Taller de Lógica Digital

Organización del Computador 1

Verano 2018

Introducción - Simulador `Logisim`

El simulador se puede bajar desde la página http://www.cburch.com/logisim/ o de los repositorios de Ubuntu. Requiere Java 1.5 o superior. Para ejecutarlo, teclear en una consola `java -jar logisim.jar`.

El simulador opera en dos modos. Edición y Simulación:

- En el modo edición podremos definir el funcionamiento del circuito con todas las entradas y salidas, las compuertas lógicas o los componentes que lo componen, más el aspecto físico que tendrá el componente a la hora de ser utilizado por otros circuitos.
- El modo simulación nos permitirá testear el funcionamiento del componente, asignando valores a las entradas y testeando el valor de las salidas.

Ejercicios - Parte 1: Combinatorios

Nota: Sólo podrán utilizarse compuertas lógicas básicas AND, OR, XOR y NOT, *splitters* y multiplexores.

(1) **Contador de unos de 4 *bits***. Circuito con una entrada **A** de cuatro *bits* y una salida **S** de tres *bits*. La salida, interpretada como un número entero sin signo, indicará la cantidad de unos de la representación binaria de la entrada. Ejemplo: `E=1011` → `S=011`

(2) **Incrementador de 4 *bits***. Circuito de una entrada **A** de cuatro *bits* y una salida: **S**. La salida deberá expresar el resultado de incrementar la entrada en una unidad.

(3) **Sumador de 4 *bits* con Flags ZCVN**. Circuito de dos entradas de cuatro *bits* **A** y **B**, y una entrada **Cin** de un *bit*. Contiene dos salidas **S** de 4 *bits* y una salida **Cout** de un *bit*. El mismo calculará la suma entre los números representados por **A**, **B** y **Cin**, y reflejará el resultado en **S** con el acarreo final en **Cout**. Además el circuito contendrá cuatro salidas adicionales **Z**, **C**, **V** y **N** de un *bit* que representarán:

- **Z** vale 1 ⇔ el resultado es cero
- **C** vale 1 ⇔ la suma binaria produjo acarreo
- **V** vale 1 ⇔ la suma interpretada en complemento a 2 dio *overflow*
- **N** vale 1 ⇔ el resultado interpretado en complemento a 2 es negativo

(4) **ALU de 4 *bits***. Circuito con tres entradas **A**, **B** y **OP**, ésta última de dos *bits*, y cinco salidas: **S**, **Z**, **C**, **V** y **N**. La salida deberá expresar el resultado de:

- **OP** vale `00` ⇔ A + B
- **OP** vale `01` ⇔ A - B
- **OP** vale `10` ⇔ A AND B (bit a bit)
- **OP** vale `11` ⇔ A OR B (bit a bit)

Aquí **C** deberá informar si la resta binaria produjo *borrow* (dame uno). En las últimas dos operaciones, no importa el valor que tomen los flags **C** y **V**. En todos los casos los *flags* reflejan el resultado de operación entre operandos en complemento a 2.

Los *flags* deben cumplir:

- Reflejar el resultado de operación entre operandos en complemento a 2.
- En el caso de resta **C** deberá informar si se produjo *borrow* (dame uno).
- Para las operaciones lógicas, **C** y **V** valdrán cero.

a) ¿Es posible utilizar esta ALU con operandos interpretados como sin signo? ¿Cómo detectaría el *overflow*?

Ejercicios - Parte 2: Secuenciales

Nota: Sólo podrán utilizarse compuertas lógicas básicas AND, OR, XOR y NOT, el buffer de 3 estados (*Controlled Buffer*) de un *bit* y el flip-flop D de un *bit*.

(5) **Componente medio compartido**. El componente llamado `medio compartido` tiene 2 entradas de datos (`A` y `B`), 2 de control `A_en`, `B_en` y una salida `R`.

a) Escribir la tabla de verdad del circuito, e identificar que combinaciones resultan válidas y cuales no.

b) Explicar qué significan los distintos colores que toman los cables.

c) Por motivos didácticos en la materia se interpretan como inválidas combinaciones que en el simulador no lo son. Indicar cuáles son y por qué.

(6) **Movimientos entre registros**. El componente llamado `EJ5` presenta la composición de varios componentes. Tiene 3 instancias del componente `registro-salida-restringida`, que se encuentra incompleto. Este componente almacena un valor interno de un *bit*.

a) Completar el componente `registro-salida-restringida` respetando el siguiente comportamiento por cada cambio en la señal `clk` (flanco acendente).

- Si `W=1`, almacena `in` como estado interno.
- Si `en_out=1`, `out` refleja el estado interno. Caso contrario debe estar en alta impedancia (**Hi-Z**).
- Por motivos didácticos se incluye la salida `out_debug` para observar el estado interno almacenado de forma asincrónica.

b) Analizar el componente `EJ5` y responder:

- ¿Cuáles son y qué respresenta cada entrada y cada salida del componente? ¿Cuáles entradas deben ser consideradas como de control?
- Las entradas `Force_input` y `en_force_input` sirven para poder introducir en el circuito un valor arbitrario. Escribir una secuencia de activación y desactivación de entradas para que el registro `R1` pase a tener el valor `1`.
- Dar una secuencia de activaciones que intercambie los valores almacenados entre `R0` y `R2`, utilizando temporariamente `R1`.

(7) **Máquina de 4 *bits***. El componente `EJ3` representa una máquina con 4 registros de *propósito general* y una ALU (ejercio prévio). Está realiza operaciones en 4 *bits* y complemento a 2. Los registros usados son instancias de `registro-4b-salida-restringida`. El mismo es una extensión del circuito `registro-salida-restringida` para 4 *bits*. Analizar el componente `EJ3` y responder:

- ¿Cuáles son y qué respresenta cada entrada y cada salida del componente? ¿Cuáles entradas deben ser consideradas como de control?
- ¿Qué información muestra cada display y cómo es interpretada esta información?
- Escribir la secuencia de activación y desactivación de señales para cargar el valor `4` en el registro `R2` y el `-3` en el registro `R3`
- Cargar los siguientes pares de valores y realizar las operaciones indicadas:

R0	R1	
4	0	OR y SUB
7	-1	SUB y AND
-8	-2	ADD y SUB
8	-9	OR y AND

  Almacenar los resultados en `R2` y `R3` para cada operación respectivamente. Escribir la secuencia completa de activación de señales para el primer caso.
- ¿Por qué se niega la señal *clk* en la ALU antes de pasarle este valor al registro de salida? ¿Qué sucede de no realizar la negación?

Detalles adicionales sobre el simulador

- Las entradas (*inputs*) se simbolizan con un cuadrado y tienen el comportamiento de una llave (Prendido = 1; Apagado = 0). Se pueden definir de más de un *bit* y modificar su estado de forma manual.
- Las salidas (*outputs*) se simbolizan con un círculo y serán "prendidas" cuando por su entrada haya un 1, o "apagadas" cuando por su entrada haya un 0. Se pueden definir de más de un *bit*.
- En caso de ser necesario que el circuito tenga una entrada fija en algún valor, podremos utilizar el componente "Wiring/Constant".
- Se recomienda utilizar Etiquetas para facilitar la comprensión de cada componente.
- El simulador permite usar cables de más de un bit por medio del componente "Wiring/Splitter".