Ejercicios de Examen de UPG (Laboratorio de Sistemas Digitales)

Ejercicio 1. Final 2017-18 (Proyecto Ejercicio1.circ). Dada la UPG proporcionada en el fichero Ejercicio1.circ, deberás implementar una unidad de control específica (figura inferior) capaz de calcular el valor absoluto de un número en Ca2.

a. Obtén un formato para las palabras de control de la UPG proporcionada indicando claramente los campos y el número de bits de cada uno. ¿Qué tamaño tienen las palabras de control?

b. Dado el algoritmo de cálculo de valor absoluto de un número en Ca2 proporcionado a continuación, obtén las palabras de control asociadas a cada instrucción y represéntalo en una tabla. Utiliza "don't care" (X) cuando sepas que el valor del bit da igual (es indiferente si toma valor 0 ó 1).

# Inst.	Nemónico	Descripción
0	IN R1	Guarda DATAIN en el registro R1
1	CMPLT R2, R1, 0	Compara si el contenido de R1 es menor que 0 y lo guarda en R2
		Salta a la instrucción 4 si la comparación en R2 es falsa (z=1). Si z=0, continúa
2	NOT R3, R1	invierte el valor de todos los bits de R1
3	ADDI R1, R3, 1	R1 = R3 + 0x01
4	OUT R1	Muestra el contenido de R1 por DATAOUT

c. Diseña la unidad de control que cumple la siguiente funcionalidad:
   - En el ciclo en el que la señal Reset vale 0 el dato que llega por el bus de entrada de la UPG es válido y comienza el algoritmo. La entrada representa un valor entero codificado en Complemento a 2.
   - Cuando la UPG tenga el resultado correcto en el bus de salida, lo indicará poniendo a 1 la señal de control FIN. La señal FIN valdrá 0 en cualquier otro caso.
   - El resultado se mantendrá en el bus de salida mientras Reset valga 0.
   - Si se recibe Reset=1 mientas se están llevando a cabo otra operación, se reinicia el algoritmo.

   1. Obtener el grafo de estados (Moore). Dibuja el grafo y explica tus decisiones en la hoja del examen.
   2. Obtener la tabla de transiciones y salidas (una sola tabla) para dicho grafo.

d. Implementa mediante una ROM la tabla de transiciones y salidas y lleva a cabo la implementación del circuito secuencial completo (UGP +UC). Guarda tu solución como **Ej1Sol**.

Ejercicio 2. Parcial 2016-17 (Proyecto Ejercicio2.circ) Se dispone de un sistema como el de la figura 1, implementado en el fichero Ejercicio2.circ. Dicho circuito tiene una entrada de datos (*A*) de 4 bits, por donde puede recibir un dato en cada ciclo de reloj. Ese dato puede ser almacenado en los registros R1 ó R2 según indique la entrada *sel_R*. Los datos de R1 y R2 pueden ser sumados o restados (señal *resta/suma*; 0->resta, 1->suma) a través de la unidad aritmética proporcionada. La unidad aritmética almacena el resultado en el registro R3 y genera una salida (*Z*) que indica si el resultado de la operación (suma o resta) ha sido 0 (Z=1).

Se desea implementar un circuito secuencial síncrono (figura 2) capaz de controlar el sistema descrito. La funcionalidad requerida será la siguiente:

- En el ciclo en el que la señal *dato_valido*=1, el dato que llega por el bus de entrada A es válido (dato d1). En el siguiente ciclo llega el segundo dato válido (dato d2) por el bus de entrada A.
- Si ambos datos son iguales, no se realiza ninguna acción más y el sistema genera una señal de FIN de forma indefinida.
- Si los datos son distintos, en el registro R3 debe guardarse la resta d1-d2 y mantener la señal FIN alta indefinidamente (suponemos que los valores d1 y d2 nunca generan *overflow* en las operaciones de la unidad aritmética).
- Las señales Ld1, Ld2 y Ld3, cuando toman valor 0, inhiben la escritura en el correspondiente registro (permitiendo la escritura en el registro cuando toman valor 1).

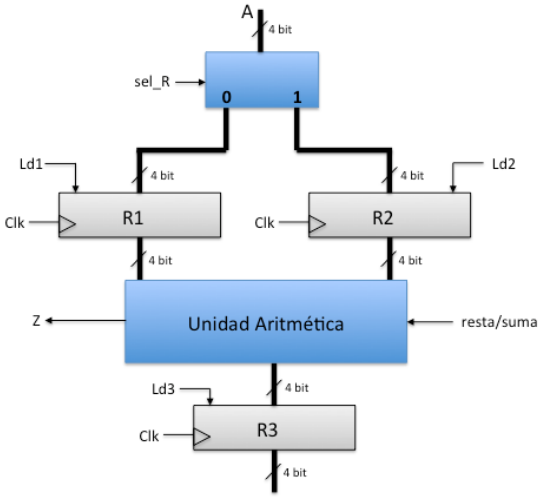


Figura 1

Figura 2

Ejercicio 3. Final 2015-16 (Proyecto Ejercicio3.circ) Se desea diseñar una unidad de control específica que se conecta a la UPG que se facilita en el circuito **Ejercicio3.circ**. La unidad de control específica debe implementar el algoritmo de Euclides para el cálculo del Máximo Común Divisor, que se corresponde con el siguiente código:

PC	Operación	Mnemónico
0	Store DATAIN in R0	IN R0
1	Store DATAIN in R1	IN R1
2	Compare R1 = R0	CMPEQ R2, R1, R0
	If z=0, go to 6, else continue	
3	Compare R0 < R1	CMPLT R2, R0, R1
	If z=0 go to 5, else continue	
4	R0 = R0 – R1	SUB R0, R0, R1
	go to 2	
5	R1 = R1 – R0	SUB R1, R1, R0
	go to 2	
6	Print Result indefinitely	OUT R0

Realiza las siguientes operaciones:

a. Obtén un formato para las palabras de control de la UPG proporcionada indicando claramente los campos y el número de bits de cada uno.
b. Dado el algoritmo de la tabla superior, escribe en tu hoja de examen las palabras de control asociadas a cada instrucción.
c. Diseña la unidad de control para el agoritmo descrito y lleva a cabo las conexiones con la UPG proporcionada.

Ejercicio 4. Parcial 2018-19 (Proyecto Ejercicio4.circ) Dada la UPG proporcionada en el fichero **Ejercicio4.circ** semejante a la utilizada en prácticas (la única diferencia es que el inmediato, IMMED, no tiene extensor de signo), deberás implementar una unidad de control específica capaz de llevar el segundero de un reloj.

d. Obtén un formato para las palabras de control de la UPG proporcionada indicando claramente los campos y el número de bits de cada uno. ¿Qué tamaño tienen las palabras de control?

e. Dado el algoritmo para el segundero proporcionado a continuación, obtén las palabras de control asociadas a cada instrucción y represéntalo en una tabla. **Trata de que la salida (OUT) tome el valor del segundo actual (R1) siempre que sea posible**. Utiliza "don't care" (X) cuando sepas que el valor del bit da igual (es indiferente si toma valor 0 ó 1).

Num. Instrucción	Mnemónico	Descripción
0	XOR R1, R1, R1	R1 = 0
1	ADDI R1, R1, 1	R1 = R1 + 1
2	CMPLT R2, R1, 60	Compara si el contenido de R1 es menor que 60
		Si z=0 (es menor que 60), salta a la instrucción 1. Si z=1 (es mayor o igual a 60), salta a la instrucción 0.

f. Diseña la unidad de control que cumple la siguiente funcionalidad:
   - En el ciclo en el que la señal Reset vale 0 el segundero comienza la cuenta.
   - Si se recibe Reset=1 mientras se están llevando a cabo otra operación, se reinicia el segundero.

Ejercicio 5. Final 2018-19 (Proyecto Ejercicio5.circ). Dada la UPG proporcionada en el fichero **Ejercicio5.circ** semejante a la utilizada en prácticas, deberás implementar una unidad de control específica capaz de convertir valores de entrada a su número par más cercano.

a. Obtén un formato para las palabras de control de la UPG proporcionada indicando claramente los campos y el número de bits de cada uno. ¿Qué tamaño tienen las palabras de control?

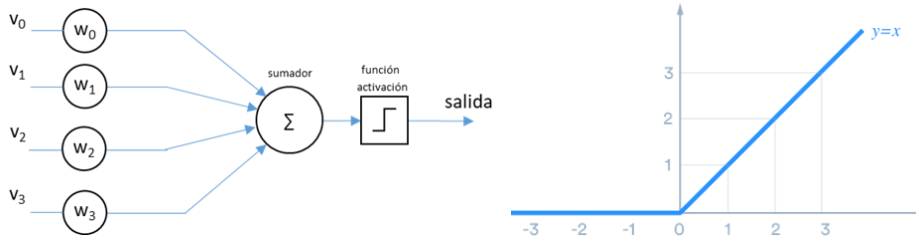
b. Dado el algoritmo para el conversor proporcionado a continuación, obtén las palabras de control asociadas a cada instrucción y represéntalo en una tabla. Utiliza "don't care" (X) cuando sepas que el valor del bit da igual (es indiferente si toma valor 0 ó 1).

Num. Instrucción	Mnemónico	Descripción
0	IN R2	R2 = input
1	ANDI R3, R2, 1	Comprobar si el número es par
		Si z=1 (es un número par), salta a la instrucción 3. Si z=0 (es impar), continua.
2	ADDI R2, R2, 1	suma 1 al registro R2
3	OUT R2	output = R2

c. Diseña la unidad de control que cumple la siguiente funcionalidad:
   - En el ciclo en el que la señal Reset vale 0 el valor en IN es válido.
   - Si se recibe Reset=1 mientras se están llevando a cabo la conversión, se interrumpe la conversión.

Dibuja el grafo de estados, obtén la tabla de transiciones/salidas, y realiza la conexión completa de tu Unidad de Control con la UPG y los flip-flop necesarios.

Ejercicio 6. Septiembre 2018-19 (Proyecto Ejercicio6.circ). El elemento básico utilizado en la construcción de redes neuronales para *Deep Learning* (tan de moda actualmente) se denomina perceptrón. Dicho elemento trata de emular el funcionamiento de una neurona, actuando como un clasificador. A partir de un vector de valores de entrada el clasificador decide si el elemento representado por dicho vector pertenece a una determinada clase.

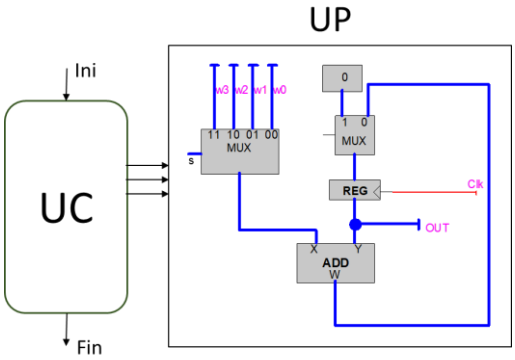


La estructura interna del perceptrón, mostrada en la figura superior izquierda, es sencilla. En primer lugar, cada valor de entrada (V_i) se multiplica por un peso (w_i). A continuación, se suman todos los valores conjuntamente (sumador). Finalmente, se aplica una función de activación que toma la decisión sobre la clasificación. En este ejercicio diseñaremos los circuitos asociados al sumador y a la función de activación. Nuestro diseño se ajustará a los siguientes parámetros:

- El sumador de nuestro perceptrón toma 4 valores como entrada. Cada valor es el resultado de multiplicar una entrada por su peso (suponemos que esa operación ya está hecha) y se representa mediante un número, en Ca2, de 4 bits.
- La función de activación, denominada ReLU, responde a la gráfica de la figura superior derecha, tomando valor 0 cuando su entrada es negativa (el resultado final del sumador es un valor menor que 0)y tomando el valor de entrada en caso contrario.

Para el bloque sumador se utilizará una unidad de proceso específica (Figura inferior). Es necesario crear una unidad de control (ver figura inferior) con la siguiente funcionalidad:

- En cada ciclo se debe acumular un elemento distinto del vector a nuestra suma.
- La unidad de control tendrá una señal de entrada (**Ini**) para indicar cuándo los valores del vector de entrada están listos para sumar. En el ciclo en el que la señal **Ini** vale 1, el registro (**REG**) debe inicializarse al valor 0, en el resto de casos, REG irá acumulando el valor de la suma calculada hasta ese ciclo, y la señal Ini debe ignorarse.
- Una señal de salida (Fin) tomará valor 1 cuando la suma de los 4 valores de entrada ha finalizado.



Lleva a cabo las siguientes tareas:

a) Implementa en Logisim la Unidad de Proceso específica, en un circuito de nombre **UP**.
b) (2p) Dibuja el grafo de estados de tu UC en la hoja del examen. Implementa el circuito asociado a dicho gráfico con una ROM y los Flip-flops necesarios, y conecta tu unidad de control al circuito UP**.**
c) (1,5p) Crea un circuito combinacional de nombre **Activación** que implemente la funcionalidad de la función de activación descrita en el enunciado, escribe su tabla de verdad en la hoja del examen.
d) (2,5p) Finalmente conecta todos los circuitos (UC + UP + Activación) para comprobar su funcionamiento. Guarda el circuito completo como **Ej6Sol**.

Ejercicio 7. Parcial 2019-20 (Proyecto Ejercicio7.circ) Dada la UPG proporcionada en el fichero **Ejercicio7.circ,** lleva a cabo las siguientes tareas:

a. Comprobarás que la ALU está completamente vacía. Utiliza los componentes de la librería de Logisim para implementar una ALU capaz de hacer las operaciones de la tabla inferior (con el código de operación indicado). Guarda los cambios en el circuito **ALU**.
b. Recuerda que la salida z de la ALU indica si el valor de la operación llevada a cabo es igual a cero (si W=0 entonces z=1). Implementa el circuito **Z** correspondiente como suma de minterms y Incluye la cápsula Z dentro de la ALU del apartado a.
c. Dadas las palabras de control de la tabla inferior, correspondientes a la UPG del ejercicio, determina el contenido del banco de registros tras la ejecución de cada una de ellas.

Tabla de Operaciones

F	W
000	NOT(X)
001	AND(X,Y)
010	ADD(X,Y)
011	XOR(X,Y)
100	CMPEQ(X,Y)
101	CMPLE(X,Y)
110	CMPLT(X,Y)
111	Y

Palabras de Control

In/AL	**WrD**	**@D**		**@A**		**@B**		**Rb/I**	**F**			**IMMED**			
0	1	0	0	0	1	0	0	0	0	1	0	1	1	1	0
0	1	1	1	0	0	0	0	1	0	1	1	0	0	1	1
0	1	0	1	1	0	0	1	1	0	0	0	1	0	1	0

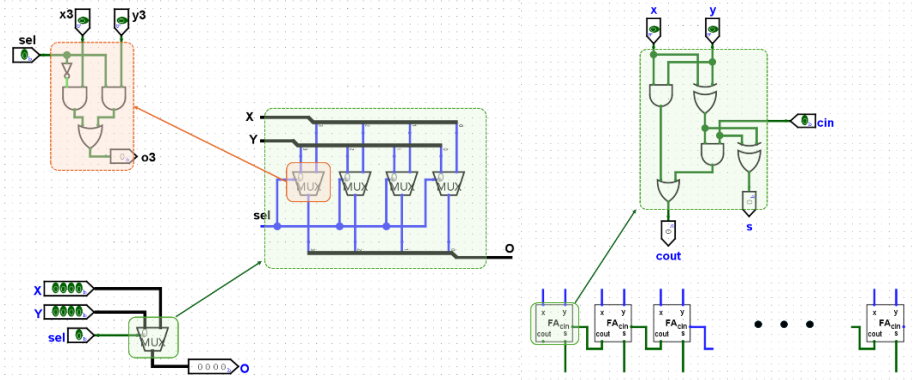
Contenido Banco Registros (decimal)

	R0	R1	R2	R3
Valor Inicial	0	0	0	0
Palabra 0				
Palabra 1				
Palabra 2				

Ejercicio 8. Parcial 2021-22 (Proyecto Ejercicio8.circ) Para este ejercicio vamos a trabajar con la UPG proporcionada en el fichero Ejercicio8.circ. En primer lugar, vamos a llevar a cabo un análisis temporal para determinar a qué frecuencia máxima de reloj puede operar nuestra UPG. Asume que el retardo de los componentes es el siguiente: Not: 20 **ns**, And/Or: 60**ns**, Xor: 100**ns**, Flip-Flop: 200**ns**. Responde a las siguientes preguntas:

a. Sabiendo que el camino crítico es de tipo Biestable-Biestable y que la operación de suma es el componente de mayor retardo de la ALU, determina el tiempo de propagación para dicho camino (Las implementaciones de Mux y Sumador corresponden con las de la figura inferior).

b. ¿Es 500KHz una frecuencia de operación razonable para nuestra UPG? Razona tu respuesta.



Hemos detectado que en dicha UPG se van a llevar a cabo muchas operaciones de multiplicación, por lo que hemos decidido modificarla para **añadir** en la ALU una unidad funcional que realice dicha operación.

c. Razona que implicaciones tendría en nuestro circuito y la palabra de control el cambio realizado.

Sabemos que la operación de multiplicación es muy costosa desde el punto de vista energético, por lo que hemos implementado un algoritmo que comprueba si uno de los operandos es cero, y en ese caso se utiliza como valor de salida directamente.

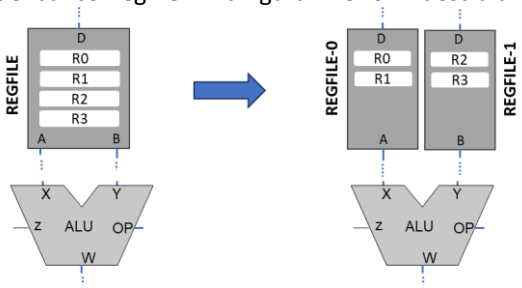
Instrucción	Mnemónico	Notas
0	IN R1	*Recibe primer operando*
1	IN R2	*Recibe segundo operando*
2	ANDI R0,R1,0xF	*Comprueba si el primer operando es 0*
	Si R0=0 (z=1) ejecuta instrucción 4, si no (z=0), ejecuta instrucción 3	
3	MULT R0,R1,R2	*Obtener la multiplicación de ambos operandos*
4	OUT R0	*Muestra el resultado de la operación*

d. En la librería de Logisim encontrarás un multiplicador. Para evitar las implicaciones del apartado c, y atendiendo al algoritmo expuesto anteriormente, sustituye la unidad funcional que creas conveniente de la ALU, por el multiplicador. Guarda tus cambios en el proyecto.

e. Si la cápsula de multiplicación presenta un tiempo de propagación del doble que la cápsula de sumador, ¿Qué le ocurrirá a la frecuencia de operación de nuestra UPG?

f. Propón un formato de palabra de control para tu UPG y obtén las palabras de control asociadas a cada instrucción del algoritmo anterior según el formato propuesto.

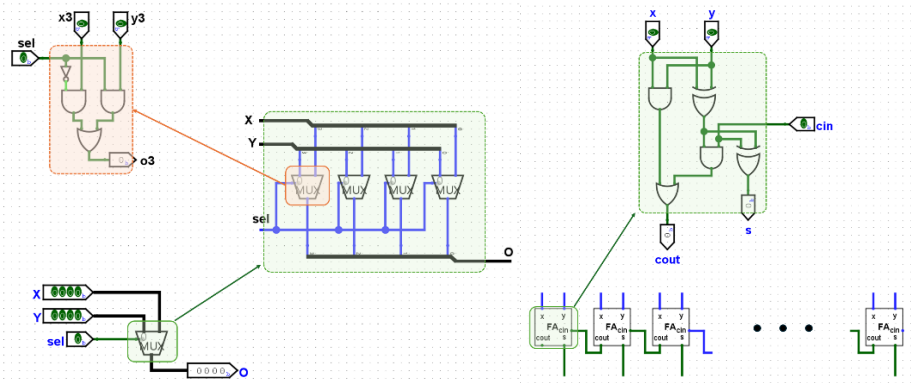
Ejercicio 9. ordinaria 2023-24 (Proyecto Ejercicio9.circ). Partiendo de la UPG del fichero Ejercicio9.circ, queremos reducir el tamaño de su palabra de control mediante dos ajustes en su diseño. El primero es muy sencillo, y consiste en la eliminación de la entrada IMMED (no se aceptan operaciones con inmediatos). El segundo consiste en cambiar el banco de registros original (4 registros, 1 puerto escritura y 2 de lectura) por dos bancos con dos registros cada uno (1 puerto de escritura y 1 de lectura). Los datos de entrada (DATAIN) solamente se pueden escribir en uno de los bancos (Regfile-0), mientras que el resultado de las operaciones de la ALU sólo se almacena en los registros del otro banco. La salida de resultados se hace a través del banco Regfile-1. La figura inferior muestra un esquema del cambio.



Lleva a cabo las siguientes tareas:

1. Modifica la UPG proporcionada para llevar a cabo los cambios solicitados. Guarda tu nueva UPG como un nuevo circuito de nombre **UPGv2**.
2. Determina cómo quedaría la nueva palabra de control para tu UPG, indica tamaño y campos de la misma.
3. Si el camino crítico de la UPG es de tipo FF-FF, calcula cuánto se reduce el tiempo de ciclo con la nueva implementación. Utiliza las implementaciones de la figura inferior y los siguientes retardos: Not: 10 **ns**, And/Or: 20**ns**, Xor: 50**ns**, Flip-Flop: 100**ns**
4. Dado el algoritmo inferior, indica los cambios necesarios para que se pueda ejecutar en tu nueva UPG y determina cómo quedarían sus palabras de control.

   [0] Store DATAIN in R1
   [1] R0= R1 + R2
   [2] Print Result (R0)



Ejercicio 10. Extraordinaria 2023-24 (Proyecto Ejercicio10.circ). Partiendo de la UPG del fichero Ejercicio10.circ, lleva a cabo las siguientes tareas:

(Parte 1) determina las palabras de control necesarias para realizar la resta de dos valores (A-B) que entran por DATAIN en los dos primeros ciclos de reloj.

0 [IN R0] *Store A in R0*
1 [IN R1] *Store B in R1*
…Completar…

(Parte 2) Implementa una Unidad de Control específica mediante una ROM (ROM-UC) que lleve a cabo dicha operación y conecta dicho circuito con tu UPG. Guarda tu circuito resultante como **SolEj10**