ALGORITMO PARA LA OBTENCIÓN DE NÚMEROS EN FORMATO FLOTANTE MEDIANTE TECLADO MATRICIAL

AUTOR: GARCÍA LOMELÍ ABRAHAM AMOS
INSTITUTO POLITÉCNICO NACIONAL
ESCOM

1 Introducción

La idea primaria de loa dispositivos electrónicos ha sido la de proporcionar precisión al momento de pretender realizar una serie de actividades que podrían ser automatizada mediante el uso de diferentes algoritmos. Este concepto permite entender que la idea principal de desarrollar una implementación dentro de un prototipo de diseño (como puede ser la Match XO2 de Lattice) obedece siempre a la máxima de lograr la automatización de determinados procesos que impliquen precisión y que en la medida de las necesidades del problema a solucionar, pueda dar soporte a la interacción entre el procesador y el usuario.

Es a partir de este punto que se pretenderá esforzar el desarrollo del presente escrito en describir uno de esos algoritmos para la obtención de datos al usuario mediante la configuración de un dispositivo de entrada, para su posterior procesamiento y despliegue de los resultados deseados.

El presente escrito tendrá como finalidad el explicar la forma en la cual, mediante un dispositivo de entrada (en este caso, el teclado matricial) se pueden obtener números de precisión sencilla en el formato IEEE 754, tomando en cuenta que el resultado obtenido será útil para poder realizar operaciones aritméticas (como es el caso del producto, proceso que ha sido descrito en entregas anteriores).

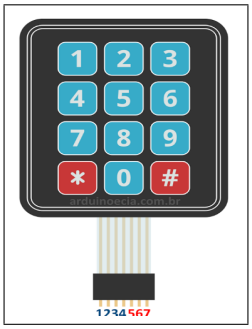
Es menester aclarar, que en cuanto a documentación que describa este proceso, de tal manera que se pueda respaldar de manera formal lo aquí descrito, no he tenido el atino de encontrar datos útiles, y con base en lo anterior, el algoritmo que se describirá a continuación aún no ha sido implementado, y siendo así no me es posible confirmar su correcto funcionamiento en su totalidad, no obstante confío en que la abstracción que he realizado con el fin de lograr representar números desde el teclado en el formato IEEE 754 satisface las necesidades propias de los objetivos del documuento.

2 Algoritmo de obtención de flotantes mediante teclado matricial

En primera instancia es necesario conocer la forma en la cual el hardware interactura, con la finalidad de entender *cómo es que los datos serán recibidos*, (en este caso por nuestro FPGA). Para ello se explicará de una manera muy breve la lógica de la configuración que se encuentra tras el teclado matricial.

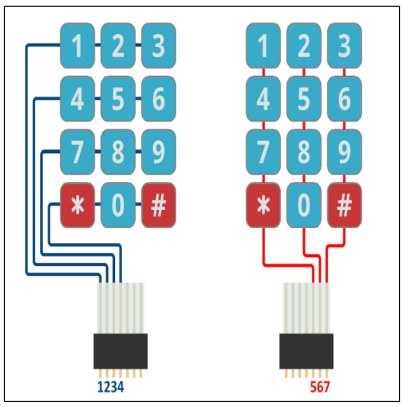
2.1 Configuración teórica del teclado matricial

Supóngase por ejemplo el siguiente teclado matricial:



Dado que se tienen 12 posibles entradas, se realiza un corriemiento en anillo el cual dara se encargará de ir hhabilitando cada fila por un perido discreto de tiempo, asegurándose que solo esa fila está habilitada. Por ejemplo, para este caso, como se tienen 4 filas la cadena a rotar será: *0001*; cada pulso de reloj se harán los siguientes cambios:

- **Pulso 1:** 0001
- **Pulso 2:** 0010
- **Pulso 3:** 0100
- **Pulso 4:** 1000
- **Pulso 5:** 0001



Esto implica que en el **pulso 2** solo estará habilitada la fila 2, es decir los números *[4,5,6]*. Para ello los siguientes 3 pines representarán las columnas de tal manera que si se tiene la siguiente cadena:

0010 010

Se dice que se habilita la fila 2 con la columna 2, es decir **el número 5**. A partir de esta lógica se pueden hacer implementaciones en VHDL o Verilog que son capaces de clasificar las diferentes entradas (por ejemplo mediante la sentencia *when case*). Para realizar el algoritmo que será explicado a continuación es necesario entender que tanto **# como** serán usados para propósitos específicos:

- **Caracter #**: será el equivalente al punto decimal
- **Caracter** : será usado para dar el comando **enviar**.

2.2 Obtención del número decimal mediante el teclado matricial

Supóngase que se desea ingresar al FPGA el número **14.32**, para ello se debe obtener del teclado matricial los siguientes elementos:

- 1000 100 (1)
- 0100 100 (4)
- 0001 001 (.)
- 1000 001 (3)
- 1000 010 (2)

Estos pulsos serán reconocidos por el FPGA dando los números que corresponden, posteriormente **se hará uso de la memoria RAM y de un contador** para almacenar cada uno de los dígitos que se han enviado, con cada envío realizado desde el teclado, se incrementará el contador que lleva el control de la memoria RAM, de tal manera que se tiene un número claro de los datos que se han ingresado.

Como el valor máximo a ingresar es de **9**, el cual tiene la representación binaria de 01001_2, querra decir que se usarán al menos 4 bits por cada número. Si se quisiera utilizar las propiedades de signo del sistema IEEE 754 se puede usar el bit mas significativo, siendo 1 la representación del signo negativo y 0 la representación del signo positivo.

También es importante mensionar que el valor del punto decimal estará dado por el número 11111 el cual **no debe ser interpretado como 31**, sinó como el punto decimal. Al final, ya que se han insertado todos los dígitos que formarán el número a representar, se usará la tecla **ENVIAR** para dar la señal a la RAM para que reinicie el contador, pero con fines de lectura. A partir de este momento se puede iniciar el algoritmo de representación para IEEE 754.

2.3 Conversión de datos a IEEE 754

Dado el ejemplo anterior, en la ram se tendrían guardados los siguientes valores:

Representación de las 5 celdas de la RAM usadas

00001	00100	11111	00011	00010
1	4	.	3	2

A continuación se realizará la conversión a IEEE 754 mediante el uso de los siguientes pasos:

1. Se cuentan *cuántos bloques de 5 bits están antes del signo* (este dato será conocido como s; en este caso hay 2 bloques. Cada bloque será llamado b.

$$S = 2 \tag{1}$$

2. Se usará un nuevo vector llamado *Enteros*

3. El valor de enteros será:

$$Enteros = \sum_{i=1}^{S} b_i * 10^{i-1} \tag{2}$$

   Por ejemplo, en este caso se tendrá que realizar:

$$Enteros = 00001_2 * 10^1 + 00100_2 * 10^0 \tag{3}$$

   Esto equivale a:

$$Enteros = 1_{10} * 10^1 + 4_{10} * 10^0 = 14_{10} = 01110_2 \tag{4}$$

4. Se realiza un proceso similar al descrito anteriomente para los números que forman la parte decimal, el resultado se guardará en el vector *Decimal*

$$Decimal = 00011_2 * 10^1 + 00010_2 * 10^0 = 100000_2 = 32_{10} \tag{5}$$

5. A continuación se declara el vector $Numero$ el cual sera igual a la concatenacion (interpretar en (6) el signo '+' como concatenación):

$$Numero = Entero + 11111 + Decimal \tag{6}$$

   Siendo así se tendrá la cadena:

Representación de la cadena Número

01110	11111	10000000
Entero	.	Decimal

6. Dentro del campo *Entero* se procede a **contar el número de bits desde el menos significativo hasta el último '1'**, es decir que para el campo *Entero* que es 01110 se tiene que de derecha a izquierda hay $x = 110$ antes del 1 mas significativo, es decir que hay **3 bits**, este valor sera conocido como m

7. Se concatenará a Decimal el valor de x al principio, por lo tanto se tendrá:

Representación de la cadena Número

01	11111	11010000000
Entero	.	Decimal

8. Como se movió m veces, se dirá que el valor del exponente será $m+127$, en este caso el expontente será:

$$Exponente = 3 + 127 = 130_{10} = 10000010_2 \tag{7}$$

9. La matiza será el resultado del producto de:

$$Matiza = Decimal * 2^m = 11010000000_2 * 2^3 = 11001010001111010111000 \tag{8}$$

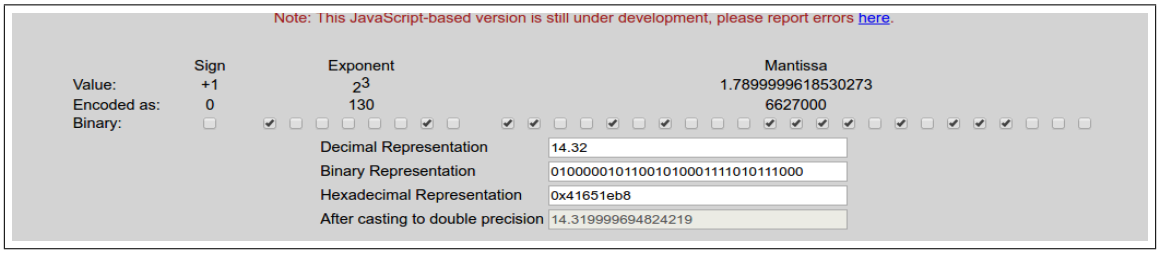
10. Por último se observa que en caso de que el valor de la matiza sea menor a 23 bits, se rellenarán los resultantes con 0, no obstante para este ejemplo no es necesario.

11. Para dar el resultado se concatenará:

Representación de la cadena Final

0	10000010	11001010001111010111000
Bit de signo	Exponente	Matiza

Mediante la ayuda de diferentes aplicaciones web, es posible corroborar nuestro resultado:



3 Bibliografía

References

[1] M. Morris Mano, *Arquitectura de computadoras*, Pearson Educación, 1994.

[2] M. Morris Mano, *Lógica digital y diseño de computadores*, Pearson Educación, 1982.

[3] David A. Patterson, John L. Hennessy, *Estructura y diseño de computadores*, Reverte, 2000.

[4] Francisco A. Martínez Gil, Gregorio Martín Quetglás, *Introducción a la programación estructurada en C*, Universitat de València, 2003.