Universidad de Buenos Aires
Facultad de Ingeniería

Especialización en Microelectrónica

Arquitectura de Sistemas Digitales

Filtros FIR

Alumno: Agustín Galdeman
Fecha: 19 de mayo de 2025

Buenos Aires, Argentina

Resumen

Este informe describe el diseño, implementación y verificación de un filtro FIR (Finite Impulse Response) en Verilog. Se analizan las decisiones técnicas tomadas, incluyendo representación numérica, ancho de registros, forma de cálculo de los coeficientes, y estrategias de testeo empleadas para validar su correcto funcionamiento.

1. Objetivo

Implementar un filtro FIR de 16 coeficientes, con entrada y salida en punto fijo con signo (formato `Q15`), que opere en hardware secuencial utilizando multiplicaciones y acumulaciones en registros internos.

2. Estructura del Filtro

El módulo principal, llamado `fir`, implementa un filtro FIR clásico basado en la ecuación:

$$y[n] = \sum_{k=0}^{15} h[k] \cdot x[n-k]$$

Para lograr esto, se emplea:

- Un **registro FIFO** de 16 posiciones (`fifo_x`) que almacena las muestras de entrada.
- Una **memoria de coeficientes** precargada con valores constantes de 16 bits con signo (`coef`).
- Una **unidad de multiplicación-acumulación** secuencial que realiza un producto y suma por ciclo de reloj.

3. Representación Numérica y Tamaños de Registro

- La entrada `x` y salida `y` están definidas como `signed [15:0]`, es decir, 16 bits con signo en complemento a dos.
- El acumulador interno (`acc`) tiene un ancho de 32 bits para prevenir overflow durante la sumatoria de los productos.
- Cada producto parcial entre un coeficiente y una muestra genera un resultado de 32 bits.
- Finalmente, se realiza un truncamiento hacia 16 bits al asignar el resultado a `y`.

Esto permite trabajar con señales en formato `Q15` (valores fraccionarios en el rango $[-1, +0{,}99997]$ aproximadamente) sin perder precisión en la acumulación.

4. Obtención de los Coeficientes

Los 16 coeficientes fueron calculados utilizando el software **MATLAB** mediante la función `fir1`:

```
h = fir1(15, 0.25, 'low');              % Filtro pasa bajos, orden 15, Wc = 0.2
h_fixed = round(h * 32767);             % Conversi n a Q15
```

Los coeficientes fueron luego truncados a 16 bits con signo y codificados directamente en el archivo Verilog como:

```
coef[0]  <= -16'sd2252;
coef[1]  <= -16'sd1122;
...
coef[15] <= -16'sd29834;
```

5. Testbench y Verificación

Se implementó un testbench para verificar dos respuestas clave del filtro:

1. Respuesta al Impulso

Se aplicó un impulso de entrada (valor máximo 32767 seguido de ceros). La salida coincidió con los 16 coeficientes cargados, confirmando que el filtro implementa correctamente la convolución.

2. Respuesta al Escalón

Se alimentó al filtro con una secuencia constante de 32767 para simular una entrada escalón. La salida fue creciente hasta estabilizarse en el valor final de:

$$y_{\text{final}} = -8755001214 \bmod 2^{16} = \boxed{19606}$$

Esto se debe a la acumulación de productos que excede los 16 bits, pero al truncar el resultado a la salida, se produce un **wraparound** válido. Como este comportamiento es consistente con la definición de truncamiento en Verilog, no se considera overflow funcional.

6. Detección de Overflow

Durante la simulación se chequeó que la salida no excediera los límites de un entero con signo de 16 bits:

$$-32768 \leq y \leq +32767$$

El diseño previno overflow real utilizando un acumulador interno de 32 bits y truncamiento controlado. El testbench detectó valores fuera de este rango si llegaban a aparecer.

7. Conclusión

El filtro FIR fue validado correctamente:

- Su salida frente al impulso coincide con los coeficientes cargados.
- La respuesta al escalón se estabiliza como se espera, sin comportamiento errático.
- El uso de registros de mayor ancho internamente previene errores por overflow.