 7 segmentos de la placa.

• **Requisitos funcionales y restricciones.**

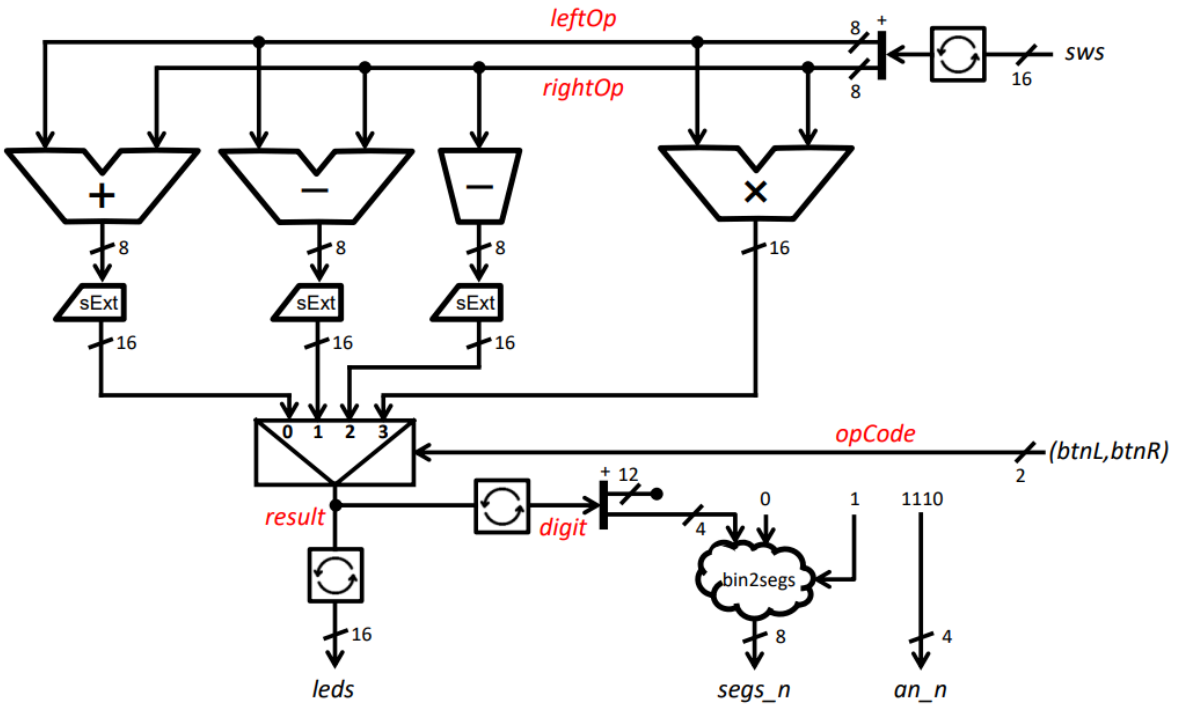
- **Requisito funcional 1**: el sistema ha de poder realizar la operación de suma, resta y multiplicación entre sus operandos, además de la negación del operando derecho. El formato tanto para los operandos como para el resultado será en complemento a 2.
- **Requisito funcional 2.1**: el sistema debe seleccionar la operación a realizar mediante dos señales std_logic de 1 bit cada una, que provendrán de dos botones ajenos al sistema.
- **Requisito funcional 2.2**: los operandos serán recibidos en una señal en formato std_logic_vector de 16 bits, siendo los 8 bits más significativos el operando izquierdo y los 8 menos significativos el operando derecho. Esta señal provendrá de 16 switches ajenos al sistema.
- **Requisito funcional 3.1**: el sistema ha de generar el resultado en formato de señal std_logic_vector de 16 bits, con el fin de ser conectada a 16 leds para mostrar el resultado de manera visual.
- **Requisito funcional 3.2**: el sistema adicionalmente ha de generar una señal en formato std_logic_vector de 8 bits, donde se guarde la cifra menos significativa del resultado en el formato necesario de los 7 segmentos (referirse a la documentación de la práctica, página 14 para comprobar el formato).

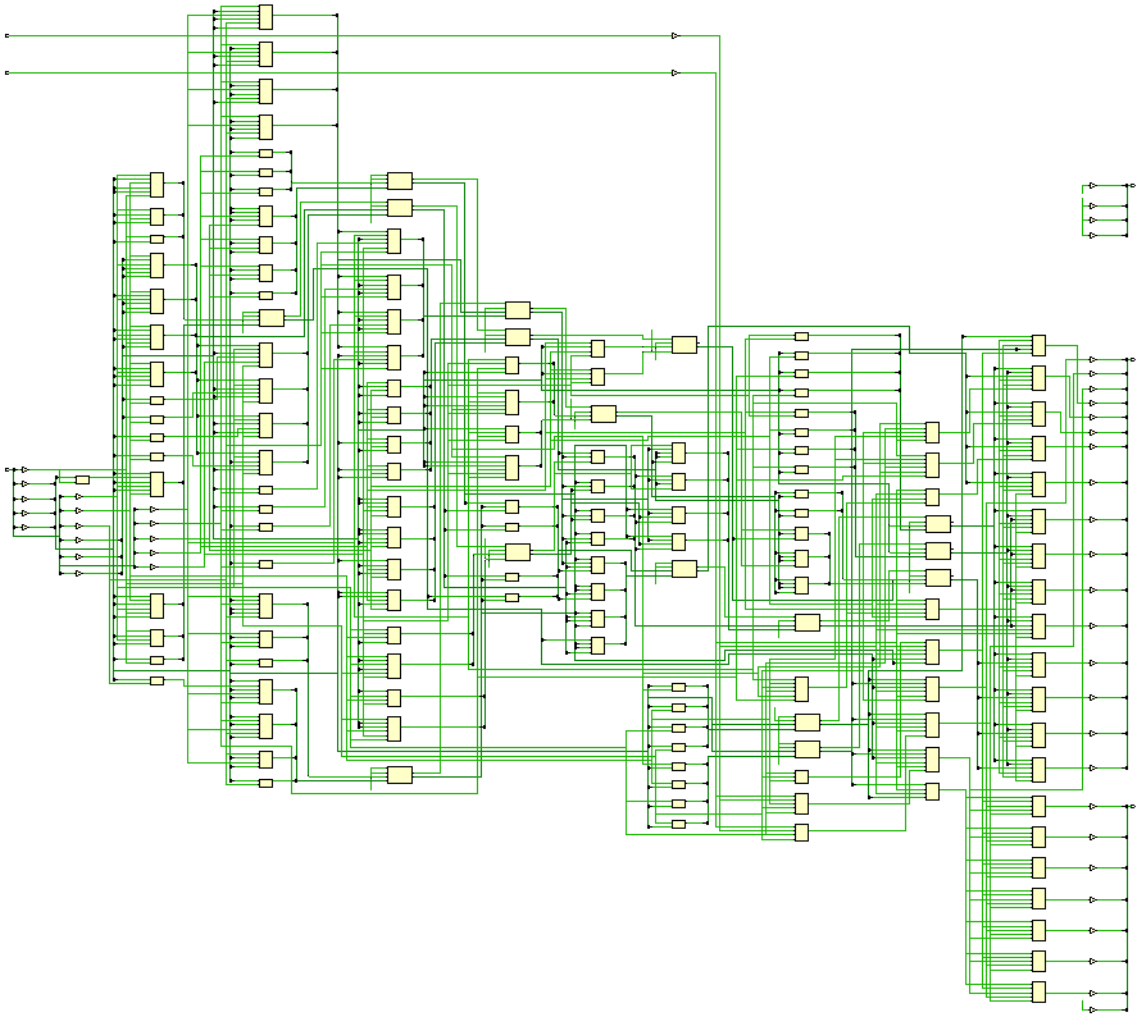
• **Supuestos adoptados.**

- **Conexión externa**: Se asume que los receptores de la salida del sistema sean descritos en los requisitos funcionales 3.1 y 3.2.
- **Representación de datos**: Se asume el uso de complemento a 2.
- **Lógica directa sw:** Se ha dado por hecho que los switches son de lógica directa (ofrecen un 1 lógico al ser activados)
- **Lógica directa btnx:** A su vez, el sistema también depende de que los botones sean igualmente de lógica directa.

2.2. Descripción de la arquitectura

Se trata de un sistema basado en datapath simple, con módulos combinacionales (sumador, restador, extensor de signo, etc).
En este caso el único módulo que ha sido diseñado en una entidad independiente ha sido el ‘bin2segs’, cuya funcionalidad es recibir un dígito codificado en 4 bits y traducirlo a formato 7 segmentos.
Adjunto imágenes tanto del diagrama proporcionado por el guión de la práctica como del esquemático del diseño implementado en Vivado.


leftOp
rightOp
8
8
+
sws
16
+
−
−
×
8
8
8
16
sExt
sExt
sExt
16
16
16
0 1 2 3
opCode
(btnL,btnR)
2
result
digit
+ 12
4
0
1
1110
bin2segs
16
8
4
leds
segs_n
an_n



2.3. Decisiones de diseño

2.3.1. Dominios de reloj

En este caso no tenemos ningún dominio de reloj, ya que se trata de un sistema puramente combinacional.

2.3.2. Dominios de reset

Dada la naturaleza combinacional y carente de biestables del sistema, no tenemos ni necesitamos una señal de reset, ya que no hay ningún biestable que resetar.
Esto también hace que pese a que trabajemos con señales asíncronas, no haya riesgo de metaestabilidad.

3. Resultados de implementación física

3.1. Utilización de recursos

Tipo	Usados	%Uso
LUTs	**102**	**0.49%**
FF	**0**	**0%**
BRAM	**0**	**0%**
DSP	**0**	**0%**
IOs	**46**	**43.4%**

3.2. Resultados de timing

Debido a que estamos ante un sistema puramente combinacional, no tenemos ningún dominio de reloj.
Por lo tanto:

- WNS**:** NA
- TNS**:** NA

Sin embargo sí que tenemos caminos sin restricciones, debido a que todas las señales son asíncronas:

- In to Out: El sistema cuenta con 10 caminos sin restricciones de este tipo. El que mayor delay presenta es el que une sws[12] con leds[15], de 18.936ns (40.548% causado por la lógica y 59.452% causado por el enrutado).

3.3. Consumo de potencia

• **Estimación estática**
0.089W
• **Estimación dinámica**
1.99W (58% Signals, 38% Logic, 4% IO)

4. Análisis crítico de resultados

El sistema es muy eficiente y usa recursos mínimos de cómputo de la FPGA, al fin y al cabo es una ALU muy sencilla.

Se podría mejorar el rendimiento introduciendo pipelining, pero esto en consecuencia tendría un enorme aumento de la complejidad del sistema, al tener que introducir un reloj y flip flops. Además es imposible para un humano introducir e interpretar operandos y resultados respectivamente a una velocidad que permita aprovechar este aumento de rendimiento.

La única mejora que se me ocurre sería bajar aún más el toggle rate de las señales para obtener una estimación del consumo de energía aún más fiel a la realidad, pero parece que no es posible en Vivado.

5. Dificultades encontradas y soluciones adoptadas

Power report: Inicialmente, cuando generé los informes de consumo estático y dinámico, el consumo dinámico se trataba de 46W. Esto es una cifra totalmente desmedida teniendo en cuenta el sistema del que hablamos.
Tras una serie de consultas a Gemini y un análisis crítico de los datos que me arrojaba vivado, pude discernir que la causa de este problema se trataba del toggle rate de las señales de salida 'leds' (31W) y 'segs_n' (13.5W).
Una vez ajusté el toggle rate para reflejar que son señales cuya conmutación es provocada por intervención humana (1-5 Hz), obtuve los resultados detallados en la sección 3.3.
Para el ajuste de los toggle rate y especificar que no hay heatsink se puede usar la interfaz gráfica de vivado, pero también se puede directamente añadir sentencias TCL en el documento de constraints.
Tras usar la interfaz gráfica observé que se añadieron las siguientes líneas de comandos TCL:

- set_operating_conditions -heatsink none
- set_switching_activity -toggle_rate 0.001 -static_probability 0.500 [get_ports segs_n]
- set_switching_activity -toggle_rate 0.001 -static_probability 0.500 [get_ports leds]

6. Anexos

- **Nombre del archivo zip con el proyecto Vivado**
  Daniel_Portuondo_lab1.zip

- **Ficheros de constraints**
  DAS_labs/source/lab1/lab1.xdc

- **Scripts de síntesis**
  DAS_labs/projects/lab1/lab1.runs/synth_1/lab1.tcl

- **Versiones de herramientas y lenguajes**
  Vivado v2023.1 y VHDL

- **Dispositivo FPGA utilizado**
  Digilent Basys 3 Artix-7