3C TIC *(Edición núm. 13)* Vol.4 – Nº 2
Junio - septiembre 2015, 113 - 119
Área de Innovación y Desarrollo, S.L.
ISSN: 2254 – 6529

Recepción: 02 de diciembre de 2014
Aceptación: 01 de junio de 2015
Publicación: 25 de junio de 2015

ALGORITMO DE BOOTH EN OPERACIONES DE ADICIÓN Y SUSTRACCIÓN

BOOTH ALGORITHM OPERATIONS ADDITION AND SUBTRACTION

Jesús Ayuso Pérez[1]

1. Compositor musical y desarrollador software. Licenciado en Ingeniería Informática por la Universidad Carlos III de Madrid (UC3M). E-mail: ayusoperez@terra.com

3C TIC *(Edición núm. 13)* Vol.4 – Nº 2
Junio – septiembre 2015, 113 - 119
Área de Innovación y Desarrollo, S.L.
ISSN: 2254 – 6529

RESUMEN

El algoritmo dado por Andrew Donald Booth en 1950 para la multiplicación, no es únicamente aplicable a dicha operación, se puede aplicar a cualquier operación algebraica que se construya sobre una operación invertible. De ahí que en el presente documento, propongamos un algoritmo de adición y sustracción basado en dicho concepto. Veremos la aportación a nivel físico de utilizar esa técnica para implementar la suma o resta de 2 números enteros, en lugar de una solución más tradicional como se construiría con celdas FULL-ADDER.

ABSTRACT

The algorithm given by Andrew Donald Booth in 1950 for multiplying is not only applicable to this operation. It can be applied to any algebraic operation that builds on an invertible operation. Hence, in this paper, we propose an algorithm for addition and subtraction based on this concept. We see the contribution to the physical level of the technique used to implement the addition or subtraction of two integers, instead of a more traditional solution as built with FULL-ADDER cells.

PALABRAS CLAVE

Algoritmo Booth; adición; sustracción; sucesor; antecesor

KEYWORDS

Booth algorithm; addition; theft; successor; predecessor

3C TIC (Edición núm. 13) Vol.4 – Nº 2
Junio – septiembre 2015, 113 - 119
Área de Innovación y Desarrollo, S.L.
ISSN: 2254 – 6529

INTRODUCCIÓN

Como adelantábamos, el concepto de Booth es aplicable a toda operación algebraica que sea definible como la sucesión de cómputos de otra operación, generalmente más simple. Si además dicha operación más simple es invertible, podemos utilizar la tabla de acciones dada por Booth para reducir el número de operaciones a realizar para obtener el cómputo final. En el caso del algoritmo original de multiplicación, enfocando ésta como una sucesión de sumas, y en el de la adición, lo haremos como una sucesión de incrementos. Además, es muy normal que se aplique en el mundo de la Criptografía a la operación de exponenciación, mirándola como una sucesión de multiplicaciones, y para este caso concreto, al ser un cómputo muy pesado, en lugar de la representación de operandos de Booth, se suele utilizar la representación NAF, la cual reduce aún más el número de operaciones que el algoritmo de Booth.

Partiendo de esto, tenemos que nuestra operación de suma o nuestra operación de resta las plantearemos como una sucesión de incrementos o decrementos, respectivamente. Esto nos permitirá poder prescindir de parte de nuestro hardware, sustituyendo las celdas FULL-ADDER por celdas HALF-ADDER. Destacar que no sólo ahorramos hardware con dicha sustitución, sino que una celda HALF-ADDER es mucho más simple, constando únicamente de 2 puertas lógicas (con cómputos paralelos), luego el retardo introducido por la celda va a ser mucho menor, algo que será de gran importancia para nuestros carry-out (además de los consiguientes parámetros de ahorro energético y problemas de disipación de calor). Para el caso de la resta, estaríamos exactamente igual, pasaríamos a cambiar nuestras celdas FULL-SUBTRACTOR por celdas HALF-SUBTRACTOR, pues sólo requerimos de la capacidad de decrementar. Cierto que, por regla general, las implementaciones hardware realizan las restas como sumas de operandos negativos (generalmente representados en complemento a 2) y en cambio nosotros, para restar, siempre necesitaremos las celdas HALF-SUBTRACTOR para hacer decrementos. Pero aun así, aunque se hagan las restas como sumas de números negativos, se requerirá de hardware para poder calcular la representación negativa del operando (y curiosamente, en caso de ser complemento a 2, puede requerir de hardware para hacer un incremento). Además, nuestro algoritmo también soportaría la resta como suma de números negativos.

3C TIC (Edición núm. 13) Vol.4 – Nº 2
Junio – septiembre 2015, 113 - 119
Área de Innovación y Desarrollo, S.L.
ISSN: 2254 – 6529

MÉTODOS

Para este apartado, entenderemos que efectivamente disponemos de un operación de sucesor y antecesor, *successor* y *predecessor*, implementadas a nivel hardware como hemos indicado en el apartado anterior, y que nos permiten realizar un incremento y un decremento, respectivamente, en la posición *i-esima* de un operando cualquiera de *n* bits.

Con ello, veremos una primera versión de los algoritmos de suma y resta basándonos en los conceptos que exponemos. Empezaremos con la suma de dos número, *a* y *b*, de longitud *n*.

```
result = a;
for(int i = 0; i < n; i++) {
     if(b[i] == 1) result = successor(result, i);
}
```

Algoritmo de suma 1.

Y el caso de la resta de dos números, *a* y *b*, de longitud *n* sería simétrico:

```
result = a;
for(int i = 0; i < n; i++) {
     if(b[i] == 1) result = predecessor(result, i);
}
```

Algoritmo de resta 1.

Ahora repasemos la tabla dada por Booth para reducir el número de operaciones necesarias, apoyándonos en la propiedad invertible de la operación con la que se construye nuestro cálculo (nuestro cálculo es la suma o resta, y las operaciones con las que se construye son las de sucesor o antecesor):

bit menos significativo	bit extra	Interpretación	Acción
0	**0**	intermedio cadena de 0s	ninguna
0	**1**	final cadena de 1s	operación
1	**0**	comienzo cadena de 1s	operación inversa / inverso misma operación
1	**1**	intermedio cadena de 1s	ninguna

Tabla de acciones de Booth.

Partiendo de la tabla anterior, nuestro algoritmo de Booth aplicado a la suma de dos números, *a* y *b*, de longitud *n* quedaría:

```
result = a;
bitExtra = 0;
for(int i = 0; i < n; i++) {
     swtich(actionBooth(b[i], bitExtra) {
          case ( 0 1 ):
```

3C TIC (Edición núm. 13) Vol.4 – Nº 2
Junio – septiembre 2015, 113 - 119
Área de Innovación y Desarrollo, S.L.
ISSN: 2254 – 6529

```
            result = successor(result, i);
            break;
      case ( 1 0 ):
            result = predecessor(result, i);
            break;
      }
      bitExtra = b[i];
}
```

Algoritmo de suma 2.

Y el caso de la resta de dos números, *a* y *b*, de longitud *n* sería quedando:

```
result = a;
bitExtra = 0;
for(int i = 0; i < n; i++) {
      swtich(actionBooth(b[i], bitExtra) {
            case ( 0 1 ):
            result = predecessor(result, i);
            break;
      case ( 1 0 ):
            result = successor(result, i);
            break;
      }
      bitExtra = b[i];
}
```

Algoritmo de resta 2.

Como decíamos al comienzo, existe una representación, conocida como NAF, que permitiría reducir aún más el número de operaciones que se realizan para la obtención del resultado final. Además, existe una alternativa que soluciona las secuencias de un único 1 en el algoritmo de Booth, ya que en estos casos es más eficiente realizar una única operación, que no las 2 operaciones que realiza Booth buscando simplificar las secuencias de 1s consecutivos.

3C TIC *(Edición núm. 13)* Vol.4 – Nº 2
Junio – septiembre 2015, 113 - 119
Área de Innovación y Desarrollo, S.L.
ISSN: 2254 – 6529

DISCUSIÓN Y CONCLUSIÓN

El aplicar el algoritmo de Booth en la implementación de las operaciones de suma y resta, nos ofrece una alternativa que implicaría cambiar los sumadores combinacionales actuales por una alternativa secuencial, apoyándonos en una o unas operaciones hardware más ligeras, que nos permite prescindir de las celdas FULL-ADDER por celdas HALF-ADDER y HALF-SUBTRACTOR, y además acortar considerablemente la latencia del cálculo de esas operaciones de incremento y/o decremento, con respecto a operaciones de suma y/o resta. Y más aún si estas últimas no implementan técnicas como el adelantamiento de acarreo. Si hicieran uso de adelantamiento de acarreo, aunque acorten la latencia de los acarreos, aumentan considerablemente el hardware necesario, luego el algoritmo que se defiende en este artículo disminuiría aún más el hardware con respecto a una implementación de suma que hiciera uso de esa técnica.

En conclusión, la idea propuesta por Booth, simplifica considerablemente no sólo la implementación de un multiplicador, sino también la de un sumador, además de permitir reducir el número de operaciones necesarias para su cálculo, con el consiguiente aumento del rendimiento.

3C TIC *(Edición núm. 13)* Vol.4 – Nº 2
Junio – septiembre 2015, 113 - 119
Área de Innovación y Desarrollo, S.L.
ISSN: 2254 – 6529

REFERENCIAS BIBLIOGRAFICAS

Booth, A. D., "A method of calculating reciprocal spacings for X-ray reflections from a monoclinic crystal," J. Sci. Instr, Vol. 22, 1945, p. 74. http://dx.doi.org/10.1088/0950-7671/22/4/404

Burks, A., Goldstein, H. and Von Neumann, J., "Logical Design of an Electronic Computing Instrument" (Princeton, 1946).

Booth, A. D. and Britten, K. H. V., "General Considerations in the Design of an Electronic COmputer" (Princeton, 1947).

Booth, A. D., "A signed binary multiplication technique", Q.J. Mech. and Appl. Math. Vol 4, No.2, 1951, pp.236-240. http://dx.doi.org/10.1093/qjmam/4.2.236