Taller de Logica Digital- Parte 2

April 24, 2025

1 Ejercicios

1.1 Componentes de 3 estados

a. Completar la siguiente tabla:

A	A_en	B	B_en	C	C_en	*Estimado*	*Obtenido*
0	0	0	0	0	0	0	undefined
0	1	1	1	0	0	error	error
1	0	1	0	1	0	undefined	undefined
1	1	1	0	1	1	error	error
0	1	0	1	0	1	0	0
0	1	1	1	1	1	error	error
1	0	1	1	1	0	1	1

b. Completar la siguiente tabla:

Color	Interpretación
Gris	El cable no esta definido
Verde Claro	Representa al bit 1
Verde Oscuro	Representa al bit 0
Azul	Esta conectado pero no definido
Rojo	Hay conflito de bits

c. Enunciar la regla:

Para no tener conflictos de bits no debemos activar al mismo tiempo variables de control que dejen pasar dos datos distintos.

d. Explicar cuales son y por que:

Los casos basuras son aquellos donde un cable o algo (entrada, salida, componente, etc) que espera recibir solo un valor obtiene dos distintos, por lo que no se puede extraer ningun resultado

1.2 Transferencia entre registros:

a. Detallar entradas y salidas:

Nombre	Entrada/Salida	Control si/no	Descripcion
Clk	Entrada	No	Representa el valor a alamacenar en los flip-flop D
$Force_{input}$	Entrada	Si	Habilita el almacenado de valores en los 3 flip-flop D
$W_i(i \in 1,2,3)$	Entrada	Si	Habilita el almacenado de valores en el i-esimo flip-flop D
Entradas conectadas al input en_{out}	Entrada	Si	Habilitan la salida del valor almacenado en el flip-flop D correspondiente
en_force_input	Entrada	Si	Regula el paso del valor que entra en force_input
$R_i(i \in 0,1,2)$	Salida	No	Representan los valores almacenados en cada uno de los flip-flop D
Salida General	Salida	No	n el caso de que no haya error devuelve los valores almacenados en el/los R_i y/o en el force_input

b. Secuencia de señales:

Force_Input (1) → en_Force_input (1) → w_1 (1) → clk (1)

En caso de que se desee alterar componendes, esta es la secuencia de pasos para no modifique a "registro_1": w_i (0). (deshabilitamos el write en R1)

c. Secuencia de señales:

en_force_input (1) (no nos importa el valor que tenga "Force_input" previamente (0 o 1) →
w_0 (1)→
clk (1) →
clk (0) →
en_Force_input (0) →
en_out (1) (en "r_0") →
w_1 (1) →
clk (1) →
clk (0) →
en_out (en "R_0") (0) →
w_1 (0) →
en_out (en "R_1") (0) →
en_out (en "R_0") (0) →
en_out (en "R_2") (1) →
clk (1) →
clk (0) →
w_0 (0) →
w_1 (1) →
en_out (en "R_2") (0) →
en_out (en "R_1") (1)→
w_2 (1) →
clk (1) →
clk (0)

1.3 Maquina de 4 registros con suma y resta

a. Detallar entradas y salidas:

Nombre	Entrada/Salida	Control si/no	Descripcion
Force_input	Entrada	No	Ingresa el valor que queremos almacenar en alguna de las memorias
en_force_input	Entrada	Si	Habilita o no el paso del valor de Force_input
Reg_i_Write ($i \in 0, 1, 2, 3$)	Entrada	Si	Permite o no en cada memoria el guardado de un valor
$Reg_i\_enableOut$ ($i \in 0, 1, 2, 3$)	Entrada	Si	Permite o no la salida del valor almacenado en cada memoria
ALU_N_Write ($N \in (A, B)$)	Entrada	Si	Permite o no almacenar el valor de un registro (o varios con el mismo valor) en la ALU
OP	Entrada	Si	Indica a la ALU que operacion realizar entre +,- ,and, or
ALU_enableOut	Entrada	Si	Permite la salida del valor resultante de la operacion realizada por la ALU
Clk	Entrada	Si	Prmite almacenar valores en los registros y realizar operaciones en la ALU, al oscilar entre 0 y 1. Tambien de regular las salidas de los registros y la ALU al cambiar de valor
N_debug ($N \in (A, B)$)	Salida	No	Permite ver en todo momento el valor almacenado en las entradas correspondientes al valor a o b de la ALU
S_debug	Salida	No	Permite ver en todo momento el resultado de la operacion realizada por la ALU
N	Salida	No	Indica si el resultado de la operacion de la ALU fue negativo en complemento a 2
Z	Salida	No	Indica si el resultado de la operacion de la ALU fue 0
V	Salida	No	Indica si el resultado de la operacion de la ALU tuvo overflow en la suma interpretada en complemento a 2 (0 sino)
C	Salida	No	Indica si el resultado de la operacion de la ALU tuvo carry en la suma y la resta (0 sino)

b. Detallar el contenido de cada display:

Display	Descripcion
Reg4_debug (display de cada registro)	Permite ver en todo momento el valor almacenado en un registro
Display inferior	Esta conectado a un cable que contiene el valor tanto del resultado de la operacion de la ALU, como de la entrada force_input, por lo que siempre que no haya conflicto con valores, nos lo mostrara.

c. Secuencia de señales:

Primero de todo suponemos todos los valores seteados a 0. Dicho esto, a continuacion la secuencia:

Seteamos (0100) en Force_input →
en_Force_input (1) →
Reg2_write (1) →
clk (1) →
clk0 →
Reg2_write (0) →
Seteamos (1101) en Force_input →
Reg3_write (1) →
clk (1) →
clk (0)

d. Completar la siguiente tabla:

Valor Inicial	Resultado op 1	Flags	Resultado op 2	Flags
(4,0)	4 en ambas interpretaciones	N=0, Z=0, V=0, C=0	4 en ambas interpretaciones	N=0, Z=0, V=0, C=0
(7,-1)	8 en sin signo y -8 en complemento a 2	N=1, Z=0, V=1, C=1	7 en ambas interpretaciones	N=0, Z=0, V=0, C=0
(-8,-2)	6 en ambas interpretaciones	N=0, Z=0, V=1, C=1	10 en sin signo y -6 en complemento a 2	N=1, Z=0, V=1, C=1
(8,-9)*	N/A	N/A	N/A	N/A

*No se pueden realizar operaciones con estos valores, pues -9 no se puede escribir en complemento a 2 ya que este funciona en el rango [7,-8]

Veamos a continuacion la secuencia de operaciones para almacenar en R2 y R3 los resultados de OR y sub con (4,0) como valores:

Seteamos (0100) en force_input →
en_force_input(1) →
Reg0_write(1)→
clk(1)→
clk(0)→
Reg0_write(0) →
Seteamos (0000) en force_input →
Reg1_write(1)→
clk(1)→
clk(0)→
Reg1_write(0)→
ALU_A_Write(1)→
Reg0_enable_out(1)→
clk(1)→
clk(0)→
Reg0_enable_out(0) →
ALU_A_Write(0)→
ALU_B_Write(1)→
Reg1_enable_out(1)→
clk(1)→
clk(0)→
Reg1_enable_out(0)→
ALU_B_Write(0)→
op 11 →
clk(1)→
clk(0)→
en_force_input(0) →
ALU_enable_out(1)→
Reg2_write(1)→
clk(1)→
clk(0)→
Reg2_write(0)→
op 01 →
clk(1)→
clk(0)→
ALU_enable_out(1)→
Reg3_write(1)→
clk(1)→
clk(0)→
Reg3_write(0)

e. Explicar

Se niega la salida del clock para que en el mismo ciclo de encendido y apagado la ALU realice la operacion (clk 1) y el registro_4bytes devuelva el resultado (clk 0, interpretado como 1 por el registro), bajando el tiempo de respuesta del componente, ya que si dicho NOT no estuviera negando al clk, entonces se neceseterian dos ciclos del mismo para cargar dos valores (suponiendo write 1 en ambos) y devolver el resultado.

2 Correccion

Integrantes:

- Nombre y apellido: Agustin Stescovich Curi LU: 184/24
- Nombre y apellido: Franco Nahuel Chazarreta Villagra LU: 669/24
- Nombre y apellido: Jasson Aldayr, Davila Bustamante LU: 59/22