

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 ($10 \times 5 + 2 \times -3 + \dots$).

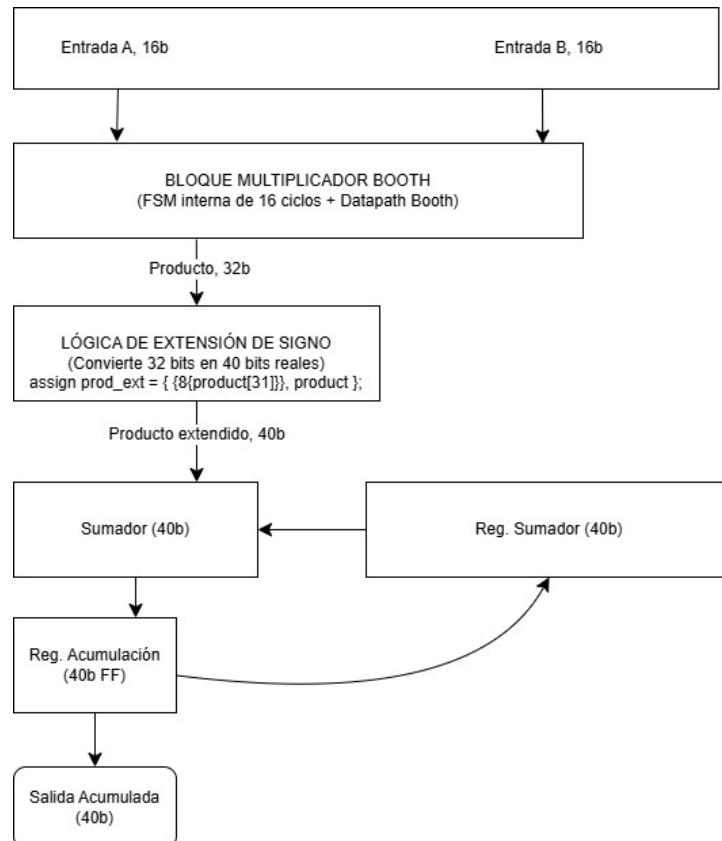
Fase 3: Evolución a Radix-4 (Optimización de Área/Tiempo)

- **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



Asignacion	Descripción	No.	Name	Descripción detallada	Estado	Comments
Test Cases	Basic Check	1	MAC_BASIC_TEST	Multiplicación básica positiva (10×10) y acumulación simple para confirmar vida.	TRUE	Básico
	Signos Mixtos	2	MAC_SIGNED_MIX_TEST	Random A (pos/neg) y Random B (pos/neg). Verifica manejo de complemento a 2.	TRUE	Básico
	Acumulación	3	MAC_ACCUM_LOOP_TEST	Realizar 20 operaciones seguidas sin limpiar el acumulador.	básico
	Ceros	4	MAC_ZERO_OPS_TEST	Operaciones con A=0 o B=0. El acumulador no debe cambiar.	FALSE	Pendiente Fase 3
	Corner Cases	5	MAC_MAX_MIN_TEST	Operandos MAX_INT y MIN_INT para estresar el ancho de banda.	FALSE	Pendiente
	Reset On-Fly	6	MAC_RST_MID_OP_TEST	Aplicar Reset en medio de una operación. FSM debe ir a IDLE y Acum a 0.	FALSE	Pendiente
Asserts	Overflow	1	ASSERT_NO_OVERFLOW	El resultado no debe exceder los 40 bits del acumulador.	FALSE	Pendiente
BFM	Random	1	SET_RAND_INPUTS	Randomiza A y B con valores válidos de 16 bits.	FALSE	Pendiente
COVER	FSM	1	CVG_FSM_STATES	Visitar todos los estados (IDLE, LOAD, CALC, DONE).	TRUE	Basico
	Signos	2	CVG_SIGN_CROSS	Bins para todas las combinaciones: ++, +- , -+, --.	FALSE	Meta
	Valores	3	CVG_VAL_RANGES	Bins para Cero, Max Pos, Max Neg, y rangos medios.	FALSE	Pendiente
	Booth Ops	4	CVG_BOOTH_OPS	(Fase 3) Asegurar que ocurrieron operaciones +M, -M, +2M, -2M, 0.	FALSE	Fase 3