

# Práctica 2 - Lógica Digital

## Organización del Computador 1

### Primer Cuatrimestre 2019 - Turno Tarde

Todas las compuertas mencionadas en esta práctica son de 1 ó 2 entradas, a menos que se indique lo contrario. Usaremos los símbolos detallados a continuación para representar las distintas funciones lógicas: XOR  $\rightarrow \oplus$ , NAND  $\rightarrow \downarrow$ , NOR  $\rightarrow \downarrow$ .

Durante la presente práctica se recomienda fuertemente la utilización de un simulador para experimentar con los componentes y circuitos propuestos y verificar las soluciones. Una recomendación es el Logisim (<http://www.cburch.com/logisim/>).

## Circuitos Combinatorios

**Ejercicio 1** Demostrar la equivalencia de las siguientes fórmulas booleanas:

a)  $p = (p.q) + (p.\bar{q})$

b)  $x.z = (x + y).(x + \bar{y}).(\bar{x} + z)$

**Ejercicio 2**<sup>1</sup> Sea  $p \oplus q = (\bar{p}.q) + (p.\bar{q})$ . Demostrar si la siguiente propiedad de distributividad es verdadera o es falsa:

$$x \oplus (y.z) = (x \oplus y).(x \oplus z)$$

**Ejercicio 3** Determinar la veracidad o falsedad de las siguientes afirmaciones:

a) Sea  $p|q = \bar{p}.\bar{q}$  ¿Alcanza con este operador para representar todas las funciones booleanas?

b) Sea  $p \downarrow q = \overline{p + q}$  ¿Alcanza con este operador para representar todas las funciones booleanas?

**Ejercicio 4** Mostrar cómo se puede construir la función booleana  $f(A, B) = A.B$  a partir de 2 compuertas NAND. Mostrar además cómo construir la función  $f(A) = \bar{A}$  utilizando únicamente compuertas NAND.

**Ejercicio 5** Dibujar un circuito que implemente la función booleana  $f(A, B, C) = A.B.C$  usando 2 compuertas NOR y varias compuertas NOT.

**Ejercicio 6** Dada la función booleana  $F$  definida a partir de la siguiente tabla de verdad,

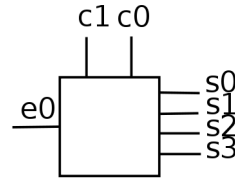
$A$	$B$	$C$	$F(A, B, C)$
1	1	0	0
1	0	0	1
1	0	1	1
0	1	0	0
0	0	1	0
0	1	1	1
1	1	1	1
0	0	0	0

<sup>1</sup>Ej 7. Capítulo 3 del L. Null & J. Lobur - Essentials Of Computer Organization And Architecture

- a) Escribir la *suma de productos* para la función  $F$ . Calcular la cantidad de compuertas que la implementación literal de la función requeriría.
- b) ¿Se puede simplificar la expresión usando propiedades del álgebra booleana? Dibujar el circuito correspondiente utilizando la menor cantidad de compuertas que pueda.

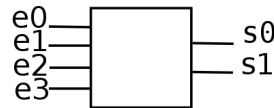
**Ejercicio 7** Dibujar el diagrama lógico de un *demultiplexor* de 2 líneas de control, 1 línea de entrada y 4 líneas de salida. Este circuito dirige la única línea de entrada a una de cuatro líneas de salida, dependiendo del estado de las dos líneas de control.

$c_1$	$c_0$	$s_i$
0	0	$s_0 = e_0, s_i = 0$ si $i \neq 0$
0	1	$s_1 = e_0, s_i = 0$ si $i \neq 1$
1	0	$s_2 = e_0, s_i = 0$ si $i \neq 2$
1	1	$s_3 = e_0, s_i = 0$ si $i \neq 3$



### Ejercicio 8

- a) Dibujar el diagrama lógico de un *codificador* de 4 líneas de entrada ( $e_i$ ) y 2 líneas de salida ( $s_i$ ). Si únicamente  $e_i$  está alta, las salidas deben representar el número  $i$  en notación sin signo. No está definido cuál es el resultado si no se cumple que sólo una de las líneas de entrada tiene valor 1.



- b) Dotar al circuito anterior de una salida adicional que indique si el estado de la entrada es válido o inválido.

### Ejercicio 9

- a) Dibujar con compuertas lógicas el circuito de un *decodificador* de 2 líneas de entrada ( $e_i$ ) y 4 líneas de salida ( $s_i$ ), cuya tabla de verdad es la siguiente:

$e_1$	$e_0$	$s_3$	$s_2$	$s_1$	$s_0$
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0

- b) Usando el circuito anterior, reescribir el *demultiplexor* de 1 línea de entrada, 2 líneas de control y 4 líneas de salida.

**Ejercicio 10** Un *carry left shifter 3-4* es un componente de 3 líneas de entrada ( $e_2, e_1, e_0$ ), 4 líneas de salida ( $s_3, s_2, s_1, s_0$ ) y un línea de control ( $c_0$ ) que se comporta de la siguiente manera:

- si  $c_0 = 0$ ,  $s_i = e_i$  para todo  $0 \leq i < 3$ ,  $s_3 = 0$
- si  $c_0 = 1$ ,  $s_{i+1} = e_i$  para todo  $0 \leq i < 3$ ,  $s_0 = 0$

- a) Dibujar el diagrama lógico de un carry left shifter 3-4.
- b) Si las líneas de entrada y de salida codifican un número entero en notación sin signo, ¿qué significa matemáticamente el shift a la izquierda? ¿Y a la derecha?

### Ejercicio 11

- a) Diseñar un *full adder* de 1 *bit*. La tabla de verdad del *full adder* es la siguiente:

$A$	$B$	$carry_{in}$	$suma$	$carry_{out}$
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

- b) Suponiendo que todas las compuertas elementales tienen el mismo retardo (*delay*)  $t$ , calcule el retardo total del circuito para producir todas sus señales de salida.

### Ejercicio 12

- a) Diseñar un *full adder* de 4 *bits* combinando 4 *full adders* de 1 *bit*.  
b) Suponiendo que los enteros se codifican con notación complemento a 2. Diseñar circuitos anexos que observen los siguientes *flags*:

Negative:  $N = 1 \iff$  la salida representa un número negativo (complemento a 2)

Overflow:  $V = 1 \iff$  el resultado no es representable (complemento a 2)

Carry:  $C = 1 \iff$  la suma de la codificación binaria produjo acarreo

Zero:  $Z = 1 \iff$  el resultado representa el número 0

- c) ¿Se puede usar el *mismo* circuito para sumar números enteros codificados en notación sin signo?  
d) ¿Cómo se podría aprovechar este sumador para realizar restas tanto de números codificados en notación complemento a 2 como en notación sin signo?  
e) Modificar la señal de *Carry* de tal modo que represente lo siguiente:
- Si la operación es suma:  $C = 1 \iff$  la suma *bit a bit* produjo acarreo
  - Si la operación es resta:  $C = 1 \iff$  la resta *bit a bit* produjo borrow (dame uno)
- f) Para comparar dos números A y B se realiza la operación  $A - B$ . Indicar los valores de los flags que caracterizan las siguientes condiciones.

condición	complemento a 2	notación sin signo
$A < B$		
$A \leq B$		
$A = B$		
$A \geq B$		
$A > B$		

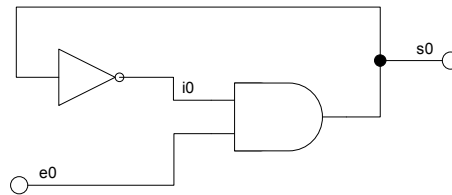
### Ejercicio 13

- a) Diseñar un componente con 4 entradas  $e_0, \dots, e_3$  y 4 salidas  $s_0, \dots, s_3$  tal que cada salida  $s_i$  valga  $\overline{e_i}$ .  
b) Diseñar un componente con 4 entradas  $e_0, \dots, e_3$  y 4 salidas  $s_0, \dots, s_3$  que calcule el inverso aditivo del número codificado en complemento a 2 por la entrada.  
c) Modificar el circuito anterior para que en una nueva salida indique si el número de la entrada no tiene un inverso aditivo representable con 4 *bits* en complemento a 2.

## Circuitos Secuenciales

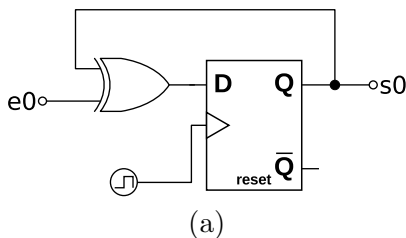
### Ejercicio 14

- a) Escribir el diagrama temporal para el siguiente circuito secuencial desde 0 ns hasta 65 ns, suponiendo
- un retardo de 15 ns para la compuerta AND,
  - un retardo de 5 ns para la compuerta NOT,
  - en el tiempo 0 ns la señal  $e_0$  cambia a 1, inicialmente en 0.
  - las señales  $i_0$  y  $s_0$  tienen valor 1 y 0 respectivamente en el tiempo 0 ns.
  - suponer que los componentes empiezan a estabilizarse cuando sus señales de entrada están estables.

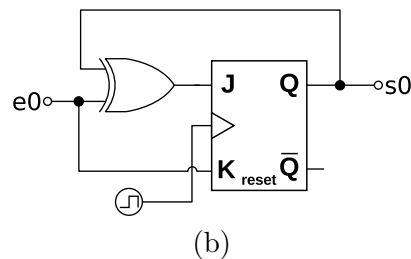


- b) ¿Podría alcanzar  $s_0$  un valor estable en el punto anterior? ¿Y en el caso en que  $e_0$  fuera 0 en lugar de 1, se estabilizaría?

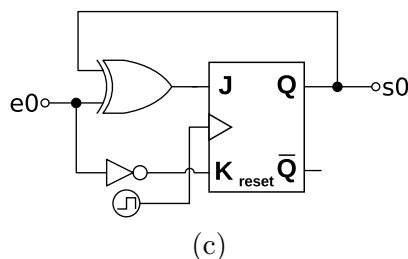
**Ejercicio 15** Escriba tablas características que especifiquen el comportamiento de cada uno de los siguientes circuitos secuenciales:



(a)

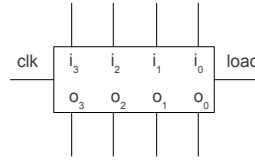


(b)

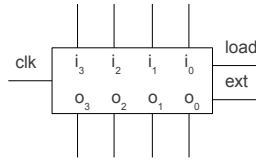


(c)

**Ejercicio 16 *Registro simple*.** Diseñar un registro *simple* de cuatro *bits*. Este tipo de registros es un circuito de seis entradas ( $i_0$  a  $i_3$ , *load*, *clk*) y cuatro salidas ( $o_0$  a  $o_3$ ), cuyo funcionamiento es el siguiente: cuando la señal *clk* alcanza su flanco ascendente, si *load* está alta, almacena las señales recibidas en  $i_0$  a  $i_3$ , si no, no cambia su contenido. Por las líneas de salida, se emite el valor almacenado en el registro.

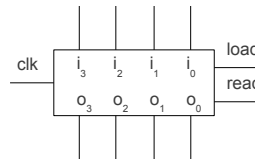


### Ejercicio 17 Extensor de signo

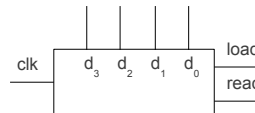


Diseñar un registro *extensor de signo* de dos a cuatro bits. Del mismo modo que un registro simple (ver ejercicio 16) este circuito toma el valor de sus cuatro entradas  $i_0$  a  $i_3$  cuando su señal  $clk$  atraviesa su flanco ascendente si la entrada  $load$  vale 1. Por sus líneas de salida ( $o_0$  a  $o_3$ ) se emite el valor almacenado si la señal  $ext$  está baja; por el contrario, si vale 1, se emite una representación de cuatro *bits* del número almacenado en los dos *bits* menos significativos del registro, interpretados como un entero codificado en complemento a 2.

**Ejercicio 18 Registro de salida restringida.** Diseñar un registro *de salida restringida* de cuatro bits. Este tipo de registros es un circuito de siete entradas ( $i_0$  a  $i_3$ ,  $load$ ,  $clk$  y  $read$ ) y cuatro salidas ( $o_0$  a  $o_3$ ), muy similar al registro simple (ejercicio 16) pero que sólo emite su salida por las líneas  $o_0$  a  $o_3$  si  $read$  está alta cuando  $clk$  alcanza su flanco ascendente. Dicha salida se debe mantener hasta el próximo flanco ascendente en  $clk$  donde  $read$  esté baja.



**Ejercicio 19 Registro bidireccional.** Diseñar un registro *bidireccional* de cuatro bits. Este tipo de registros es un circuito con tres entradas ( $load$ ,  $read$ ,  $clk$ ) y cuatro señales de entrada y salida ( $d_0$  a  $d_3$ ). Su funcionamiento es el siguiente: si la señal  $load$  vale 1 cuando  $clk$  alcanza su flanco ascendente, almacena los valores recibidos en  $d_0$  a  $d_3$ ; en cambio, si  $read$  está alta, se emite el valor almacenado en el registro por esas mismas líneas<sup>2</sup>. Las señales  $read$  y  $load$  nunca valen 1 simultáneamente.

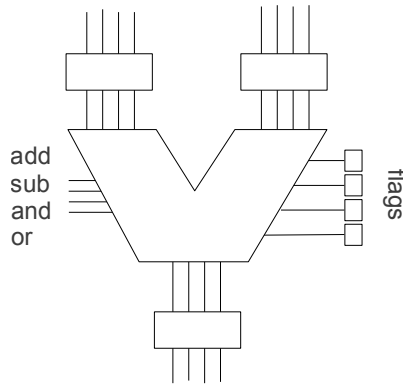


### Ejercicio 20 ALU

Diseñe una ALU con las siguientes características:

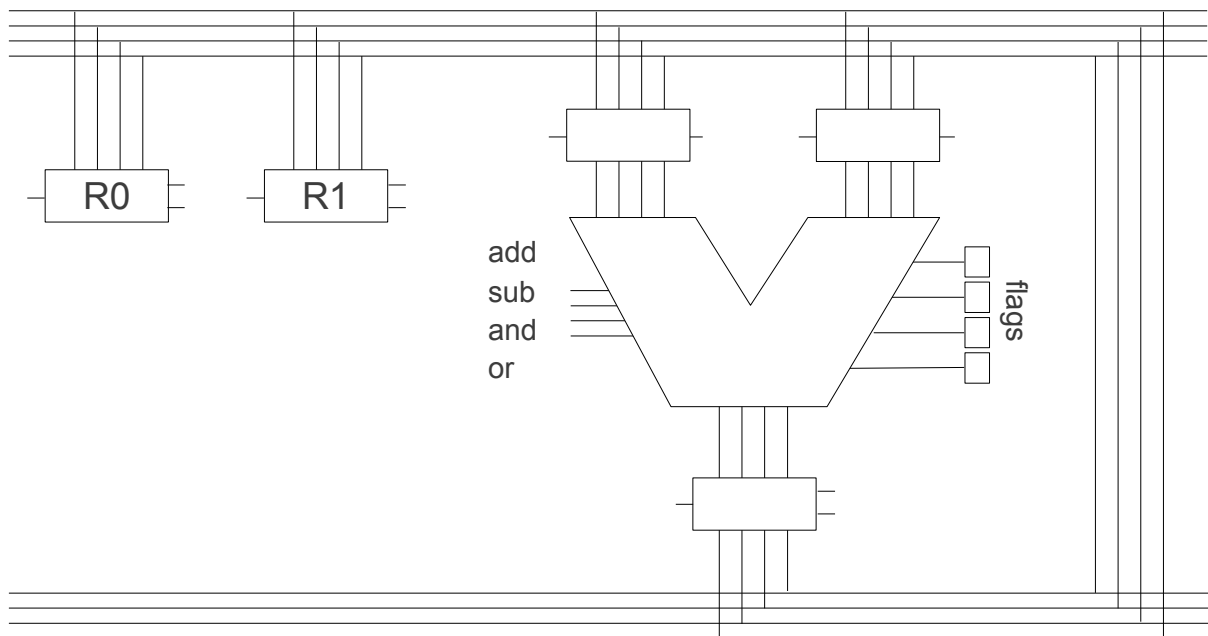
- cuatro señales de entrada que indican la operación a realizar: *add*, *sub*, *and*, *or*;
- dos registros simples, donde se almacenan los operandos a utilizar;
- un registro de salida restringida, donde se almacena el resultado;
- cuatro *flags* cuyos valores son determinados por la última operación realizada.

<sup>2</sup> Ayuda: utilice componentes de tres estados.



### Ejercicio 21

Dado el siguiente circuito, indique mediante un diagrama de tiempos la secuencia de activaciones y desactivaciones de señales de control necesarias para que el valor almacenado en el registro bidireccional (ejercicio 19) R0 se sume al valor del registro bidireccional R1 y el resultado se almacene en el registro R0.

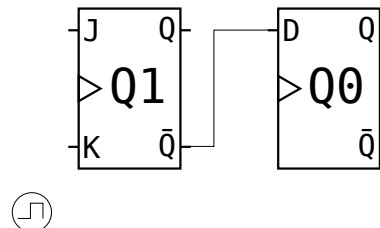


### Ejercicios tipo parcial

**Ejercicio 22** Para el presente ejercicio puede utilizar los siguientes componentes: SUMADOR COMPLETO de 1 *bit*, compuertas lógicas de 2 entradas y negadores de 1 *bit*. Está prohibido utilizar componentes de 3 estados.

- Armar un componente que tenga como entrada 1 número binario ( $A$ ) en complemento a 2 de 3 *bits* y que calcule su inverso aditivo ( $-A$ ), en caso de que exista.
- Armar un componente que tenga como entrada 2 números binarios ( $A$  y  $B$ ) en complemento a 2 de 3 *bits* y que calcule  $A - B$ . Además se pide que el componente tenga 4 salidas adicionales correspondientes a los flags Z, C, V y N con su interpretación habitual.
- Dada la siguiente tabla que se encontró incompleta y el circuito también incompleto, completarlos para que sean consistentes.

$Q_1(t)$	$Q_0(t)$	$Q_1(t+1)$	$Q_0(t+1)$
0	0	1	?
0	1	0	?
1	0	0	?
1	1	0	?



d) ¿El circuito anterior cambia su valor a cada clock? ¿O para alguna entrada es estable?

**Ejercicio 23**<sup>3</sup> Una fábrica de lavarropas nos pide el desarrollo de la circuitería para su nuevo modelo super-económico, con una única función de lavado de treinta minutos de duración.

Los elementos con los que debe contar el lavarropas son los siguientes:

- Una llave de encendido: **START/PAUSE** y un botón de **RESET**.
- Una pantalla que muestra el tiempo restante en minutos.
- Un sistema de seguridad por sobrepeso que no permite arrancar en caso de sobrecarga.

El comportamiento es el siguiente: el lavarropas se encuentra listo para funcionar al ser enchufado, por ello muestra un 30 en su pantalla. Al accionar la llave de **START**, si el sistema no está sobrecargado, el contador debe ir disminuyendo hasta llegar a 0, con la salida  $S_0$  activa. El usuario puede deshabilitar la llave de **START** para cargar más ropa. Luego, al re-activarse debe continuar desde donde se interrumpió. Al finalizar debe apagarse la salida  $S_0$ . En caso de tener peso de más, no debe arrancar o continuar luego de ser interrumpido. El sensor de sobrepeso mantiene una salida alta mientras el peso supere el límite establecido. El botón de **RESET** debe regresar el estado del lavarropas al estado inicial, **sólo** en caso de no estar andando.

La empresa posee una amplia experiencia en este tipo de circuitos por lo que posee varias cosas desarrolladas previamente. Entre ellas, podemos nombrar:

- Una pantalla de dos dígitos que muestra el número ingresado por las 6 entradas que posee, interpretadas como un número sin signo.
- Flip-Flops D con entradas de *set* y *reset*.
- Circuitos Contadores de 16 bits, con *reset*.
- Clock de 100HZ, con entrada de *enable*.

Se pide:

- Construir un *registro restador* de 6 bits, con entrada de *clock* y *reset* que lo pone en el valor 30.<sup>4</sup>
- Construir un circuito combinatorio de 16 entradas y una salida, tal que la salida se active cuando el número recibido sea  $(6000)_{10}$ .
- Construir el circuito del lavarropas solicitado.

**Ejercicio 24** La conjetura de Collatz, es un famoso problema matemático aún no resuelto. Esta conjetura enuncia la siguiente función  $f : \mathbb{N} \mapsto \mathbb{N}$ , aplicable a cualquier número entero positivo:

$$f(n) = \begin{cases} \frac{n}{2} & \text{si } n \text{ es par} \\ 3n + 1 & \text{si } n \text{ es impar} \end{cases}$$

Se dice que si se toma cualquier número y se aplica esta función reiteradas veces, el resultado siempre converge a 1.

<sup>3</sup>Ejercicio tomado en el primer parcial del verano de 2010.

<sup>4</sup>Ayuda: vale usar un *full-adder* de 6 bits.

- a) Construir un circuito combinatorio que realice la función  $f(n)$  para una entrada de 5 bits.
- b) Construir un circuito secuencial, que aplique reiteradas veces la función anterior por cada ciclo de reloj.
- c) Modificar el circuito anterior de forma que si el valor de entrada es 1, entonces la salida también sea 1.