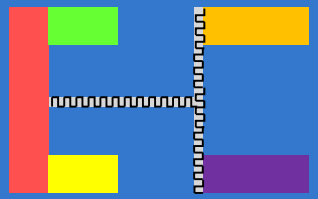


Práctica 2.
Circuito VHDL

Índice

- Práctica 2.1
  - Diseño de un circuito de desplazamiento de 4 bits.
    - El objetivo de esta práctica es el diseño de un circuito de desplazamiento utilizando VHDL. El circuito debe describirse utilizando el modelo estructural.
- Práctica 2.2
  - Diseño de la Unidad Aritmética y Lógica simplificada del MIPS.
    - El objetivo de esta práctica es el diseño de una Unidad Aritmética y Lógica (ALU) utilizando VHDL. La ALU debe ser definida utilizando el modelo de comportamiento.

Realización de las Prácticas

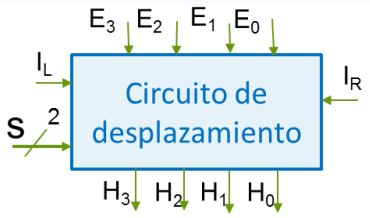
- La realización de las prácticas implica la entrega del trabajo propuesto que se detalla a continuación.
- Cada una de las prácticas se entregará en un **archivo comprimido** que deberá contener:

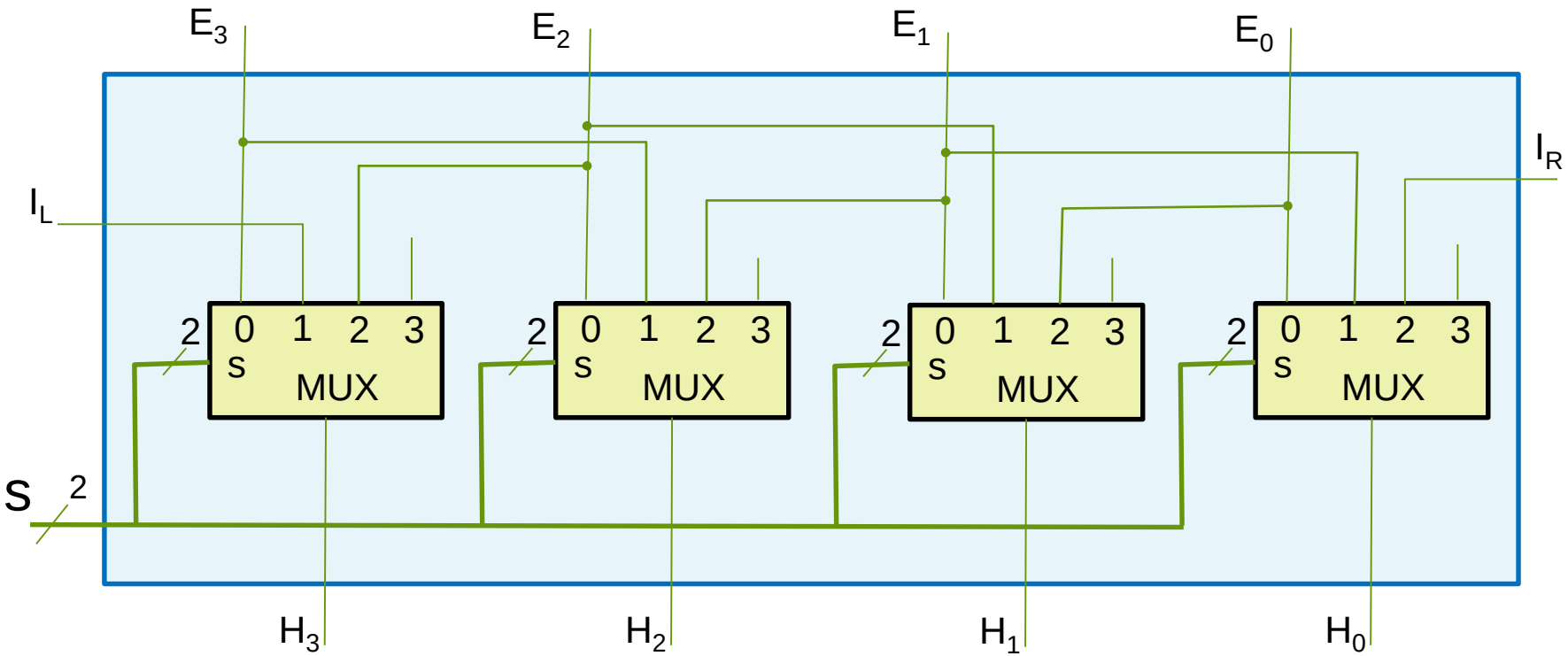
1. El archivo con la **memoria documental** que incluya:
   - La descripción de cómo se ha diseñado y se ha desarrollado la práctica.
   - El código en VHDL que se ha implementado.
   - Volcados de pantalla que demuestren su correcto funcionamiento.
2. Los archivos asociados a la **implementación** para que el profesor pueda ejecutarlos.

Práctica 2.1. Circuito de Desplazamiento

Realización práctica 2.1

- Diseña un circuito simple de desplazamiento de 4 bits como el que se muestra en la figura que permita desplazar un bit a la derecha o a la izquierda. El circuito de desplazamiento debe ser descrito en VHDL utilizando el modelo estructural.





Práctica 2.1. Circuito de Desplazamiento

Realización práctica 2.1

- Diseño y prueba del circuito
  - Diseñar en VHDL el circuito de desplazamiento de 4 bits utilizando el modelo estructural. Previamente se deberá modelar en VHDL un circuito multiplexor de 4 a 1 y posteriormente describir el circuito de desplazamiento a partir de dichos multiplexores utilizando la sentencia PORT MAP.
  - El circuito de desplazamiento realizará el desplazamiento de un bit a la derecha o de un bit a la izquierda según el valor de la señal de control S de dos bits $S=S_1S_0$:

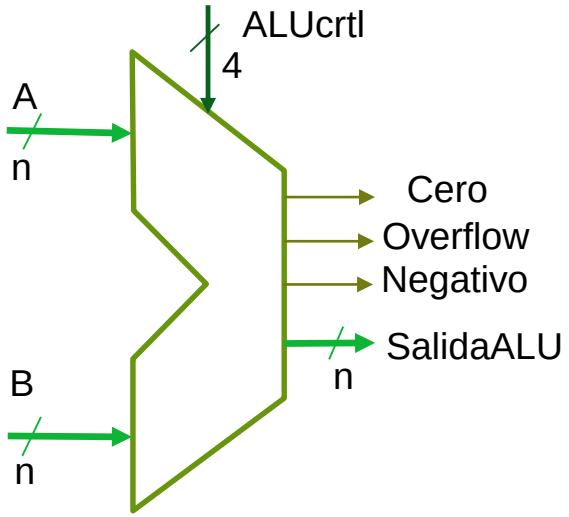
S_1S_0	Operación
00	No hay desplazamiento
01	Desplazamiento a la derecha
10	Desplazamiento a la izquierda

  - El desplazamiento será lógico con lo que el valor de los bits I_L e I_R será 0.
  - Escribe un testbench que nos permita comprobar su funcionamiento.

PRÁCTICA 2.2. ALU SIMPLIFICADA DEL MIPS

Realización práctica 2.2

- Diseña una Unidad Aritmética y Lógica como la que ilustra la figura. Las especificaciones de las instrucciones que debe implementar la ALU se muestran en la tabla, en ella aparecen las señales de control y las operaciones que debe implementar. La ALU debe ser definida en VHDL utilizando el modelo de comportamiento.



ALUctrl	Operación
0000	And
0001	Or
0010	Suma
0110	Resta
0111	Set on less than
1100	nor

PRÁCTICA 2.2. ALU SIMPLIFICADA DEL MIPS

- Diseño del circuito
  - Escribe el código VHDL de la ALU que realice las operaciones que aparecen en la tabla anterior para las señales de control dadas en dos entradas de n bits A y B. El resultado de la operación aparecerá en un puerto de salida SalidaALU de n bits después de 100ps. La ALU también obtiene los indicadores de resultado Cero, Negativo y Overflow.
  - Cero: Este flag se pone a uno si el resultado de la operación es cero, en otro caso se pone a cero. Este flag se activará tanto para las operaciones lógicas como aritméticas.
  - Negativo: Este flag se pone a 1 si el resultado de la operación ha sido negativo, en otro caso se pone a cero. Este flag se activa sólo para las operaciones aritméticas, para las operaciones lógicas debe ser siempre cero.
  - Overflow: Para las operaciones aritméticas, este flag se pone a uno si el acarreo de entrada del bit más significativo del resultado no es igual al acarreo de salida del bit más significativo del resultado de la ultima operación aritmética, y se pone a cero en otro caso. Para las operaciones lógicas (AND, OR y NOT) este flag siempre está a cero.
  - Considerar que el valor de n es 32 (A, B y SalidaALU son te 32 bits)

- Prueba del circuito.
  - Calcula y obtén los valores de ALUout, Cero, Negativo y Overflow para las entradas que aparecen en la tabla.
  - Escribir un testbench para probar el funcionamiento del diseño realizado para las entradas A, B y ALUCrtl que aparecen en la tabla y compara los resultados con los valores calculados previamente.

Número	A_{hex}	B_{hex}	ALUctrl	$SalidaALU_{hex}$	Cero	Negativo	Overflow
1	FFFFFFFF	00000000	0000				
2	98989898	89898989	0001				
3	00000001	FFFFFFFF	0010				
4	6389754F	AD5624E6	0010				
5	00000000	00000001	0110				
6	F9684783	F998D562	0110				
7	45B23452	45AF9352	0111				
8	9ABCDEDF	9ABCDEFD	1100				
9	89BCDE34	C53BD687	1111				