Nro. ord.	Apellido y nombre	L.U.	#hojas
1	Kruel Magali	1257/23	4

SISTEMAS DIGITALES - **Primer Parcial**

Segundo Cuatrimestre 2024

Ej.1	Ej.2	Ej.3	Ej.4	Nota
3	1	1.5	4	9.5

Correctorx: JUANI ALVAREZ COLOMBO

Aclaraciones

- Anote apellido, nombre, LU y numere *todas* las hojas entregadas, entregando los distintos ejercicios en hojas separadas.
- El parcial **no es a libro abierto**, justifique sus respuestas.
- El parcial se aprueba con 6 y se deben tener ambos parciales aprobados para aprobar la materia (promoción directa).

Ejercicio 1 (3 pts.) Responder y justificar la respuesta:

- ¿Cuál es el rango de representación de un número de cuatro bits en signo y magnitud? ¿Cuál es el rango de representación de un número de ocho bits en complemento a dos?
- ¿Cómo se calcula el inverso aditivo de un número n en complemento a dos de k bits?
- Para dos números a y b de k bits, para una operación de suma cuyo resultado nombramos c, ¿cómo se determina el `carry`, sobre qué tipo de datos lo observamos? ¿Cómo se determina el `overflow`, sobre qué tipo de datos lo observamos? Explique por qué se definen de esta manera.

Ejercicio 2 (1 punto) Responder:

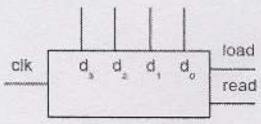
1. Sea $p|q = \overline{p.q}$ ¿Alcanza este único operador (`NAND`) para representar todas las funciones booleanas?
2. Sea $p \downarrow q = \overline{p+q}$ ¿Alcanza este único operador (`NOR`) para representar todas las funciones booleanas?

Consejo: recuerde que las operaciones de conjunción (`AND`), disyunción (`OR`) y negación son suficientes para representar todas las funciones booleanas.

Ejercicio 3 (2 pts.) Dibujar circuitos que implementen las siguientes funciones booleanas:

1. $f(A, B, C) = A.B.C$ usando 2 compuertas NOR y varias compuertas NOT.
2. $f(A, B) = \overline{((A.B) + (\overline{B}.A))}.\overline{B}$ ¿Para qué valores de A y B la función devuelve un 1?

Ejercicio 4 (4 pts.) ***Registro bidireccional*** Diseñar un registro *bidireccional* de cuatro bits. Este tipo de registros es un circuito con dos señales de control de entrada (load, read, el clk) y cuatro señales de entrada y salida de datos (d_0 a d_3). Su funcionamiento es el siguiente: si la señal load vale *1* cuando clk alcanza su flanco ascendente, almacena las 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[1]. Las señales read y load nunca valen *1* simultáneamente.



[1] *Ayuda:* utilice componentes de tres estados.

Kruel Magali	LU: 1257/23
	Hoja Nº: 1

Ejercicio 1

1/1 ▪ El rango de representación de un número de 4 bits en signo + magnitud es $[-2^{4-1}+1 ; 2^{4-1}-1] = [-7 ; 7]$. ✓

El rango de representación de un número de 8 bits en complemento a dos es = $[-128 ; 127]$ ✓

1/1 ▪ El inverso aditivo de un número n que ya está escrito en complemento a dos se calcula como inv(n-1) ✓

1/1 ▪ El carry se determina mirando el (k+1)-ésimo bit de c. Para sumar 2 números de k bits, hago la suma bit a bit, y si esta suma se saliera del rango de representación, es decir que es una unidad mayor, se "pasa" 1 unidad a los bits a su izquierda, y esta unidad se llama carry. ✓ Cuando estamos en la suma de los primeros bits (los más significativos de a y b), si esta llegase a dar carry, no puedo "pasarlo" a sus vecinos izquierdos porque no tienen, y luego debo ponerlo junto con el resultado c.

El overflow depende de cómo esten representados a y b:

↳ Para sin signo, el carry determina el overflow, ✓ ya que, como explique anteriormente, que haya carry indica que la suma se sale del rango de representación. (con Carry me refiero al CarryOut, o el carry de salida).

↳ Para complemento a dos, usamos lo siguiente.

↳ Si dos números con mismo signo devuelven un número con el signo opuesto, hay Overflow. ✓ Esto se encuentra mirando los bits más significativos de cada número. Luego,

$\ominus + \ominus = \oplus \Rightarrow$ Overflow $\quad \ominus + \oplus = \oplus \Rightarrow$ OK

$\oplus + \oplus = \ominus \Rightarrow$ Overflow $\quad \ominus + \oplus = \ominus \Rightarrow$ OK

Kruel Magali | LU = 1257/23
Hoja N° = 2

Ejercicio 2

0.5/0.5 ① Sí, alcanza este único operador. Para justificarlo, reescribo las operaciones AND, OR y la negación:

* negación: $\overline{A} \overset{\text{idempotencia}}{=\!=} \overline{A.A}$ $\approx \overline{p.q}$, $p = A$, $q = A$
* AND: $A.B = \overline{\overline{A.B}} \underset{\text{De Morgan}}{=\!=} \overline{\overline{A}+\overline{B}} \overset{\text{identidad}}{=\!=} \overline{1.(\overline{A}+\overline{B})}$ $\approx \overline{p.q}$, $p = 1$, $q = \overline{A}+\overline{B}$
* OR: $A+B = \overline{\overline{A+B}} \underset{\text{De Morgan}}{=\!=} \overline{\overline{A}.\overline{B}}$ $\approx \overline{p.q}$, $p = \overline{A}$, $q = \overline{B}$

0.5/0.5 ② Sí, alcanza este único operador. Para justificarlo, reescribo las operaciones AND, OR y la negación:

* negación: $\overline{A} \overset{\text{idempotencia}}{=} \overline{A+A}$ $\approx \overline{p+q}$, $p = A$, $q = A$
* AND: $A.B = \overline{\overline{A.B}} \underset{\text{De Morgan}}{=} \overline{\overline{A}+\overline{B}}$ $\approx \overline{p+q}$, $p = \overline{A}$, $q = \overline{B}$
* OR: $A+B = \overline{\overline{A+B}} \underset{\text{De Morgan}}{=} \overline{\overline{A}.\overline{B}} \underset{\text{identidad}}{=} \overline{0+(\overline{A}.\overline{B})}$ $\approx \overline{p+q}$, $p = 0$, $q = \overline{A}.\overline{B}$

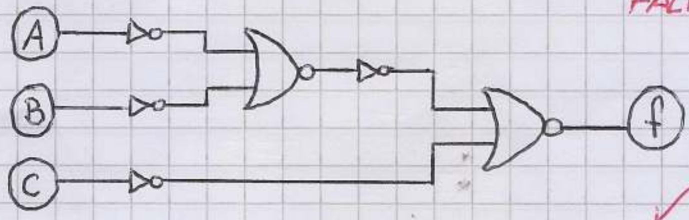
Kruel Magali | LU: 1257/23 | Hoja Nro: 3

Ejercicio 3

0.5/1

1)

FALTA LA JUSTIFICACIÓN



para $f(A,B,C) = A.B.C$

✓

2) Veamos para qué valores la función devuelve 1 con su tabla:

1/1

A	B	$\overline{B}$	$A.B$	$\overline{B}.A$	$(A.B)+(\overline{B}.A)$	$\overline{(A.B)+(\overline{B}.A)}.\overline{B}$
0	0	1	0	0	0	1
0	1	0	0	0	0	0
1	0	1	0	1	1	0
1	1	0	1	0	1	0

$\Rightarrow \overline{A}.\overline{B}$ devuelven 1

Veamos que $\overline{A}.\overline{B} = \overline{A+B}$, por lo que basta con una compuerta NOR

✓

FALTA LA RESPUESTA: $A = 0$ y $B = 0$

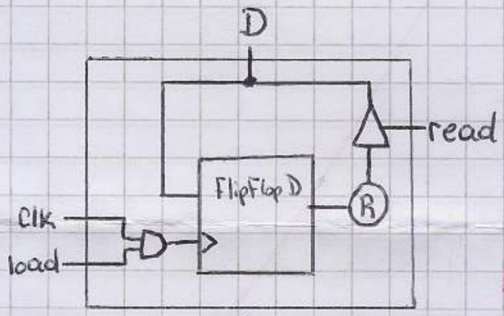
Kruel Magali | LU: 1257/23 | Hoja Nº: 4

4/4

Ejercicio 4

Por cuestiones de prolijidad, haré un circuito simple y luego lo extenderé a 4 bits.

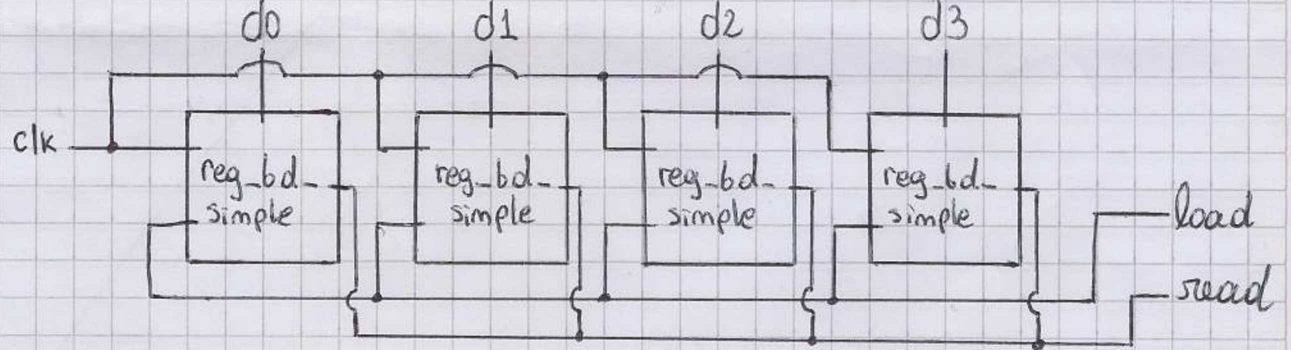
Notemos que necesitamos un registro que guarde un bit cuando clk = 1↑ y load = 1, y que retorne el valor guardado cuando read = 1. Para eso utilizo un registro FlipFlopD y un componente de 3 entradas que controle la salida del valor:



reg_bd_simple

donde Ⓡ es el bit almacenado. Llamo "reg_bd_simple" a este circuito para reutilizarlo.

Ahora sí, mi registro bidireccional de 4 bits es:



Notar que se usan el mismo clk, load y read para los cuatro registros para que operen al mismo tiempo.