Departamento de Cs. e Ingeniería de la Computación
Universidad Nacional del Sur

Arquitectura de Computadoras

Trabajo Práctico Nº 4

Implementación de las Operaciones Básicas: Suma y Resta

Primer Cuatrimestre de 2010

Ejercicios

1. Verificar la validez de las siguientes expresiones analizando todas las combinaciones posibles:

   a) Expresión lógica para la suma:

   $$\begin{aligned} s_i &= x_i \oplus y_i \oplus c_i \\ c_{i+1} &= x_i \cdot y_i + (x_i \oplus y_i) \cdot c_i \end{aligned}$$

   b) Expresión lógica para la resta:

   $$\begin{aligned} s_i &= x_i \oplus y_i \oplus b_i \\ b_{i+1} &= \overline{x}_i \cdot y_i + (\overline{x}_i \oplus y_i) \cdot b_i \end{aligned}$$

2. Reescribir las expresiones para la suma y la resta introducidas en el ejercicio anterior de forma tal que para una posición genérica i, el valor de s_i, c_{i+1} o de b_{i+1} se calcule como una suma de productos entre x_i, y_i, c_i o b_i según corresponda.

3. Esquematizar un sumador paralelo de 32 bits empleando ocho sumadores **CLAA** de cuatro bits.

   a) En configuración *ripple*.

   b) En configuración *Carry Look-Ahead* de dos niveles.

   c) En configuración *Carry Look-Ahead* de tres niveles.

   Calcular en cada caso el tiempo que insume la suma asumiendo que los retardos de los distintos componentes son los siguientes:

   - Para los **CLAA**:
     - $6t$ para obtener la suma.
     - $4t$ para obtener el c_{out}.
     - $2t$ para obtener p y g.
   - Para los **CLAG**:
     - $1t$ para obtener p y g.
     - $2t$ para obtener los c_{n+x}.

4. Suponiendo un retardo de $2t$ en cada uno de los *full adders* de un *ripple adder* de 8 (ocho) bits, llevar adelante las siguientes tareas:

   a) Esquematizar el diagrama asociado a este sumador.

   b) ¿Cuál es el tiempo de retardo máximo que se debe contemplar para el tiempo de suma al operar de forma **sincrónica**?

   c) En el caso de operar de forma **asincrónica**, asumido un carry inicial $c_0 = 1$, determinar el tiempo de suma requerido en cada uno de los siguientes casos:

      1) $X = 1\,1\,0\,1\,0\,0\,1\,0$ e $Y = 0\,0\,1\,0\,1\,1\,0\,1$.
      2) $X = 1\,1\,0\,1\,0\,0\,1\,0$ e $Y = 0\,0\,1\,1\,1\,1\,0\,1$.
      3) $X = 1\,1\,0\,1\,0\,0\,1\,0$ e $Y = 0\,0\,1\,0\,0\,1\,0\,1$.

      Obs: Tener en cuenta que el carry de entrada a una posición cualquiera $i + 1$ se expresa como $c_{i+1} = x_i y_i + (x_i + y_i) \times c_i = g_i + p_i c_i$, donde g_i denota la generación de carry y p_i la propagación de carry.

5. Diseñar un circuito *Carry-Skip Adder* para 32 bits realizando la partición de la siguiente manera:

   a) Usando ocho sumadores ripple de 4 bits cada uno.

   b) Usando ocho sumadores ripple, pero con las siguientes capacidades: 2, 3, 4, 5, 6, 5, 4 y 3 bits.

   Ponderar los retardos en cada caso, asumiendo un retardo t por cada dos niveles de compuertas. Verificar si es el caso que la suma ha sido realizada *en cada bloque* en el tiempo propuesto.

6. En el contexto de un circuito *Carry-Skip Adder* para 32 bits, ¿cuál sería la partición óptima si los distintos bloques tuvieran que ser del mismo tamaño? Ponderar una vez más los retardos, asumiendo un retardo t por cada dos niveles de compuertas, verificando que la suma haya sido realizada *en cada bloque* en el tiempo propuesto.

7. Diseñar un *Carry-Select Adder* de 32 bits con seis bloques en configuración *ripple*, agrupando los *full-adders* de forma tal de optimizar los niveles de compuerta que deben atravezarse para completar la suma. Luego, adaptar el diseño propuesto para que admita 64 bits, usando en esta ocasión siete bloques. Finalmente, comparar ambos resultados en cuanto a cómo creció la complejidad en niveles de compuertas.

   Obs: Tener en cuenta para el diseño que a medida que se avanza hacia las posiciones más significativas se deben incorporar módulos con *full-adders* adicionales.

Referencias

[Bae80] Baer, J. L. *Computer Systems Architecture*. Computer Science Press, 1980.

[HP96] Hennessy, J., and Patterson, D. *Computer Architecture*, second ed. Morgan Kaufmann, 1996.