Filtro FIR

Arquitecturas Avanzadas y de Propósito Específico

Cheuk Kelly Ng Pante (alu0101364544@ull.edu.es)

13 de junio de 2024

Índice general

1. Introducción

FIR es un acrónimo en inglés para *Finite Impulse Response* o Respuesta Finita al Impulso. Se trata de un tipo de filtro digital que si su entrada es un impulso (una delta de Kronecker), su salida es un número limitado de términos no nulos. Este tipo de filtro se utiliza comúnmente en aplicaciones de procesamiento de señales, como en la industria de las telecomunicaciones, el procesamiento de audio y la ingeniería de control.

Su expresión en el dominio n es la siguiente:

$$y_n = \sum_{k=0}^{N-1} b_k x_{n-k}$$

donde x_n es la entrada, y_n es la salida, b_k son los coeficientes del filtro y N es el orden del filtro.

La estructura básica de un filtro FIR es la siguiente:

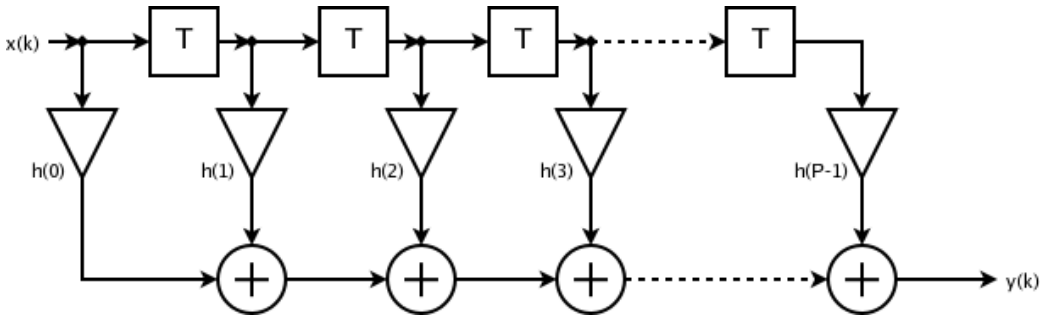

Figura 1.1: Estructura básica de un filtro FIR

2. Características del proyecto

El proyecto se divide en 5 versiones de filtros FIR:

- Versión 1: Filtro FIR
- Versión 2: Incluye el uso de keywords, como **const** y **restrict**
- Versión 3: Desenrrollado manual y optimizacion de bucles, condicionales, etc.
- Versión 4: Utilización de **pragmas**
- Versión 5: Uso de **intrínsecos**

En cada versión incluye la implementación de un filtro FIR, dependiendo de la versión, y la lectura de los valores de los coeficientes (Coeficientes.csv) y los valores de los datos de entrada (musica4.csv).

En cuanto al profiling se optado por usar *clock()*, que es una función que mide el tiempo de ejecución de un programa. En principio se pensó en usar *gprof*, que es una herramienta de análisis de rendimiento para aplicaciones UNIX, pero no se pudo ya que los tiempos de ejecución eran muy pequeños y no se podía medir correctamente.

Además, se ha calculado los ciclos de reloj de cada versión del filtro FIR. Para ello, se ha creado una función utilizando la instrucción *rdtsc* (Real Time Stamp Counter) que mide el número de ciclos de reloj desde el último reinicio. La función es la siguiente:

```
uint64_t rdtsc(){
  unsigned int lo, hi;
  __asm__ __volatile__("rdtsc" : "=a"(lo), "=d"(hi));
  return ((uint64_t)hi << 32) | lo;
}
```

Lo que hace la función es utiliza la instrucción de ensamblador *rdtsc* para leer el contador de tiempo de la CPU, y devuelve este valor como un número de 64 bits pero dividido en dos partes de 32 bits, *lo* es la parte baja y *hi* es la parte alta. Entonces, se ejecuta la instrucción *__asm__ __volatile__* que es una instrucción de ensamblador en línea que se utiliza para incrustar código ensamblador en el código C.

El *profiling* se ha hecho de la siguiente forma:

```
start_cycle = rdtsc();
start = clock();
// --- Codigo a medir ---
end = clock();
end_cycle = rdtsc();
```

Por otra parte, para calcular el tiempo de ejecución y los ciclos de reloj se ha repetido N veces, para obtener un promedio de los valores. En este caso, se ha decidido hacer 1000 repeticiones, pero se puede cambiar el valor de N ya que se define como una constante en cada versión del filtro FIR.

```
#define REPETICIONES 1000

// Aplicacion del filtro FIR
for (i = 0; i < REPETICIONES; i++) {
  start_cycle = rdtsc();
  start = clock();
  firfilter(vector_coef, vector_in, result);
  end = clock();
  end_cycle = rdtsc();

  elapsed = (double)(end - start) * 1000.0 / CLOCKS_PER_SEC;
  mean_time[i] = elapsed;
  mean_cycles[i] = end_cycle - start_cycle;
}
```

3. Implementación

Lectura de los coeficientes

4. Bibliografía

Bibliografía

[1] Ng Pante, C. (2001). Titulo. Nombre pagina web. Recuperado de `http://url.com`

[2] `https://es.wikipedia.org/wiki/FIR_(Finite_Impulse_Response)`

[3] `https://es.wikipedia.org/wiki/Diseo_de_Filtros_de_Respuesta_Finita_al_Impulso`

[4] `https://en.wikipedia.org/wiki/Gprof`