Especificación de Diseño: Unidad MAC con Algoritmo de Booth

1. Introducción y Motivación

La unidad **Multiply-Accumulate (MAC)** es el corazón de los sistemas de Procesamiento Digital de Señales (DSP). Operaciones críticas como filtros FIR/IIR, Transformadas Rápidas de Fourier (FFT) y, más recientemente, el cálculo de Tensores en Inteligencia Artificial, dependen fundamentalmente de la operación:

$$S = S + (A \times B)$$

El objetivo de este proyecto es diseñar, implementar y verificar una unidad MAC eficiente en hardware, optimizada para manejar números con signo (Complemento a 2) sin necesidad de conversores externos.

2. Roadmap de Desarrollo (Fases del Proyecto)

El desarrollo se estructura en cuatro fases evolutivas, permitiendo una verificación incremental y robusta.

Fase 1: El Multiplicador Booth Radix-2 (La Base)

- **Estado: COMPLETADO**
- **Descripción:** Implementación del algoritmo de Booth original.
- **Microarquitectura:**
    - Máquina de Estados Finitos (FSM) que controla el flujo bit a bit.
    - Datapath con capacidad de suma y resta aritmética.
    - Manejo nativo de Complemento a 2.
- **Rendimiento:** Latencia de N ciclos para operandos de N bits (16 ciclos para 16 bits).

Fase 2: Conversión a Unidad MAC (El Acumulador)

- **Estado: COMPLETADO**
- **Descripción:** Integración de un sumador de acumulación y registros de almacenamiento.
- **Reto Técnico:** Sincronización entre el fin de la multiplicación y la actualización del acumulador.
- **Arquitectura:**
    - Expansión del resultado a 32 bits (producto) + 8 bits (guarda) = 40 bits totales.
    - Señal de control load para cargar valores iniciales o acumular sobre el histórico.
- **Resultado Verificado:** Validación exitosa de operaciones secuenciales ($10 \times 5 + 2 \times -3 +$ ...).

Fase 3: Evolución a Radix-4 (Optimización de Área/Tiempo)

- **Estado: EN PROCESO**
- **Objetivo:** Reducir la latencia a la mitad ($N/2$ ciclos).
- **Estrategia:**
  - Implementar **Codificación Booth Modificada (Radix-4)**.
  - Analizar ventanas de **3 bits** (x_{i+1}, x_i, x_{i-1}) en lugar de 2.
  - Generar operaciones complejas: $0, \pm M, \pm 2M$.
  - Realizar desplazamientos de 2 bits por ciclo.
- **Meta de Hardware:** Mantener el uso de LUTs bajo mientras se duplica la velocidad de cálculo.

Fase 4: Pipelining (Optimización de Velocidad)

- **Estado: PLANEADO**
- **Objetivo:** Aumentar la frecuencia máxima de reloj (F_{max}) y el caudal de datos (Throughput).
- **Estrategia:**
  - Romper el camino crítico (Critical Path) insertando registros entre las etapas de lógica combinacional.
  - Permitir que la unidad empiece una nueva multiplicación antes de que termine la anterior.

MICROARQUITECTURA