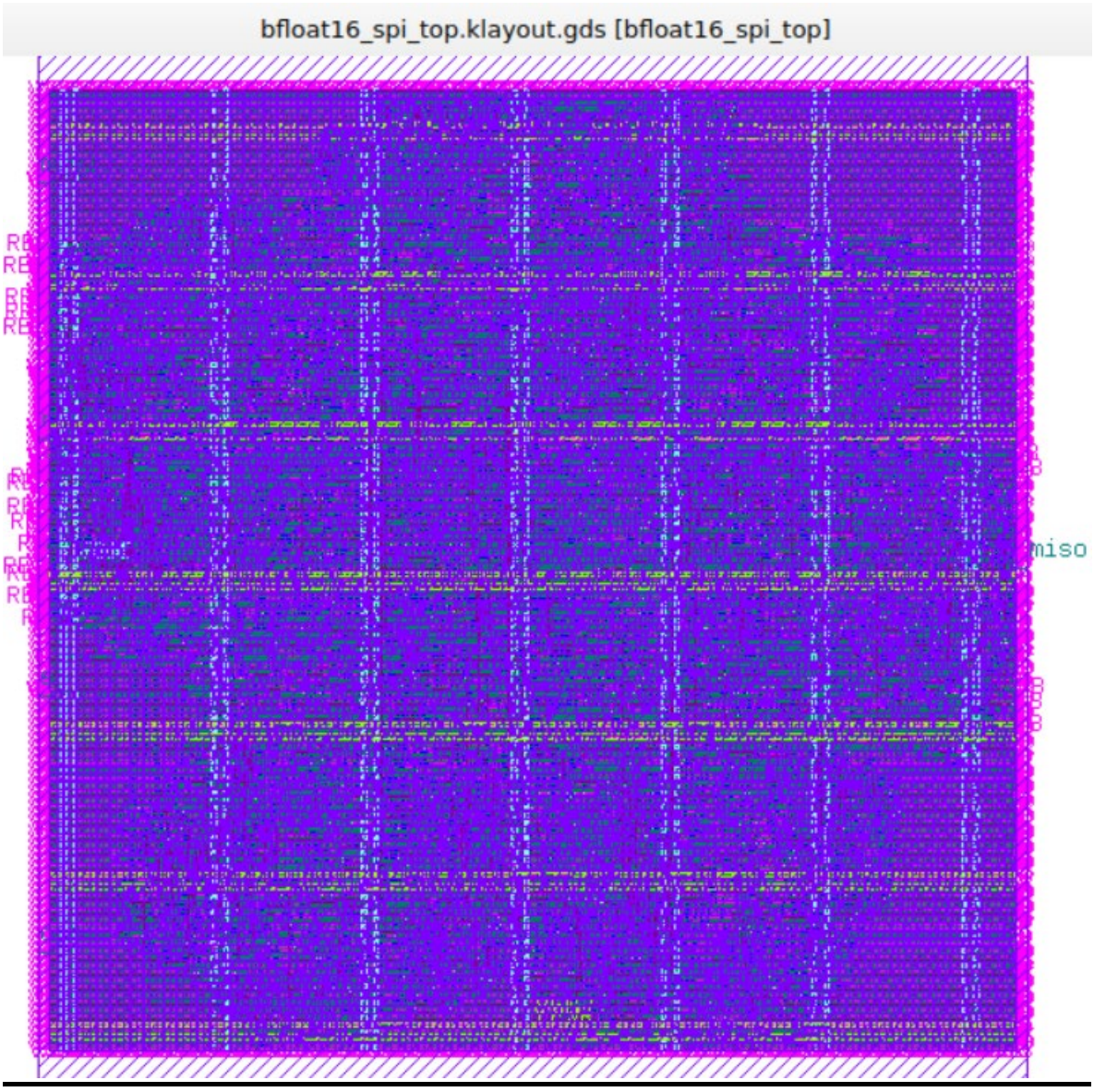
Microdesafío #3

Diseño de Circuitos Integrados

FECHA: 23/10/2025

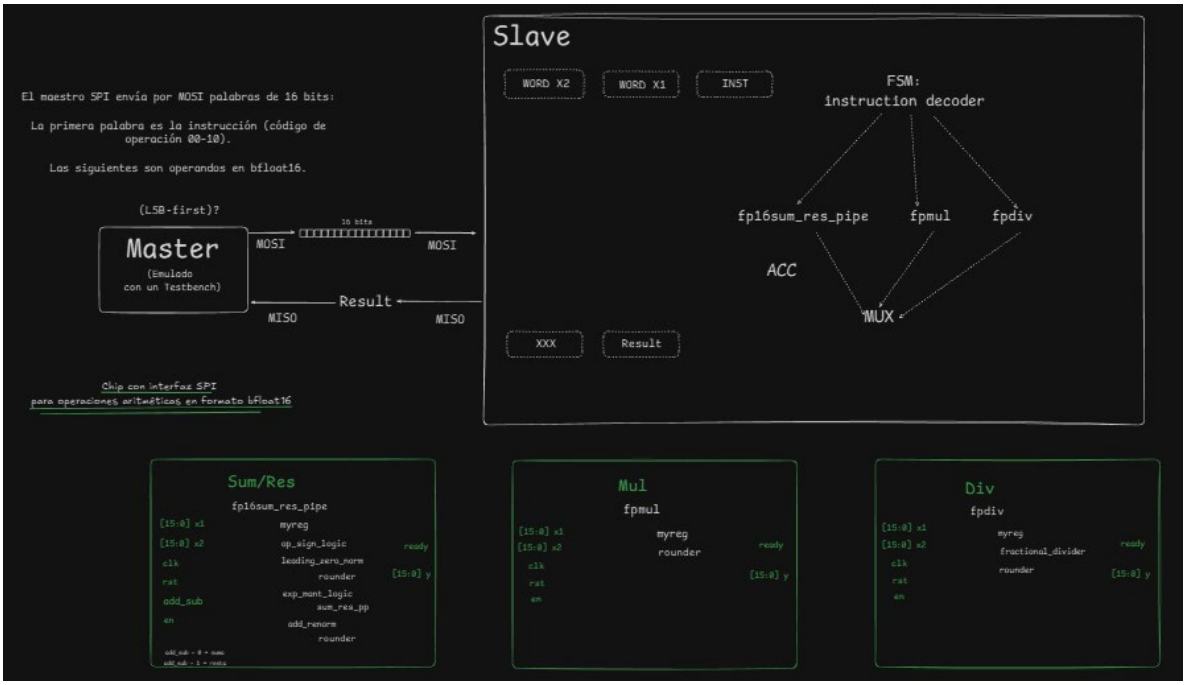
Grupo: Miguel Angel Tobón Morales - Santiago Escobar

Chip con interfaz SPI para realizar operaciones aritméticas en formato bfloat16



Resumen:

Este proyecto implementa un chip acelerador aritmético con interfaz SPI para operaciones en formato bfloat16 (Brain Floating Point 16). El diseño integra un acumulador interno (ACC) de 16 bits y módulos hardware dedicados para suma/resta, multiplicación y división, permitiendo ejecutar 9 instrucciones diferentes que combinan operaciones puntuales y acumulativas. La comunicación ocurre mediante transacciones SPI full-duplex donde el LSB se transmite primero. El sistema incluye un decodificador de instrucciones en una FSM que gestiona el flujo de datos entre registros de entrada SIPO/PISO y las unidades funcionales. La arquitectura soporta operaciones secuenciales como SUM y SUB que acumulan múltiples operandos sobre el registro ACC, además de operaciones puntuales (ADD2, SUB2, MPY2, DIV2) que no modifican el acumulador. El diseño se caracteriza mediante síntesis con OpenLane para obtener métricas de área, timing y layout final en tecnología abierta, validando el comportamiento con testbenches en Verilog y verificación de resultados mediante scripts en Python.



Conceptos:

- **interfaz SPI**

Serial Peripheral Interface es un protocolo de comunicación síncrono full-duplex que opera en modo maestro-esclavo. Utiliza cuatro señales principales: MOSI (Master Out Slave In), MISO (Master In Slave Out), SCLK (reloj) y CS (chip select). En este chip, se configura para transmitir palabras de 16 bits donde el bit menos significativo (LSB) se envía primero. La interfaz permite al maestro enviar instrucciones y operandos mientras simultáneamente recibe resultados, maximizando el throughput en operaciones aritméticas consecutivas.

- **bfloat16**

Formato de punto flotante de 16 bits optimizado para aprendizaje automático. Compone: 1 bit de signo, 8 bits de exponente (igual que FP32) y 7 bits de mantisa. A diferencia de IEEE FP16 (5-10-5), sacrifica precisión para expandir el rango dinámico, facilitando la conversión con FP32 mediante truncamiento de la mantisa. Este diseño lo emplea porque ofrece balance ideal entre precisión suficiente para redes neuronales y eficiencia hardware, reduciendo la complejidad de multiplicadores y sumadores comparado con FP32.

- **LSB**

Least Significant Bit (bit menos significativo). En transmisión serial, indica que el bit de menor peso (bit 0) del dato se envía primero por el bus. Esta convención afecta directamente el diseño de registros de desplazamiento SISO/SIPO/PISO, ya que el hardware debe alinear correctamente los bits recibidos para formar la palabra paralela de 16 bits antes de procesarla en las unidades aritméticas.

- **SIPO**

Serial-In Parallel-Out (registro de desplazamiento). Es un componente esencial en la interfaz de recepción: captura los bits de MOSI secuencialmente (uno por ciclo de SCLK) y, tras recibir 16 bits, los presenta en paralelo al instruction decoder y al banco de registros de operandos. En este chip, el SIPO se activa con el flanco de SCLK y habilita la palabra completa cuando se completa la transmisión de los 16 bits, sincronizándola con la FSM de control.

- **PISO**

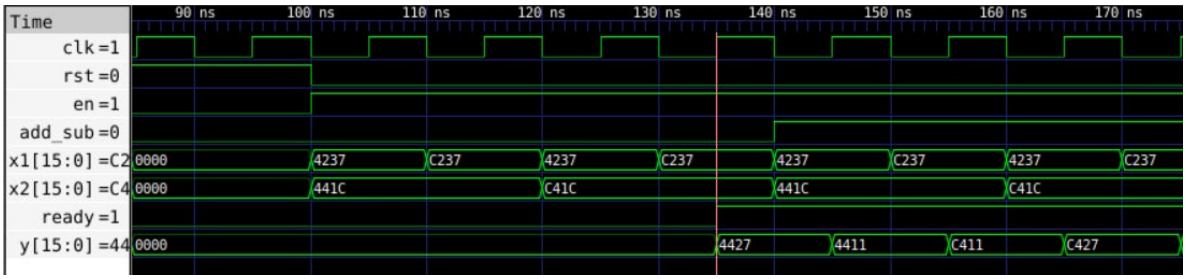
Parallel-In Serial-Out (registro de carga paralela, desplazamiento serial). Se usa en la etapa de transmisión del resultado por MISO. Cuando la unidad funcional completa una operación, el PISO carga el resultado de 16 bits en paralelo y lo desplaza bit a bit (LSB primero) hacia el maestro, permitiendo la respuesta full-duplex. En instrucciones como LOAD_ACC o al final de SUM/SUB, el PISO habilita la salida secuencial del resultado acumulado mientras el maestro envía datos de relleno.

Desarrollo

A. Simulación en IcarusVerilog y SynthesisExploration para caracterización del chip

Cada operación en bfloat16 tarda:

- **4 ciclos** para suma/resta



SYNTH_STRATEGY	Gates	Area (μm^2)	Worst Setup Slack (ns)	Total -ve Setup Slack (ns)
AREA 0	1007	13973.601600	5.6838602100386595	0.0
AREA 1	1003	13884.696000	4.8921263980067495	0.0
AREA 2	1046	14027.428800	4.509858392159427	0.0
AREA 3	1451	16533.946800	6.9948280258935105	0.0
DELAY 0	1133	15485.639400	3.2585419729263227	0.0
DELAY 1	1122	15243.341400	0.425139935289539	0.0
DELAY 2	1147	15511.003200	2.200399924549241	0.0
DELAY 3	1091	15178.098600	-0.3389848497851552	-4.422950131333836
DELAY 4	1307	16970.839200	6.291306320644187	0.0

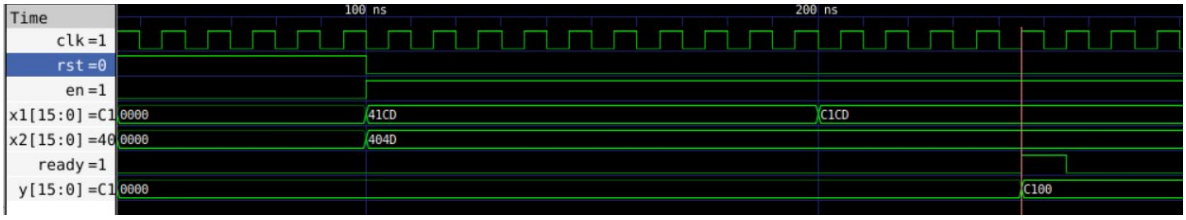
10ns – 6.995ns = 3.005ns | 332.779 MHz

- **3 ciclos** para multiplicación

SYNTH_STRATEGY	Gates	Area (μm^2)	Worst Setup Slack (ns)	Total -ve Setup Slack (ns)
AREA 0	709	7963.888000	-0.11740297955002438	-0.11740297955002438
AREA 1	705	7885.062400	-0.735788339971172	-1.298239774651365
AREA 2	707	7876.304000	0.845800786681853	0.0
AREA 3	1093	9979.571200	4.689171850205597	0.0
DELAY 0	901	10072.160000	2.48950067607033	0.0
DELAY 1	883	9669.273600	2.281402686895227	0.0
DELAY 2	893	9815.664000	3.514658885631371	0.0
DELAY 3	896	9918.262400	3.3577341857329897	0.0
DELAY 4	840	9042.422400	1.0208287837317684	0.0

10ns – 4.689ns = 5.311ns | 188.288MHz

- **14 ciclos** para división



SYNTH_STRATEGY	Gates	Area (μm²)	Worst Setup Slack (ns)	Total -ve Setup Slack (ns)
AREA 0	445	5169.958400	3.4215768964358397	0.0
AREA 1	447	5179.968000	3.4224104519063028	0.0
AREA 2	445	5158.697600	2.9766741025706214	0.0
AREA 3	669	6413.651200	5.626006042500005	0.0
DELAY 0	549	6137.136000	4.41987126075487	0.0
DELAY 1	518	5878.137600	4.283967080110032	0.0
DELAY 2	512	5876.886400	5.254896666544172	0.0
DELAY 3	518	5823.084800	3.5356736317250235	0.0
DELAY 4	624	6612.592000	3.037830517567115	0.0

10ns – 5.626 ns = 4.374ns | 228.624MHz

La frecuencia global máxima del chip está limitada por la operación con el mayor retardo (ruta crítica).

Operación	Path delay (ns)	f_max individual (MHz)
Sum/Res	3.003	333.0
Mul	5.311	188.3
Div	4.374	228.6

$$fmax, chip \approx 188.3\ MHz$$

Por tanto, aunque el sumador podría funcionar a 333 MHz, el multiplicador no; el reloj debe ajustarse a la unidad más lenta.

Se elige una frecuencia para el chip de *150 MHz = 6.667 ns* & la estrategia de síntesis de *AREA 3*

Sum / Res Caracterizado

SYNTH_STRATEGY	Gates	Area (μm²)	Worst Setup Slack (ns)	Total -ve Setup Slack (ns)
AREA 0	1007	13973.601600	2.350860461726226	0.0
AREA 1	1003	13884.696000	1.5591262056050943	0.0
AREA 2	1046	14027.428800	1.1768581997577716	0.0
AREA 3	1451	16533.946800	3.661828055536466	0.0
DELAY 0	1133	15485.639400	-0.0744577753861106	-0.0744577753861106
DELAY 1	1122	15243.341400	-2.907859813022894	-53.965118345891995
DELAY 2	1147	15511.003200	-1.1326007119416368	-18.961385886999942
DELAY 3	1091	15178.098600	-3.6719845980975885	-71.08294598576094
DELAY 4	1307	16970.839200	2.9583063502871427	0.0

Mul Caracterizado

SYNTH_STRATEGY	Gates	Area (μm²)	Worst Setup Slack (ns)	Total -ve Setup Slack (ns)
AREA 0	769	11758.672800	2.8558847206551117	0.0
AREA 1	770	11696.832000	2.7352723082947317	0.0
AREA 2	763	11711.536200	2.740403093125843	0.0
AREA 3	1014	13049.013600	3.525635438941673	0.0
DELAY 0	976	13626.446400	2.681121844870975	0.0
DELAY 1	975	13588.268400	2.6846414740031834	0.0
DELAY 2	1022	13988.759400	2.498269217766811	0.0
DELAY 3	978	13847.387400	2.4778615416401926	0.0
DELAY 4	872	12747.331800	2.726334790649124	0.0

Div Caracterizado

SYNTH_STRATEGY	Gates	Area (μm²)	Worst Setup Slack (ns)	Total -ve Setup Slack (ns)
AREA 0	464	7975.497600	3.4639596615995716	0.0
AREA 1	466	8000.823600	3.4551493755161844	0.0
AREA 2	456	7902.732600	3.4556536388282306	0.0
AREA 3	647	8879.446800	4.1437894288779455	0.0
DELAY 0	554	8753.308200	2.9432621617441783	0.0
DELAY 1	552	8635.712400	3.6664576856800877	0.0
DELAY 2	562	8726.243400	3.8421085178644327	0.0
DELAY 3	565	8780.751000	3.6306181311193413	0.0
DELAY 4	650	9719.211600	3.9549351579775243	0.0

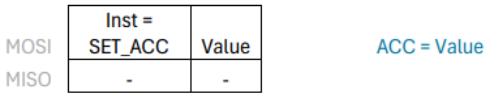
B. Simulación de Funcionalidad de las Instrucciones con banco de pruebas en IcarusVerilog

ZERO

Instrucción 00, encargada de limpiar el acumulador (ACC).

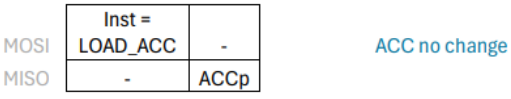
SET_ACC

Instrucción 01, encargada cargar en el acumulador un valor.



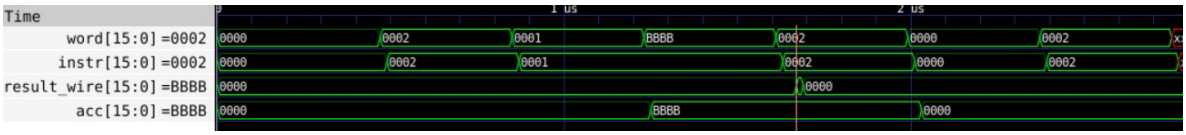
LOAD_ACC

Instrucción 03, encargada de retornar el valor del acumular (ACC).



En la siguiente simulación, se realizan las pruebas de las 3 instrucciones comentadas anteriormente (ZERO, SET_ACC, LOAD_ACC).

```
// Pruebas ACC
test_data [0] = 16'h0002; // LOAD_ACC para obtener resultado
test_data [1] = 16'h0001; // SET_ACC para cargar un valor
test_data [2] = 16'hbbbb;
test_data [3] = 16'h0002;
test_data [4] = 16'h0000; //Zero para limpiar ACC
test_data [5] = 16'h0002;
```



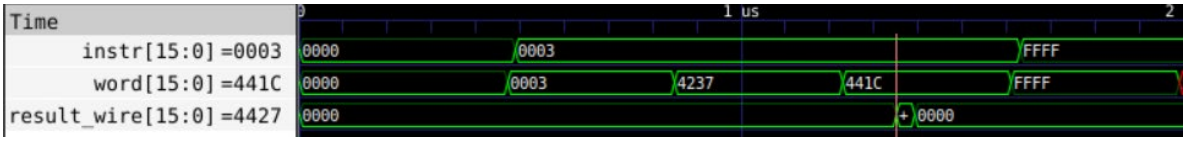
ADD2

Instrucción 04, realiza la suma de dos números, retorna el valor y no altera el acumulador ACC.

MOSI	Inst = ADD2	V1	V2	-
MISO	-	-	-	V1+V2

ACC no change

```
// Tercera operación: Suma
test_data[0] = 16'h0003;
test_data[1] = 16'h4237;
test_data[2] = 16'h441c;
test_data[3] = 16'hFFFF; // Resultado (4427 bfloat16)
```



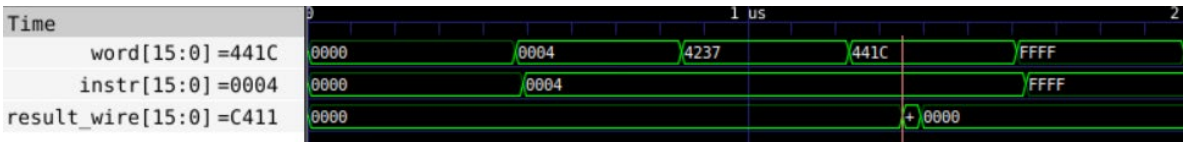
SUB2

Instrucción 05, realiza la resta de dos números, retorna el valor y no altera el acumulador ACC.

MOSI	Inst = SUB2	V1	V2	-
MISO	-	-	-	V1-V2

ACC no change

```
// Cuarta operación: Resta c411
test_data[0] = 16'h0004;
test_data[1] = 16'h4237;
test_data[2] = 16'h441c;
test_data[3] = 16'hFFFF; // Resultado (c411 bfloat16)
```



MPY2

Instrucción 06, realiza la multiplicación de dos números, retorna el valor y no altera el acumulador ACC.

MOSI	Inst = MPY2	V1	V2	-	ACC no change
MISO	-	-	-	V1*V2	

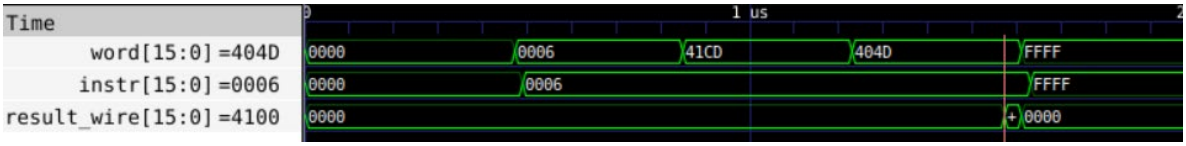
```
// Primera operación: Multiplicación
test_data[0] = 16'h0005;
test_data[1] = 16'h4148;
test_data[2] = 16'h404d;
test_data[3] = 16'hFFFF; // Resultado (4220 bfloat 16)
```

DIV2

Instrucción 07, realiza la división de dos números, retorna el valor y no altera el acumulador ACC.

MOSI	Inst = DIV2	V1	V2	-	ACC no change
MISO	-	-	-	V1/V2	

```
// Segunda operación: División
test_data[0] = 16'h0006;
test_data[1] = 16'h41cd;
test_data[2] = 16'h404d;
test_data[3] = 16'hFFFF; // Resultado (4100 bfloat16)
```



SUM

Instrucción 08, encargada de realizar una sumatoria, suma un flujo de datos de entrada sobre el acumulador ACC, hasta que se envíe un valor de control que detiene el flujo de la sumatoria.

MOSI	Inst = SUB	V1	V2	V3	ACC = ACCp-V1-V2-V3
MISO	-	-	ACCp-V1	ACCp-V1-V2	

```
//Pruebas SUM (Sumatoria)
test_data[0] = 16'h0007;
test_data[1] = 16'h3f80; // 1.0 decimal
test_data[2] = 16'h4000; // 2.0 decimal
test_data[3] = 16'h4040; // 3.0 decimal,
test_data[4] = 16'h3f80; // 1.0 decimal, suma esperada 6.0 (40E0 bfloat16)
test_data[5] = 16'hffff; // Fin de datos
test_data[6] = 16'h0002; // LOAD_ACC para obtener resultado
test_data[7] = 16'hffff;
```

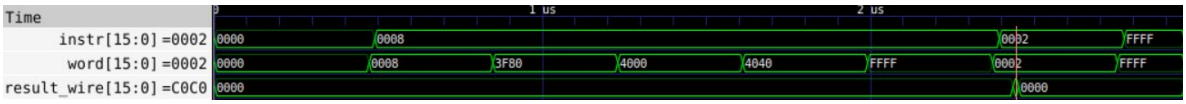
SUB

Instrucción 09, encargada de realizar una resta acumulada, resta un flujo de datos de entrada sobre el acumulador ACC, hasta que se envíe un valor de control que detiene el flujo de la resta acumulada.

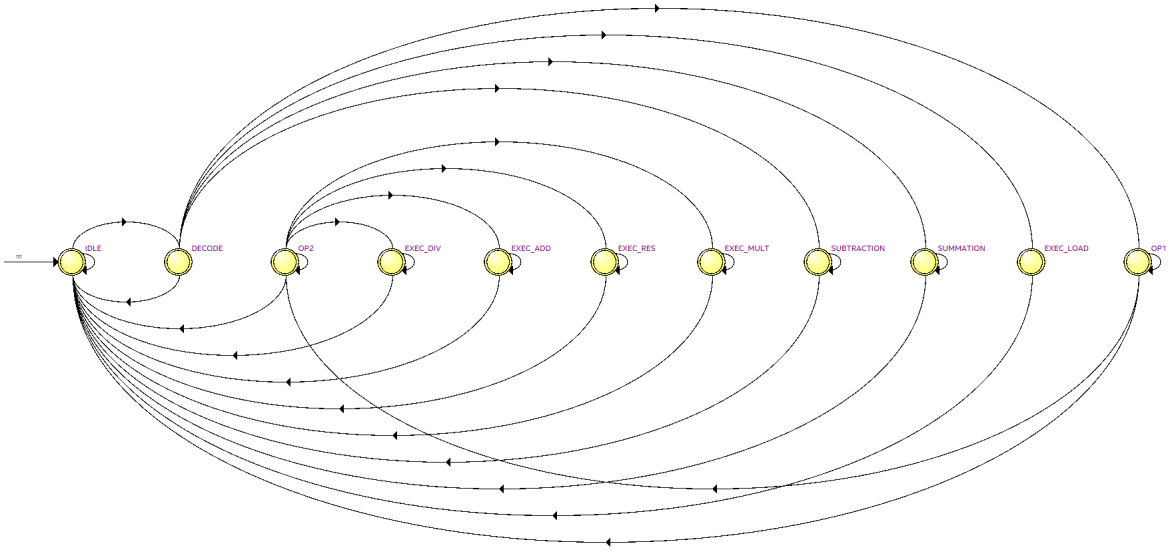
MOSI	Inst = SUB	V1	V2	V3
MISO	-	-	ACCp-V1	ACCp-V1-V2

ACC = ACCp-V1-V2-V3

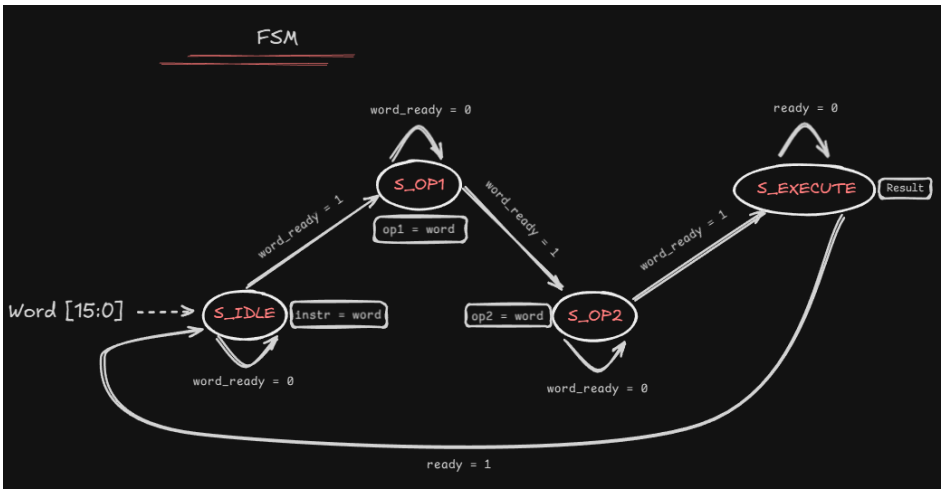
```
//Pruebas SUB (restatoria)
test_data[0] = 16'h0008; // Instrucción SUB
test_data[1] = 16'h3f80; // 1.0 decimal
test_data[2] = 16'h4000; // 2.0 decimal
test_data[3] = 16'h4040; // 3.0 decimal, resta esperada -6.0 (c0c0 bfloat16)
test_data[4] = 16'hffff; // Fin de datos
test_data[5] = 16'h0002; // LOAD_ACC para obtener resultado
test_data[6] = 16'hffff;
```



C. Máquina de Estados



El diseño implementa una máquina de estados finitos síncrona que funciona como el cerebro del chip, coordinando secuencialmente todas las operaciones. Esta máquina es binaria y totalmente dependiente del reloj, lo que significa que cada transición y decisión ocurre en flancos ascendentes del clock, asegurando un comportamiento predecible y sincronizado con la interfaz SPI.



La máquina se organiza en un flujo natural de cuatro etapas principales, comenzando siempre en el estado de reposo (IDLE), donde espera pacientemente a que el maestro SPI envíe la primera palabra de 16 bits. Cuando detecta una instrucción válida, avanza al estado de decodificación (DECODE), donde interpreta el código de operación y determina qué camino seguir. Esta decisión es crucial porque cada instrucción tiene requisitos diferentes: algunas necesitan cero operandos, otras uno o dos, y algunas operan sobre un flujo continuo de datos.

Para las instrucciones que manipulan dos números específicos, como las operaciones aritméticas básicas, la máquina entra en la fase de captura de operandos. Primero pasa por OP1 para almacenar el primer valor y luego a OP2 para el segundo. Una vez que ambos datos están disponibles, salta a los estados de ejecución (EXEC_ADD, EXEC_RES, EXEC_MULT, EXEC_DIV), donde activa la unidad aritmética correspondiente. Estos estados son persistentes: la máquina no avanza hasta que

recibe una señal de "listo" (ready) de la unidad funcional, garantizando que la operación se complete correctamente antes de liberar el bus.

Una característica es el manejo del acumulador (ACC), un registro de 16 bits que mantiene su valor entre operaciones. Las instrucciones SET_ACC y LOAD_ACC permiten escribir y leer este registro respectivamente, mientras que ZERO lo reinicia. Para operaciones de flujo como SUM y SUB, la máquina entra en estados especiales que ciclan indefinidamente, acumulando o restando cada nuevo dato que llega, hasta recibir un valor especial (0xFFFF) que indica el fin de la secuencia.

El diseño es esencialmente secuencial y no bloqueante, ya que cada estado espera condiciones específicas antes de transicionar, evitando carreras de datos. Sin embargo, presenta **limitaciones funcionales**: no implementa las instrucciones MAC y MAS (multiplicación-acumulación), y la instrucción LOAD carece de verificación completa de la transmisión. Además, ante un código de operación inválido, simplemente retorna al reposo sin reportar error, lo que podría dificultar el depurado del sistema.