Organización del Computador
Práctico de Taller 1: Descripción de Hardware en Lava

B. Gonzalez Kriegel - N. Wolovick

10 de abril de 2001

1 Problemas

1.1 Ordenador de 4 bits

Un ordenador de bits es un circuito que dada una entrada de n bits, genera una salida también de n bits con los mismos valores de entrada, pero ordenados de manera creciente. Como sólo tenemos dos elementos en el dominio, el resultado será simplemente poner todos los 0's a la izquierda y los 1's a la derecha.

```
Main> simulate bitSorter [low,high,high,low,low]
[low,low,low,high,high]
```

- Definir un ordenador de 2 bits `bitSorter2` que toma como entrada un par de bits y devuelve el mismo conjunto de bits pero con el menor en el lado derecho y el mayor en el izquierdo. Simule y compruebe su correcto funcionamiento en todo su dominio.
- Verifique que es conmutativo y que la salida es ordenada (el bit izquierdo es menor que el derecho).
- Verifique que la salida es permutación de la entrada (contiene los mismos bits).
- * A partir de este bloque constructivo defina una red que ordena 4 bits `bitSorter4`. Simule el diseño en algunos puntos del dominio y compruebe que se comporta como un ordenador.
- * Para este caso también verifique que la salida cumple la propiedad de estar ordenado.
- ** Verifique que la salida es permutación de la entrada.
- *** Obtener un diseño general que ordene n bits. Simule para casos de prueba y verifique las propiedades relevantes para $n = \{8, 16, 32, 64, 128\}$.

1.2 Contador 0 a 9 en Display 7 Segmentos

- Describa un decodificador binario a 7 segmentos `digitalDisplay (a0,a1,a2,a3) = (a,b,c,d,e,f,g)...` y simule en todo su dominio para comprobar su funcionamiento.
- Describa un contador binario ascendente de 4 bits con reset `upCounter4 reset = (q0,q1,q2,q3)....` Simule para `reset=0`, `reset=1` y para cambios de `reset`.
- Interconecte los dos circuitos de forma tal que la cuenta sea $(0 \ldots 9)^*$. Muestre mediante simulación que el circuito hace lo que se busca.

1.3 Restador de n bits

- Describa un semirestador `halfSub (borrIn,a) = (diff,borrOut)...` y un restador completo `fullSub (borrIn,(a,b)) = (diff,borrOut)...`. Mediante simulación compruebe el correcto funcionamiento de ambos en todo su dominio.
- Verifique conmutatividad para ambos circuitos.
- Mediante patrones de conexión defina `bitSubber` que resta un bit a un número binario. Simúlelo para casos puntuales.
- Verifique que se cumple $a - 0 = a$.
- * ¿Se cumple la propiedad $(a-1)+1 = a$ y $(a+1)-1 = a$? ¿Por qué? Probablemente necesitará utilizar `bitAdder` del módulo `Arithmetic`.
- ** ¿Como verificaría esta idea en un dominio restringido?
- Mediante patrones de conexión defina un restador genérico de n bits `subber`. Simúlelo para casos puntuales.
- Verifique para $n = \{8, 16, 32, 64, 128\}$ que el circuito es conmutativo y que cumple con $a - a = 0$.
- ** Verifique para ciertos n que en un dominio restringido $(a+b) - b = a$.

1.4 Carry Lookahead Adder (CLA) Puro

En un CLA puro, a partir de las señales p_i y g_i, (propagación y generación del bit i-ésimo) se calculan todos los c_i en paralelo y en profundidad constante, pero fan-in creciente en sus compuertas.

- Describa un CLA puro de 4 bits CLA_4
  `claAdder4 (c0,((a0,a1,a2,a3),(b0,b1,b2,b3))) = ((s0,s1,s2,s3),c4)...` y simule en casos puntuales para testearlo.
- Verifique la equivalencia con `adder` particularizado a 4 bits `adder4 (c0,((a0,b0),(a1,b1),(a2,b2),(a3,b3))) = ((s0,s1,s2,s3),c4)`. ¿Tiene sentido verificar más propiedades? ¿Por qué?
- *** Describa un CLA_n utilizando definiciones de circuitos con ecuaciones recursivas. Simule en algunos puntos del dominio y verifique equivalencia con `adder`.

1.5 Carry Select Adder ($CSeA$) de Bloque $\frac{n}{2}$

Un $CSeA$ de bloque $\frac{n}{2}$, divide las palabras a sumar en dos bloques de $\frac{n}{2}$ bits. El bloque de menor peso se suma de manera tradicional, mientras que el de mayor peso se suma dos veces, una con $c_{\frac{n}{2}} = 0$ y la otra con $c_{\frac{n}{2}} = 1$. Un $\frac{n}{2} \times MUX2-a-1$ selecciona el primer o segundo bloque según el valor real de $c_{\frac{n}{2}}$ que sale del primer bloque.

- Describa un $CSeA$ de bloque $\frac{n}{2}$ de 4 bits `cseAdder4` utilizando un `adder2 (c0,((a0,a1),(b0,b1))) = ((s0,s1),c2)...`, que será obtienido particularizando el RCA genérico `adder`. Simule en algunos puntos del dominio a manera test inicial.
- Verifique la equivalencia con `adder` particularizado a 4 bits `adder4`. ¿Tiene sentido verificar más propiedades? ¿Por qué?
- ** Describa un $CSeA_n$ utilizando patrones de circuitos. Simule en algunos puntos del dominio y verifique equivalencia con `adder`.

1.6 Carry Skip Adder (*CSkA*) de Bloque $\frac{n}{2}$

El *CSkA* de bloque $\frac{n}{2}$, al igual que el *CSeA*, divide en dos la suma. Cada bloque es un sumador de $\frac{n}{2}$ bits de algún tipo (podría ser un híbrido) conectados en cascada a la manera del *RCA*. La diferencia radica en que c_n se calcula distinto para que el acarreo pueda “saltar” el segundo bloque.

$$c_n = \tilde{c}_n + P_{\frac{n}{2},n-1} \cdot c_{\frac{n}{2}}$$

donde $P_{\frac{n}{2},n-1} = \prod_{i=\frac{n}{2}}^{n-1} p_i$ es el bit de propagación del grupo $[\frac{n}{2}, n-1]$ y $\tilde{c}_n$ es el acarreo de salida del segundo bloque. Cuando $P_{\frac{n}{2},n-1} \cdot c_{\frac{n}{2}} = 1$ el acarreo de salida se está calculando más rápido que $\tilde{c}_n$.

- Describa un *CSkA* de bloque $\frac{n}{2}$ de 4 bits utilizando un `adder2 (c0,((a0,a1),(b0,b1))) = ((s0,s1),c2)...`, el que será obtienido particularizando el *RCA* genérico `adder`. Simule en algunos puntos del dominio a manera test inicial.
- Verifique la equivalencia con `adder` particularizado a 4 `adder4`. ¿Tiene sentido verificar más propiedades? ¿Por qué?
- ** Describa un $CSkA_n$ utilizando patrones de circuitos. Simule en algunos puntos del dominio y verifique equivalencia con `adder`.

1.7 Ordenador de Dos Palabras

Este ordenador es un circuito que toma dos palabras de n bits y las devuelve en el orden correcto, es decir primero la menor.

- Describa un intercambiador de bits controlado `swapper1 (swap,(a,b)) = (c,d)...` donde la salida `(c,d)` = `(a,b)` si `swap` es bajo, sino `(c,d) = (b,a)`. Simule en todo el dominio y compruebe que cumple con lo esperado. Verifique que su salida siempre es una permutación de la entrada.
- Describa un intercambiador de cuatro bits `swapper4 (swap,((a0,a1,a2,a3),(b0,b1,b2,b3)))...` a partir del anterior. Verifique que su salida siempre es una permutación de la entrada.
- * Describa el circuito `gt2` que toma un par de bits `(a,b)` y devuelve alto si $a > b$. A partir de `gt2` y `equiv` describa un circuito `ge4 ((a0,a1,a2,a3),(b0,b1,b2,b3)) = c...` donde c es alto cuando $(a_3a_2a_1a_0)_2 \geq (b_3b_2b_1b_0)_2$. Verifique que efectivamente `ge4` es el mayor igual (*greater equal*) contrastando con un circuito que toma un par de números binarios de 4 bits, los transforma en enteros y aplica la compuerta de dominio entero y rango booleano `gte` definido en el módulo `Arithmetic`.
- Utilizando `swapper4`, `ge4` y patrones de conexión, describa el ordenador de dos palabras de cuatro bits `sorter2_4`. Verificar su correcto funcionamiento validando que la salida es una permutación de la entrada y que la primera salida es menor que la segunda.
- *** Generalice `swapper4` a `swapper`, `ge4` a `ge` y `sorter2_4` a `sorter2` utilizando patrones de conexión y definiciones de circuitos mediante equaciones recursivas.

1.8 * Un Sumador General

El sumador general de n bits que se definió en el apunte requería de una condición que no fue hecha explícita: que las dos palabras fueran de igual largo. Para sumar palabras de diferente largo podemos dar dos soluciones: usar *padding* con señales `low` extras, o bien utilizar semisumadores para propagar el acarreo final al resto del número más largo.

- Implemente estos dos métodos.

- Verifique que estas dos implementaciones son equivalentes para sumar números de tamaños distintos. Compruebe con todas las combinaciones de números de 8, 16 y 32 bits.

1.9 Números *Carry Save*

Para representar números naturales, comunmente utilizamos representación binaria. La representación *carry save* utiliza pares de bits en vez de bits simples para cada 2^i.

La función de representación es la siguiente

$$((a_{n-1}, b_{n-1}), \ldots, (a_0, b_0))_{CS} = \sum_{i=0}^{n-1} (a_i + b_i) 2^i$$

y por ejemplo

$$((0,1)(1,1)(1,0))_{CS} = (0+1) \times 2^2 + (1+1) \times 2^1 + (1+0) \times 2^0$$

Notar que existen diferentes representaciones para un mismo número.

- Defina un conversor *carry save* a binario `carry2bin`. El circuito deberá tomar un número *carry save* de cualquier tamaño y devolver el mismo número pero en representación binaria (notar que este ocupa $n+1$ bits). Simúlelo en casos puntuales para testear su funcionamiento.
- Piense en un conversor al revés. ¿Qué sería problemático si queremos utilizar el conversor para la verificación?

2 Entrega

Se deberán **entregar 2 de los 9 ejercicios** propuestos por grupo. Cada ejercicio deberá estar claramente identificado y contener lo siguiente:

- Un archivo Lava con el *código* de los circuitos definidos en cada ejercicio, junto a todas las funciones auxiliares, de tal forma que sea un archivo autocontenido. No olvide marcar para cada definición que ejercicio o parte de ejercicio está resolviendo.
- Un archivo *ASCII*, *TEX*, *DVI* o *PS* con la *descripción* de las respuestas a los ejercicios, es decir las motivaciones de las decisiones que hayan tomado. Es recomendable mostrar lo obtenido en la simulación y/o verificación. Tampoco olviden incluir en el inicio del informe el número de grupo y sus integrantes con apellido, nombre y L.E. .

Se recomienda, a fin de facilitar el trabajo de corrección, mezclar código y descripción en un sólo archivo `.hs` de tal forma que sirva para ejecutarlo y entenderlo. Recuerde que los comentarios en Lava se pueden hacer de una o varias líneas.

```
-- Comentario de una linea

{- Un comentario de varias lineas
puede ser escrito utilizando
esta forma -}
```

El o los archivos deberán ser enviados a la dirección de correo de la materia `odc@hal.famaf.unc.edu.ar` conteniendo como subject "`Taller 1 Grupo` *n*". La fecha límite de entrega es la hora y día de comienzo del primer parcial.

No existen límites respecto a la cantidad de veces que el trabajo puede ser entregado hasta que haya sido aprobado. Cuando el trabajo está aprobado, se generará una calificación que comprenderá el **30% de la nota del primer parcial**.