TP2 - Lógica Secuencial: Grupo 5 (Linkin Park)

Sistemas Digitales

Primer Cuatrimestre 2025

Integrantes: Facundo Otero Zappa, Leonardo Domínguez y Juan Manuel Bailleres.

Ejercicio 1: Componentes de 3 estados:

A) Completar la tabla:

A	A_en	B	B_en	C	C_en	Estimado	Obtenido
0	0	0	0	0	0	Hi-Z	U (High-Z)
0	1	1	1	0	0	Error	E (Error)
1	0	1	0	1	0	Hi-Z	U
1	1	0	0	0	1	Error	E
0	1	0	1	0	1	Error	0
0	1	1	1	1	1	Error	E
1	0	1	1	1	0	1	1

B) Completar la tabla:

Color	Interpretación
Gris	Cuando el cable está sin señal (no conectado a nada).
Verde claro	Cuando hay flujo de corriente (en alta).
Verde Oscuro	Cuando el cable está conectado, pero no hay flujo de corriente sobre el circuito (en baja).
Azul	Cuando el circuito tiene alta impedancia (Hi-Z ó U).
Rojo	Cuando el circuito tiene un error (2 valores lógicos distintos en un mismo bus)

C) Enunciar la regla:
La regla para que el circuito nunca se ponga en rojo consiste en que no esté activado (en “enable”) más de un tri-buffer simultáneamente al mismo bus.
Aún así hay casos que en la práctica no son deseables, como el caso 010101 del ítem A). Así que para ser más precisos, para que no este el circuito en rojo, no debe haber 2 o más tri-buffer activados y que tengan salidas lógicas distintas en el mismo bus.

D) Explicar cuáles son y por qué:
Las combinaciones que se detectan como basura y que en el simulador parecen no serlo, son todos los casos en los que más de un "enable" está funcionando (está activado) y comparten la misma salida.
Lo ideal es que cuando uno de los componentes A, B o C esté funcionando, el resto estén en Hi-Z.

Ejercicio 2: Transferencia entre registros:

A) Detallar entradas y salidas:

El componente llamado ej-2 está compuesto en total por 9 entradas y 4 salidas.

Entradas:

- Un clk (clock) general que funciona periódicamente enviando una onda cuadrada.
- Una entrada Force_Input, que determina, cuando se prenda en_Force_Input, si se encenderán las entradas Reg-in de los Flip-flops. Estos valores se almacenarán en ellos y se transmitirán.
- La entrada en_Force_Input deja pasar la corriente de Force_input hacia los flip flops cuando vale 1 (cuando este activa).
- 3 entradas w's (w0, w1, w2) que son entradas de control que funcionan como señales de escritura para los registros correspondientes. Se almacena el valor proporcionado por en valor de Reg_In cuando dicha entrada está activa y ocurre un flanco ascendente del clk.
- 3 en_out's (en_out0, en_out1, en_out2) son entradas de control que funcionan para que el valor almacenado en el registro se propague a las salidas correspondientes

Salida:

- 3 salidas R's (R0, R1, R2) son los valores de salidas de cada registro.
- Reg_output es el valor de salida general del circuito si los en_out's están activados

Las entradas que consideramos de control son el en_Force_Input general y los 3 w's y los 3 en_out's.

B) Secuencia de señales:

Una posible secuencia de activación para almacenar el valor 1 en R1:

-Permitimos el paso de corriente haciendo: en_Force_input = 1.
-Colocamos Force_Input = 1.
-Activamos w1 = 1 para la escritura en el Flip-Flop 1 (registro de salida restringida 1).
-Esperamos el flanco ascendente del clock (en este caso lo activamos a mano).

Una posible secuencia de desactivación (luego de haber efectuado la secuencia de activación):

-Primero desactivamos el w1 (w1=0).
-Después desactivamos el Force_Input (lo ponemos en 0) y el en_Force_Input = 0.

C) Secuencia de señales:

Cargar el valor en R0:
-Activamos en_Force_input = 1
- Activamos w0 = 1
- Esperamos el flanco ascendente del clk
- Desactivamos w0
- Desactivamos en_force_input

Transferir el valor de R0 a R1:
-Activamos en_out0 = 1 para propagar el valor almacenado en R0 a través del circuito
-Activamos w1 para almacenar del valor de R0 en R1
-Esperamos el flanco ascendente del clk
-Desactivamos w1
-Desactivamos en_out0.

Luego para transferir de R2 a R0 y de R1 a R2 realizamos los mismos pasos anteriores con sus respectivas entradas.
En general:

Transferir de Rx a Ry (siendo Rx y Ry entradas cualesquiera)
-Activamos en_outx = 1 para propagar el valor almacenado en Rx a través del circuito
-Activamos wy para almacenar del valor de Rx en Ry
-Esperamos el flanco ascendente del clk
-Desactivamos wy
-Desactivamos en_outx.

Ejercicio 3: Máquina de 4 registros con suma y resta:

A) Detallar entradas y salidas:

Las entradas son las siguientes:

- Un clk (clock) general que se activa periódicamente
- Force_Input que permite la entrada de un valor al circuito
- en_force_input que habilita el acceso del valor de Force_Input al circuito
- 4 Reg_Write's que al igual que antes habilitan la escritura en sus respectivos registros
- 4 Reg_enableOut's que permiten la salida del valor registrado en cada circuito
- ALU_A_Write y ALU_B_Write los cuales cargan los valores del Reg_enableOut activado a los operando A o B de la ALU
- OP que representan las operaciones de la ALU
- ALU_enableOut que habilita la salida de la ALU.

Las salidas son las siguientes:

- Reg4_output de cada registro correspondiente que sirven como debug pues muestran el contenido almacenado en dicho registro
- Los A_Debug y B_Debug que muestran el valor almacenado en los operandos de la ALU
- Las flags de la ALU
- S_Debug y reg4_Output que muestran el valor final de la ALU

B) Detallar el contenido de cada display:

Los displays que salen de R0,R1,R2,R3 muestran la salida debug del registro en notación sin signo, es decir, muestran lo que está guardado en el registro.

Las salidas que salen de la ALU muestran lo siguiente:
Los dos primeros display's muestran los números guardados en los registros internos de la ALU (o sea los operandos A y B), y el tercero el resultado de la operación (en representación sin signo).

Finalmente se encuentra el del bus que muestra la salida de la ALU cuando está activado el en_out_alu o el force_input, si es que se encuentra activado en_force_input.

C) Secuencia de señales:

Una posible secuencia de señales es la siguiente:

Cargar el valor 4 en R2:

- Escribimos Force_input = 0100 (4 en binario)
- Activamos en_force_input = 1
- Activamos reg2_write = 1
- Activamos el clk = 1 (esperamos el flanco ascendente)
- Desactivamos reg2_write, clk y en_force_input a 0

Cargar el valor -3 en R3:

- Escribimos Force_input = 1101 (-3 en binario complemento a dos)
- Activamos en_force_input = 1
- Activamos reg3_write = 1
- Activamos el clk = 1 (esperamos el flanco ascendente)
- Desactivamos reg3_write, clk y en_force_input a 0

D) Completar la siguiente tabla:

Valor inicial	Resultado operación 1	Flags	Resultado Operación 2	Flags
(4,0), OR, SUB	4	N = 0 Z = 0 V = 0 C = 0	4	N = 0 Z = 0 V = 0 C = 0
(7,-1), SUB, AND	8 (no representable 1000)	N = 1 Z = 0 V = 1 C = 1	7	N = 0 Z = 0 V = 0 C = 0
(-8,-2), ADD, SUB	6 (en realidad da -10, no representable)	N = 0 Z = 0 V = 1 C = 1	-6	N = 1 Z = 0 V = 0 C = 1
(8,-9), OR, AND	-9 no es representable		-9 no es representable	

Secuencia detallada para el primer caso

- **Cargamos 4 en R0**
    - Escribimos Force_input = 0100 (4 en binario)
    - Activamos en_force_input = 1
    - Activamos reg0_write = 1
    - Activamos el clk = 1 (esperamos el flanco ascendente)
    - Desactivamos reg0_write, clk y en_force_input a 0

- R1 ya tiene guardado 0 (en caso de no, realizamos la secuencia anterior en R1 y con Force_input = 0000)
- **OR**
    - Activamos OP = 11 para la opracion OR
    - Activamos Reg0_EnableOut
    - Activamos ALU_A_Write
    - Esperamos el flanco ascendente
    - Desactivamos ALU_A_Write
    - Desactivamos Reg0_EnableOut
    - Activamos Reg1_EnableOut
    - Activamos ALU_B_Write
    - Esperamos el flanco ascendente
    - Desactivamos ALU_B_Write
    - Desactivamos Reg1_EnableOut
    - Finalmente ALU_EnableOut
- **AND**
    - Solo cambiamos el OP a 01 o repetimos la secuencia anterior del OR en caso de no tener los operandos alamacenados

E) Explicar:

Se niega la señal CLK antes de llegar al registro de salida para que el valor de la ALU tenga tiempo de estabilizarse. Así se evita que el registro guarde un resultado incorrecto o intermedio. El pulso negado actúa como un retardo que asegura que se guarde el valor correcto después del cálculo.