DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN
ESCUELA DE INGENIERÍA
PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE

IIC2343 - Arquitectura de Computadores (I/2020)

Ayudantía 1

Representación de enteros y circuitos lógicos combinacionales

1. Representación de enteros

Convierte de decimal a base 2 con complemento a 2 y 8 *bits* los siguientes números:

1. 10.
2. -5.
3. -150.

2. Circuitos lógicos combinacionales

Para cada uno de los siguientes componentes, crea la tabla de verdad, fórmula lógica y circuito lógico correspondiente.

1. **Enabler:** recibe un dato y una señal de control y solo deja pasar el dato si la señal está en 1.
2. **Multiplexor/Mux:** recibe dos datos (digamos A y B) y una señal de control. Si la señal de control es 0, sale A. De lo contrario, sale B.
3. **Demultiplexor/Demux:** recibe un dato, una señal de control y tiene dos outputs. Si la señal de control es 0, el dato sale por la primera salida. De lo contrario, sale por la segunda.
4. **Shifter lógico:** recibe un entero de 8 bits y una señal de control. Si la señal de control es 0, realiza la operación SHL sobre el número. De lo contrario, realiza la operación SHR sobre el número.

Solución

1.1) 10

Para hacer cambio de base hemos visto dos métodos. Uno es una suma de potencias y otro es iterando divisiones.

Si intentáramos hacer esta transformación con el primer método, pasaría lo siguiente.

$$X_2 = 1 \cdot (diez_2)^1 + 0 \cdot (diez_2)^0$$

Ese $diez_2$ tiene que ser la representación de 10 en base 2, pero eso es lo que queremos calcular, por lo que este método no nos sirve.

Generalmente, utilizaremos la suma de potencias cuando mi base de origen es menor que mi base de destino, e iterar divisiones cuando mi base de origen es mayor que mi base de destino.

Según este método, vamos a iterar divisiones, primero tomando nuestro número y dividiéndolo por la base a la que queremos llegar y luego, tomando el resultado de la última división que hicimos y convirtiéndolo en el dividendo de la nueva. Solo pararemos cuando el resultado pueda ser escrito mediante un solo símbolo de nuestra base de destino.

$$10 : 2 = 5 \mid \text{y el resto es } 0$$

$$5 : 2 = 2 \mid \text{y el resto es } 1$$

$$2 : 2 = 1 \mid \text{y el resto es } 0$$

Bien, llegamos a un resultado que sí se puede escribir mediante un único símbolo en base 2: 1. Ahora vamos a armar nuestro número.

Para armar el resultado final, lo que hacemos es tomar el resultado de la última división y esta será nuestra cifra más significativa. Luego, tomamos el resto de cada una de las divisiones que hicimos y los ponemos, siendo menos significativa la cifra mientras más antigua sea esa división.

De este modo, tenemos que 10 en base 2 es: 1010 (los colores indican de dónde sale cada cifra).

1.2) -5

Utilizando el método de las divisiones, porque no hay otra opción plausible, tenemos que:

$$5 : 2 = 2 \mid \text{y el resto es } 1$$

$$2 : 2 = 1 \mid \text{y el resto es } 0$$

Y 5 en base 2 es 0b00000101. Ahora necesitamos calcular su complemento a 2.

Al negar todos los bits obtenemos: 0b11111010.

Le sumamos 1 y obtenemos: 0b11111011.

¿Cómo podemos comprobar que ese número es -5? Si sumamos 5 + (-5) deberíamos obtener 0, que es el caso (0b00000101 + 0b11111011 = 0b100000000, el bit en rojo es el carry).

1.3) -150

$$150 : 2 = 75 \mid \text{y el resto es } 0$$

$$75 : 2 = 37 \mid \text{y el resto es } 1$$

$$37 : 2 = 18 \mid \text{y el resto es } 1$$

$$18 : 2 = 9 \mid \text{y el resto es } 0$$

$$9 : 2 = 4 \mid \text{y el resto es } 1$$

$$4 : 2 = 2 \mid \text{y el resto es } 0$$

$$2 : 2 = 1 \mid \text{y el resto es } 0$$

Por lo que 150 es 10010110. Notemos que ocupa los 8 bits y el más significativo es un 1, por lo que técnicamente nos está dando un número negativo. Si no tuviéramos complemento a 2 sí estaríamos ante un 150.

Si le aplicamos el complemento a 2 al 10010110 obtenemos que es 01101010. ¿Qué número es? Ejecuta el programa n150.c que está subido en la carpeta de la ayudantía 1 y compruébalo tú mismo.

2.1) Enabler

Un enabler es un componente que recibe un dato y una señal de control, según el valor que tenga la señal de control, deja pasar o no el dato.

En los computadores solo tenemos dos valores posibles: 0 y 1. Y, en realidad, no existe algo como "mandar una señal". Si no hay nada, es 0. Si no deja pasar el dato, sale 0. Si no tiene nada conectado a su input, recibe 0. Esto es para cualquier componente, no solo el enabler.

A	**S**	**Out**
0	0	**0**
0	1	**0**
1	0	**0**
1	1	**1**

Para crear las fórmulas lógicas, para cada output donde veamos un 1, escribiremos una fórmula con los inputs y las separaremos todas mediante conectores or. En este caso, la fórmula lógica que creamos a partir de la tabla es...

$$out \equiv A \wedge S$$

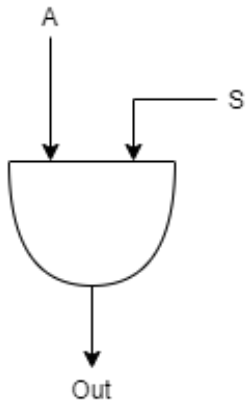

Figura 1: Enabler.

2.2) Multiplexor (Mux)

Un multiplexor o mux es un componente que recibe dos datos y, según el valor de una señal de control, seleccionará uno de los dos valores y lo pasará por la salida.

Esta es la tabla de verdad que armamos a partir de la descripción:

A	**B**	**S**	**Out**
0	0	0	**0**
0	0	1	**0**
0	1	0	**0**
0	1	1	**1**
1	0	0	**1**
1	0	1	**0**
1	1	0	**1**
1	1	1	**1**

Y al crear la fórmula lógica obtenemos:

$$out \equiv (\neg A \wedge B \wedge S) \vee (A \wedge \neg B \wedge \neg S) \vee (A \wedge B \wedge \neg S) \vee (A \wedge B \wedge S)$$

Pero si la trabajamos para simplificarla, utilizando el axioma de lógica proposicional que dice que $(A \wedge \neg B) \vee (A \wedge B) \equiv A$, obtenemos la siguiente fórmula lógica mucho más pequeña:

$$out \equiv (A \wedge \neg S) \vee (B \wedge S)$$

Finalmente, al transformarla en circuito obtendríamos:

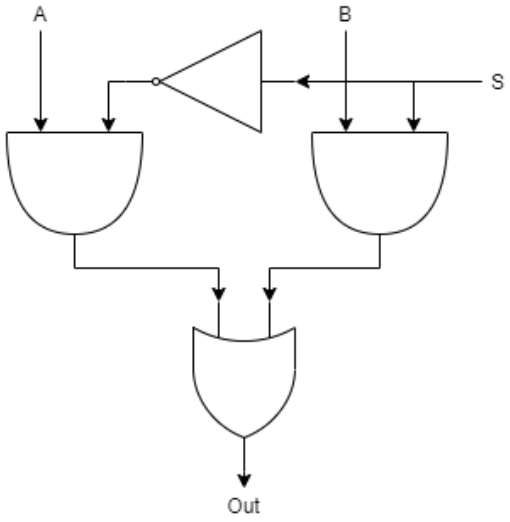

Figura 2: Mux

2.3) Demultiplexor (Demux)

Un demux es un componente que recibe un solo dato pero que tiene dos salidas y, según una señal de control, decide por cuál de ellas lo enviará.

A partir de esta descripción creamos la siguiente tabla de verdad:

A	**S**	**Out 1**	**Out 2**
0	0	**0**	**0**
0	1	**0**	**0**
1	0	**1**	**0**
1	1	**0**	**1**

Y la fórmula lógica para cada salida es:

$$out1 \equiv \neg(A \rightarrow S) \equiv A \wedge \neg S$$
$$out2 \equiv A \wedge S$$

Luego, al transformarlo en circuito obtenemos:

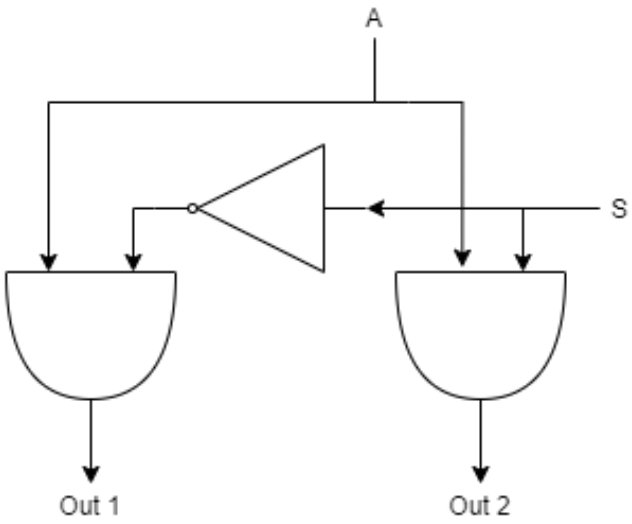

Figura 3: Demuxer

2.4) Shifter lógico

El *shifter lógico* es un componente que es capaz de realizar las operaciones de shifting lógicas, tanto hacia la izquierda como hacia la derecha, dependiendo de la señal de control que le demos. Lo armaremos en base a multiplexores, en lugar de manejar la tabla de verdad. Lo haré con 4 bits en honor al espacio y tiempo de diagramado. El input será A con bits a_0, a_1, a_2 y a_3, donde a_0 es el bit menos significativo.

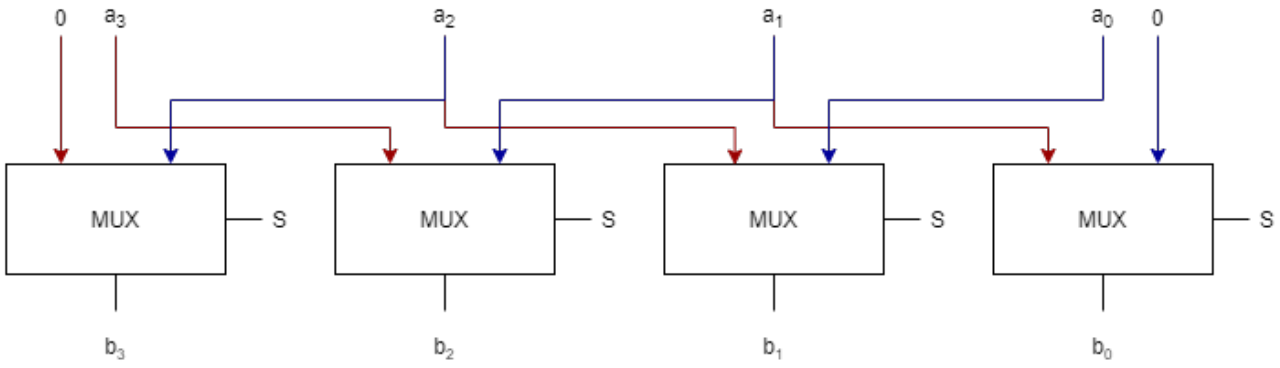

Figura 4: Logic shifter.

Si la señal S es 0, el output será 0 a_3 a_2 a_1, lo que se corresponde con un *shift right*.
Si la señal S es 1, el output será a_3 a_2 a_1 a_0 0, lo que se corresponde con un *shift left*.