Taller de lógica Digital - Parte 2

Organización del Computador 1

Primer Cuatrimestre 2023

Ejercicios

1) Componente de 3 estados

a) Completar la siguiente tabla:

A	Aen	B	Ben	C	Cen	Estimado	Obtenido
0	0	0	0	0	0	Z	Z
0	1	1	1	0	0	Conflicto	Conflicto
1	0	1	0	1	0	Z	Z
1	1	0	0	0	1	Conflicto	Conflicto
0	1	0	1	0	1	0	0
0	1	1	1	1	1	Conflicto	Conflicto
1	0	1	1	1	0	1	1

b) Completar la siguiente tabla:

Color	Interpretación
Gris	Circuito desconectado
Verde Claro	Entrada de control activa y el bit ingresante es 1
Verde Oscuro	Entrada de control activa y el bit ingresante es 0
Azul	Todas las entradas de control desactivadas
Rojo	Dos entradas de control activas con bits ingresantes distintos

c) Enunciar la regla:

Cuando las entradas de control estén activas, que los bits ingresantes sean del mismo valor.

d) Explicar cuales son y porque:

Si como valor de entrada ingresa Z al buffer ya que este no afecta al comportamiento del programa en Logisim, por lo que en un mismo nodo tener Z y un bit 1/0 no produce ningún problema. Otro caso basura es cuando por un nodo está ingresando el mismo dato repetido, ejemplo, dos entradas de control están activadas y por las dos ingresa el bit con el valor 1, el programa va a funcionar pero es innecesario tener el valor repetido 2 veces a la salida.

2) Transferencia entre registros

a) Detallar entradas y salidas:

- **Force_input**: Bit que se desea almacenar.
- **clk** : Señal de reloj de registro_salida_restrictiva. En conjunto con la señal w, permite almacenar el dato. Cuando clk genera un flanco de subida (pasa de 0 a 1), y w está habilitada, el valor presente en Force_input se almacena.
- **en_Force_input** (control): Señal de habilitación del Force_input. Permite cortar la señal (generando una salida en alta impedancia) o habilitarla, permitiendo el paso del valor ingresado hacía el registro.
- **w** (control): En cada registro de salida restringida, esta señal, junto al clk, determina si se debe almacenar el dato. Además, puede influir en la generación del flanco de subida que habilita el almacenamiento.
- **en_out** (control): Cada en_out controla la salida o no por el Reg_output del registro_salida_restringida.
- **R0 a R1**: Salida debug que muestra el bit almacenado en registro_salida_restringida.

b) Secuencia de señales:

Activación:

1 - Colocar el dato a almacenar (1) en **Force_input**.

2 - Habilitar **en_Force_input** para permitir el paso del valor hacia el registro.

3 - Establecer **w** en 1 en R1 para almacenar.

4 - Activar el **clk** (pasarlo de 0 a 1). Este flanco de subida, junto con las señales anteriores activas, hace que el valor de Force_input se almacene en R1.

Desactivación:

1 - Bakar **clk** a 0

2 - Bajar w de **R1** y **en_Force_input** a 0

c) Secuencia de señales:

Activación:

1 - Colocar el dato a almacenar (Valor arbitrario) en **Force_input**.

2 - Habilitar **en_Force_input** a 1 para permitir el paso del valor hacia el registro.

3 - Establecer **w** en 1 en R0 para almacenar.

4 - Activar el **clk** (pasarlo de 0 a 1). Este flanco de subida, junto con las señales anteriores. activas, hace que el valor de Force_input se almacene en R0.

Desactivación:

1 - Bakar **clk** a 0.

2 - Bajar **w** de R0 y **en_Force_input** a 0.

Activación:

1 - Establecer **en_out** de R0 en 1.

2 - Establecer **w** en 1 en R1 para almacenar.

3 - Activar el **clk** (pasarlo de 0 a 1). Este flanco de subida, junto con las señales anteriores. activas, hace que el valor arbitrario de **en_out** de R0 se almacene en R1.

Desactivación:

1 - Bakar **clk** a 0.

2 - Bajar **w** de R1 y **en_out** de R0 a 0.

Activación:

1 - Establecer **en_out** de R2 en 1.

2 - Establecer **w** en 1 en R0 para almacenar.

3 - Activar el **clk** (pasarlo de 0 a 1). Este flanco de subida, junto con las señales anteriores. activas, hace que el valor arbitrario de **en_out** de R2 se almacene en R0.

Desactivación:

1 - Bakar **clk** a 0.

2 - Bajar **w** de R0 y **en_out** de R2 a 0.

Activación:

1 - Establecer **en_out** de R1 en 1.

2 - Establecer **w** en 1 en R2 para almacenar.

3 - Activar el **clk** (pasarlo de 0 a 1). Este flanco de subida, junto con las señales anteriores.

4 - activas, hace que el valor arbitrario de **en_out** de R1 se almacene en R2.

Desactivación:

1 - Bakar **clk** a 0.

2 - Bajar **w** de R2 y **en_out** de R1 a 0.

3) Máquina de 4 registros con suma y resta.

a) Detallar entradas y salidas:

- **ALU_A_Write** (control): Permite o no la entrada de la variable A.
- **ALU_B_Write** (control): Permite o no la entrada de la variable B.
- **OP** (control): Indica qué operación va a hacer el ALU.
- **ALU_enableOut** (control): Controla el output de ALU_regs.
- **Reg0_Write** (control): Determina si se debe almacenar el dato en registro00.
- **Reg0_enableOut** (control): Controla la salida o no por el Reg4_output de registro00.
- **Reg1_Write** (control): Determina si se debe almacenar el dato en registro01.
- **Reg1_enableOut** (control): Controla la salida o no por el Reg4_output de registro01.
- **Reg2_Write** (control) :Determina si se debe almacenar el dato en registro02.
- **Reg2_enableOut** (control): Controla la salida o no por el Reg4_output de registro02.
- **Reg3_Write** (control): Determina si se debe almacenar el dato en registro03.
- **Reg3_enableOut** (control): Controla la salida o no por el Reg4_output de registro03.
- **Force_Input**: El dato de entrada de 4 bits.
- **en_Force_Input** (control): Controla el paso del Force_Input.
- **Z** : El flag que dice si es 0000.
- **N** : El flag que dice si es negativo.
- **V** : El flag que dice si hubo overflow.
- **C** : El flag que dice si hubo carry.
- **A_DEBUG**: Muestra el valor almacenado en la ALU_regs de A.
- **B_DEBUG**: Muestra el valor almacenado en la ALU_regs de B.
- **S_DEBUG**: Muestra el resultado de la operación que está almacenado en ALU_regs.
- **Reg4_Debug**: Es el almacenamiento de cada registro.
- **clk**: Señal de reloj
- **Reg4_output:** Muestra el resultado de la operación realizada en la ALU_regs

b) Detallar el contenido de cada display

Display (REG4_DEBUG): El dato almacenado en el registro correspondiente.
Display (A_DEBUG): Muestra el dato almacenado de la variable A en la ALU_regs.
Display (B_DEBUG): Muestra el dato almacenado de la variable B en la ALU_regs.
Display (S_DEBUG): Muestra el resultado almacenado de la operación hecha en la ALU_regs.
Display (REG4_OUTPUT / FORCE_INPUT): Muestra el Output de la operación realizada en la ALU o bien muestra el dato de 4 bits de FORCE_INPUT

c) Secuencia de señales

Activación:
1 - Colocar el dato 0100 (valor 4) en **Force_input**.
2 - Habilitar **en_Force_input** en 1 para permitir el paso del valor hacia los registros.
3 - Establecer **reg2_write** en 1 para permitir la escritura en reg2.
4 - Activar el **clk** (pasarlo de 0 a 1). Este flanco de subida, junto con las señales anteriores activas, hace que el valor de Force_input se almacene en reg2.
Desactivación:
1 - Bajar **clk** a 0.
2 - Bajar a 0 las señales: **en_Force_input**, **reg2_write**.
Activación:
1 - Colocar el dato 1101 (valor -3 en complemento a dos) en **Force_input**.
2 - Habilitar **en_Force_input** en 1 para permitir el paso del valor hacia los registros.
3 - Establecer **reg3_write** en 1 para permitir la escritura en reg3.
4 - Activar el **clk** (pasarlo de 0 a 1). Este flanco de subida, junto con las señales anteriores activas, hace que el valor de Force_input se almacene en reg3.
Desactivación:
1 - Bajar **clk** a 0.
2 - Bajar a 0 las señales: **en_Force_input**, **reg3_write**.

d) Completar la siguiente tabla:

Complemento a 2									Sin Signo								
Valor inicial	Res operación (OR)				Res operación (SUB)				Valor inicial	Res operación (OR)				Res operación (SUB)			
(4,0)	0100				0100				(4,0)	0100				0100			
	Flags				Flags					Flags				Flags			
	N	Z	V	C	N	Z	V	C		N	Z	V	C	N	Z	V	C
	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0
Valor inicial	Res operación (SUB)				Res operation (AND)				Valor inicial	Res operación (SUB)				Res operation (AND)			
(7,-1)	1000				0111				(7,-1)	X				X			
	Flags				Flags					Flags				Flags			
	N	Z	V	C	N	Z	V	C		N	Z	V	C	N	Z	V	C
	1	0	1	1	0	0	0	0		X	X	X	X	X	X	X	X
Valor inicial	Res operación (ADD)				Res operación (SUB)				Valor inicial	Res operación (ADD)				Res operación (SUB)			
(-8,-2)	0110				1010				(-8,-2)	X				X			
	Flags				Flags					Flags				Flags			
	N	Z	V	C	N	Z	V	C		N	Z	V	C	N	Z	V	C
	0	0	1	1	0	1	0	1		X	X	X	X	X	X	X	X
Valor inicial	Res operación (OR)				Res operation (AND)				Valor inicial	Res operación (OR)				Res operación(AND)			
(8,-9)	X				X				(8,-9)	X				X			
	Flags				Flags					Flags				Flags			
	N	Z	V	C	N	Z	V	C		N	Z	V	C	N	Z	V	C
	X	X	X	X	X	X	X	X		X	X	X	X	X	X	X	X

X = No se puede interpretar el número en la codificación indicada.

d)

Activación:
Colocar el dato 0100 (valor 4) en **Force_input**.
Habilitar **en_Force_input** en 1.
Establecer **reg0_write** en 1 para permitir la escritura en reg0.
Activar el **clk** (pasarlo de 0 a 1). Este flanco de subida, junto con las señales anteriores activas, hace que el valor de Force_input se almacene en reg0.
Desactivación:
Bajar **clk** a 0.
Bajar a 0 las señales: **en_Force_input**, **reg0_write**.

Nota: reg1 ya contiene por defecto el valor 0, no se realiza ninguna operación de carga para este registro.

Activación:
Establecer **reg0_enOut** en 1 para permitir la salida del valor de reg0.
Establecer **Alu_A_Write** en 1 para cargar el valor en el registro A de la ALU.
Activar el **clk** (pasarlo de 0 a 1). Este flanco de subida, junto con las señales anteriores

activas, hace que el valor de reg0 se almacene en A.
Desactivación:
Bajar **clk** a 0.
Bajar a 0 las señales: **reg0_enOut**, **Alu_A_Write**.
Activación:
Establecer **OP** = 11 para seleccionar la operación OR en la ALU.
Activar nuevamente el **clk** (pasarlo de 0 a 1) para ejecutar la operación OR entre A (4) y B (0).
Desactivación:
Bajar **clk** a 0 para que la operación se realice con éxito.
Activación:
Establecer **Alu_enOut** en 1 para permitir la salida del resultado de la ALU.
Establecer **reg2_write** en 1 para permitir la escritura en reg2.
Activar el **clk** (pasarlo de 0 a 1). Este flanco de subida hace que el resultado (4) se almacene en reg2.
Desactivación:
Bajar **clk** a 0.
Bajar a 0 las señales: **reg2_write**, **Alu_enOut**.
Activación:
Cambiar **OP** = 01 para seleccionar la operación SUB en la ALU.
Activar nuevamente el **clk** (pasarlo de 0 a 1) para que se realice la operación SUB entré A (4) y B (0).
Desactivación:
Bajar **clk** a 0 para que la operación se realice con éxito.
Activación:
Establecer **Alu_enOut** en 1 para permitir la salida del resultado.
Establecer **reg3_write** en 1 para permitir la escritura en reg3.
Activar el **clk** (pasarlo de 0 a 1). Este flanco de subida hace que el resultado (4) se almacene en reg3.
Desactivación:
Bajar **clk** a 0.
Bajar a 0 las señales: **reg3_write**, **Alu_enOut**.

e) Explicar:

La señal de **clk** que ingresa en el **registro _4 bytes salida ->** (Registro que está conectado al output del ALU) se niega así este tiene un flanco de activación opuesto a la secuencia de cargar y operar los datos A y B en la ALU. Permitiendo así que no se pasen datos erróneos al **registro_4bytes_salida** que almacena el resultado de la operación realizada en la ALU.

Corrección

Integrantes:

Nombre y Apellido: Francisco Alfredo Asplindh LU: 1231/23
Nombre y Apellido: Manuel Moujan Vilas LU: 299/24
Nombre y Apellido: Agustin Alejandro Schniepp LU: 744/24

Para uso de los docentes:

1	2	3