Instituto Tecnológico de Costa Rica

Escuela de Ingeniería Electrónica

Programa de Licenciatura en Ingeniería Electrónica

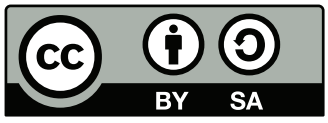
Optimización de un acelerador para multiplicación matricial mediante computación aproximada

Informe de Trabajo Final de Graduación para optar por el título de
Ingeniero en Electrónica

con el grado académico de
Licenciatura

Carlos Adrián Cerdas Mora

Cartago, 15 de noviembre de 2024



Declaro que el presente documento de Trabajo Final de Graduación ha sido realizado enteramente por mi persona, al utilizar literatura referente al tema e introducir conocimientos y resultados experimentales propios.

En los casos en que he utilizado bibliografía he procedido a indicar las fuentes mediante las respectivas citas bibliográficas. En consecuencia, asumo la responsabilidad total por el trabajo de tesis realizado y por el contenido del presente documento.

Carlos Adrián Cerdas Mora

Cartago, 15 de noviembre de 2024
Céd: 1-1783-0309

Instituto Tecnológico de Costa Rica
Escuela de Ingeniería Electrónica
Trabajo Final de Graduación
Acta de Aprobación

Defensa de Trabajo Final de Graduación
Requisito para optar por el título de Ingeniero en Electrónica
Grado Académico de Licenciatura

El Tribunal Evaluador aprueba la defensa del Trabajo Final de Graduación denominado *Optimización de un acelerador para multiplicación matricial mediante computación aproximada*, realizado por el señor Carlos Adrián Cerdas Mora, y, hace constar que cumple con las normas establecidas por la Escuela de Ingeniería Electrónica del Instituto Tecnológico de Costa Rica.

Miembros del Tribunal Evaluador

FRANCISCO OMAR NAVARRO HENRIQUEZ (FIRMA)
Firmado digitalmente por FRANCISCO OMAR NAVARRO HENRIQUEZ (FIRMA)
Fecha: 2024.11.28 12:07:48 -06'00'

M.Sc. Francisco Navarro Henríquez
Profesor Lector

JUAN CARLOS JIMENEZ ROBLES (FIRMA)
Firmado digitalmente por JUAN CARLOS JIMENEZ ROBLES (FIRMA)
Fecha: 2024.11.20 00:18:01 -06'00'

Ing. Juan Carlos Jiménez Robles
Profesor Lector

JORGE ALBERTO CASTRO GODINEZ (FIRMA)
PERSONA FISICA, CPF-01-1236-0930.
Fecha declarada: 19/11/2024 10:48:46 AM
Razón: Aprobación

Dr.-Ing. Jorge Castro-Godínez
Profesor Asesor

Cartago, 15 de noviembre de 2024

Resumen

La Inteligencia Artificial (IA) ha transformado la tecnología, al mejorar la eficiencia en diversas industrias. Sin embargo, las aplicaciones de IA requieren un alto rendimiento computacional, lo que plantea desafíos de recursos y consumo energético. Los aceleradores de hardware especializados, como los de multiplicación de matrices, mejoran el rendimiento y la eficiencia energética.

El trabajo se enfoca en investigación y puesta en práctica de técnicas para computación aproximada, como el truncamiento y la cuantización, con optimización en la multiplicación de matrices en sistemas de IA. El objetivo es conocer la mejor aproximación que equilibre eficiencia y precisión de los recursos disponibles para 4 distintos niveles de profundidad en cada tipo de diseño. En particular, la aproximación conjunta con 4 bits mostró ser la mejor opción entre todas las configuraciones evaluadas. Este diseño maximiza la eficiencia computacional y mantiene un nivel de error aceptable, lo que la convierte en una opción ideal para diversas aplicaciones que requieren un buen equilibrio entre rendimiento y precisión, como en sistemas embebidos o dispositivos de bajo consumo.

Palabras clave: Computación aproximada, aceleradores, multiplicación de matrices, inteligencia artificial.

Abstract

Artificial Intelligence (AI) has transformed technology by improving efficiency in various industries. However, AI applications require high computational performance, which poses resource and power consumption challenges. Specialized hardware accelerators, such as matrix multiplication accelerators, improve performance and energy efficiency. The work focuses on investigating and evaluating approximate computing techniques, such as truncation and quantization, to optimize matrix multiplication in AI systems. The objective is to find the best approximation that balances efficiency and accuracy as a function of available resources for 4 different depth levels in each type of approximation. In particular, the joint approximation with 4 bits was shown to be the best choice among all the evaluated configurations. This combination maximizes computational efficiency and maintains an acceptable error level, making it an ideal choice for various applications that require a good balance between performance and accuracy, such as in embedded systems or low-power devices.

Keywords: Approximate computing, accelerators, matrix multiplication, artificial intelligence.

a mis queridos padres, hermanos y novia

Agradecimientos

El resultado de este trabajo no hubiese sido posible sin el conocimiento y el apoyo del Dr. Jorge Castro-Godínez, el cual me brindó las herramientas y la oportunidad para el desarrollo del mismo, gracias por creer en mi. Quiero agradecer a mi padres, Carlos Cerdas Calderón y Sonia Mora Castro, que me hicieron la persona que soy hoy con el amor y cariño que solo ellos me pudieron dar, además del apoyo incondicional no solo en mi etapa universitaria pero también a lo largo de toda mi vida. A los profesores que me ayudaron a convertirme en el profesional que soy y a mis compañeros de carrera que estuvieron a mi lado en este proceso.

Un agradecimiento muy especial para mis abuelos, Carlos (Q.E.D), Miguel, Josefa y Nora, quienes con mucho amor nunca dudaron de mis capacidades y me dieron la inspiración para cumplir todos mis objetivos, les debo todo lo que soy.

Carlos Adrián Cerdas Mora

Cartago, 15 de noviembre de 2024

Índice general

Índice de figuras

Índice de tablas

Lista de abreviaciones

Abreviaciones

CNN	Redes Neuronales Convolucionales
CPM	Método de Ruta Crítica
CPT	Rastreo de Ruta Crítica
CPU	Unidad Central de Procesamiento
CSR	Unidad de Control y Registro de Estados
CSV	Archivo de Valores Separados por Coma
GeMM	Multiplicador General de Matrices
GPU	Unidad de Procesamiento Gráfico
IA	Inteligencia Artificial
IoT	Internet de las Cosas
MAC	Matriz de Multiplicación Acumulada
MED	Error de Distancia Promedio
RISC	Computadora con Conjunto de Instrucciones Reducido
RTL	Nivel de Transferencia de Registros
SNAX	Extensión de Acelerador para Snitch
SoC	Sistema en Chip
TCDM	Memoria de Datos Estrechamente Acoplada
TSMC	Compañía de Manufactura de Semiconductores Taiwanesa
VCC	Voltaje Digital de Colector
VDD	Voltaje Digital de Drenador
VLSI	Integración a Escala muy Grande

Capítulo 1

Introducción

Debido al rápido avance en el desarrollo tecnológico, el uso de dispositivos inteligentes para realizar las actividades diarias es cada vez mayor, ya que aumenta la eficiencia de cada persona en las labores que realiza. Dispositivos como el computador o teléfonos inteligentes forman parte de nuestra cotidianidad ya que permiten realizar tareas complejas en poco tiempo y de una forma más simple.

Para este contexto de desarrollo tecnológico, una de las innovaciones más recientes y de alto impacto es el uso de inteligencia artificial (IA). Dicha tecnología aumenta la productividad, reduce los costes y acelera el desarrollo de productos, al influir significativamente en la vida cotidiana y en diversas industrias de naciones a nivel mundial [1]. Estos algoritmos de inteligencia artificial tienen un impacto significativo en la sociedad al mejorar las operaciones empresariales, mejorar la experiencia de los clientes, aumentar la productividad y facilitar la toma de decisiones basada en datos obtenidos por medio del acceso a los grandes datos de Internet [2].

En la era de la IA la demanda de computación de alto rendimiento ha crecido de manera exponencial, impulsada por aplicaciones que requieren el procesamiento de grandes volúmenes de datos y el entrenamiento de modelos complejos. Entre estas aplicaciones, el uso intensivo de operaciones matemáticas, como la multiplicación de matrices, tiene un rol fundamental para algoritmos de aprendizaje profundo, visión por computadora, y procesamiento de lenguaje natural, entre otros. Sin embargo, la implementación eficiente de estas operaciones en *hardware* presenta un desafío, debido a las limitaciones de recursos computacionales, consumo energético y tiempo de procesamiento.

Para superar estos obstáculos, los aceleradores de *hardware* especializados, como los aceleradores de multiplicación de matrices, han surgido como una solución eficaz. Esto se debe a que permiten mejorar el rendimiento y reducir el consumo de energía en comparación con las unidades de procesamiento convencionales, como la Unidad Central de Procesamiento (CPU) o la Unidad de Procesamiento Gráfico (GPU) [3]. A pesar de estos avances, surge la necesidad de optimizar aún más el desempeño de los aceleradores, sobre

todo en entornos donde los recursos son limitados o el consumo energético es crítico. En este contexto, la computación aproximada ha ganado relevancia como una técnica que permite aumentar la eficiencia de los aceleradores al sacrificar un margen controlado de precisión en los resultados [4].

En este contexto, el acelerador *General Matrix Multiplication* (OpenGeMM), desarrollado por MICASLab (Apéndice B.1) de la Universidad de Católica de Leuven (KULeuven) (Apéndice A.2) y diseñado mediante el lenguaje de descripción de hardware, Chisel, se presenta como una solución altamente eficiente para la multiplicación de matrices, una de las operaciones fundamentales en muchos algoritmos de inteligencia artificial. OpenGeMM está orientado a mejorar el rendimiento en operaciones de álgebra lineal, ofreciendo ventajas significativas en términos de velocidad de cómputo y consumo energético, factores cruciales en sistemas que deben manejar grandes volúmenes de datos de forma continua y eficiente [5]. Además, OpenGeMM se integra en conjunto con el clúster de procesamiento heterogéneo SNAX (*SNitch Acceleration eXtension*). Snitch es un proyecto de investigación sobre hardware RISC-V de código abierto de la ETH de Zúrich, Apéndice A.3, el cual persigue la máxima eficiencia energética posible. El sistema está diseñado en torno a un núcleo versátil y de número entero reducido, al que llamamos Snitch [6]. La Figura 1.1 muestra el diagrama general del cluster SNAX.

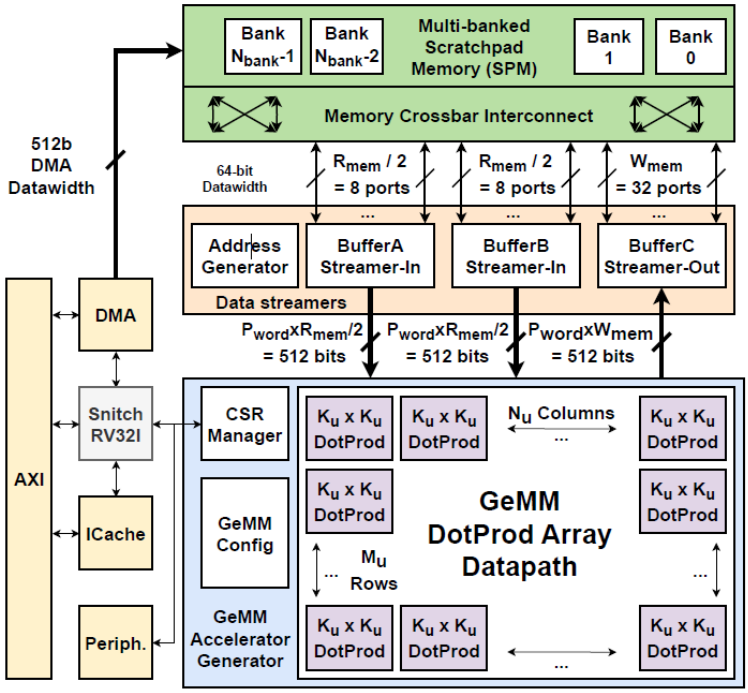


Figura 1.1: Visión general de la plataforma OpenGeMM (tomada de [5]).

Este desarrollo fue financiado y promovido por el proyecto CONVOLVE, que busca el desarrollo en el área de *edge computing* con la colaboración de diversos países de la Unión Europea[1] [7].

[1]Para desarrollar más sobre este tema observe el apéndice A.1

Se espera que el sistema sea altamente parametrizable y adecuado para muchos casos de uso, desde pequeños núcleos sólo de control hasta grandes sistemas multinúcleo para el cálculo de números en el ámbito de la computación de alto rendimiento [6]. La Figura 1.2 muestra una visión simplificada del clúster SNAX.

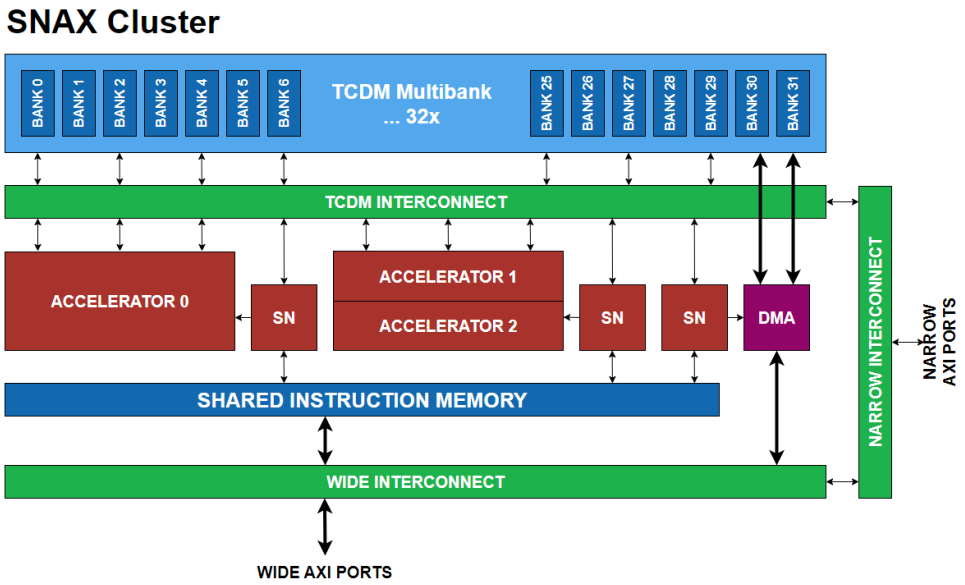


Figura 1.2: Visión general del clúster SNAX (tomada de [6]).

El clúster de la Figura 1.2 está específicamente diseñado para aplicaciones de inteligencia artificial, y combina diversas arquitecturas de procesamiento para maximizar la eficiencia en tareas demandantes de recursos computacionales. SNAX integra diferentes tipos de núcleos de procesamiento y optimiza su funcionamiento para cargas de trabajo específicas, lo que lo convierte en un sistema clave para mejorar el rendimiento en entornos de IA.

Dado este contexto de innovación y avance tecnológico, desde el ECASLab B.2, de la Escuela de Ingeniería Electrónica del Instituto Tecnológico de Costa Rica, propone una solución para mejorar el procesamiento de las operaciones matemáticas necesarias para los algoritmos de ejecución de modelos basados en IA. El estudio se centra en la identificación y puesta en práctica de cuales tipos de aproximaciones son los más adecuadas para operaciones de matrices en sistemas como el descrito recientemente, así como analizar el impacto de estas aproximaciones en función al nivel que se implementen, con respecto a precisión de datos y eficiencia computacional.

1.1. Diagrama general de la solución

El diseño de aproximaciones para aceleradores dedicados a la multiplicación de matrices permite el aumento de la eficiencia de la operación completa. Esto permite el uso específico de estos circuitos para aplicaciones directas que ayuden a la sociedad en función de los algoritmos de inteligencia artificial y su disponibilidad de forma accesible.

El desarrollo de esta investigación permitirá encontrar la mejor metodología de aproximación para esta arquitectura completa, por lo que se describen los tipos de aproximación orientados a multiplicadores de matrices.

El primero es el algoritmo de aproximación de truncamiento, el cual consta de realizar un recorte de bits a ambos datos provenientes y hacer la multiplicación con este menor tamaño. Luego el siguiente algoritmo de aproximación es la cuantificación, el cual se define como un ajuste adaptativo del dato en función de su longitud y peso de los bits menos significativos [8].

Para el caso específico del planteo de la solución, se basa en unir los dos conceptos anteriormente descritos, lo que me permite aprovechar las ventajas que ofrece cada uno. En el capítulo 2 describe específicamente cada una de ellas.

La Figura 1.3 presenta un diagrama de bloques para la solución propuesta desde el ambiente general del multiplicador de matrices. Por lo que se plantea aplicar la aproximación específicamente en esta área, ya que se identificó que en esta unidad es donde mejor se adapta el uso de una aproximación algebraica.

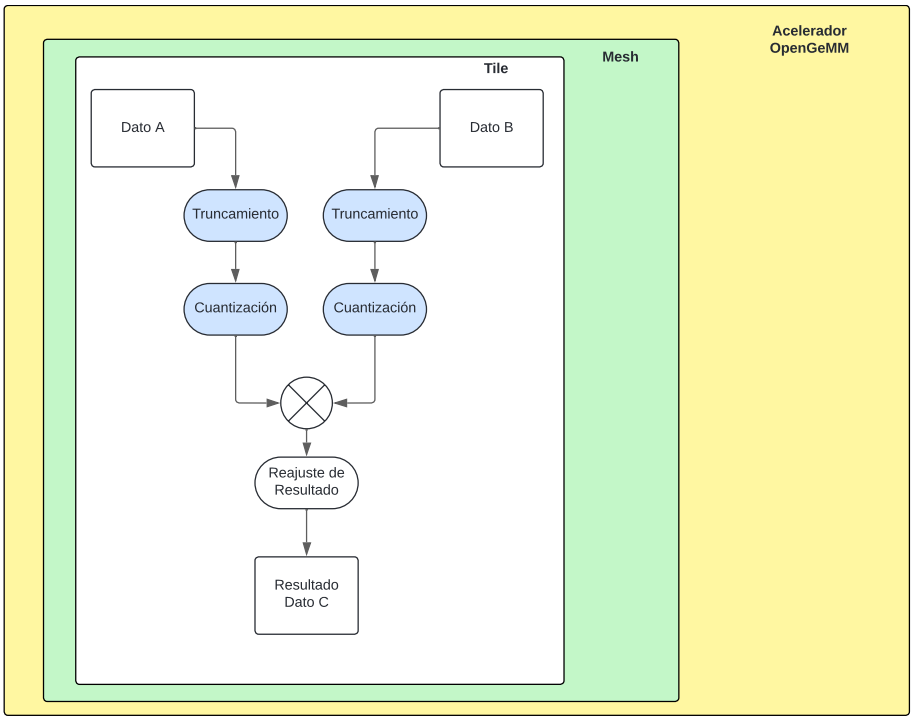


Figura 1.3: Diagrama de solución para aproximaciones en OpenGeMM

El objetivo de la Figura 1.3 es que se pueda observar el concepto de capas superiores que posee la arquitectura interna del acelerador, por lo que dentro del acelerador tenemos el bloque llamado *Mesh* y dentro de el existe el bloque de *Tile*, que tiene n cantidad de ellos en función del tamaño de la matriz de entrada. Además de que se observe que las aproximaciones realizadas están realizadas previamente a la multiplicación de los datos, seguido de esto el algoritmo realiza la suma completa de cada registro n para dar el resultado final de la matriz resultante.

1.2. Objetivos y estructura del documento

Este informe tiene como objetivo desarrollar una aproximación computacional, para el aumento en el rendimiento del procesador con acelerador de multiplicación de matrices. Con el fin de cumplir con este objetivo general del proyecto, es necesario analizar en qué lugar se debe de realizar la aproximación, qué tipo es el más indicado y por último a qué nivel de implementación es posible hacerlo.

Por último es importante obtener los resultados del hardware diseñado, en los aspectos de rendimiento, tamaño del área, consumo energético y error aproximado de los valores, ya que de esta forma es posible comparar y dar una nota con la factibilidad de utilización de la aproximación o el circuito exacto.

Una vez conocidos los objetivos, en el capítulo 2 se presentan los diferentes conceptos necesarios para comprender lo desarrollado como solución de este proyecto.

Capítulo 2

Antecedentes y trabajo relacionado

En capítulo presenta los conceptos necesarios para la correcta comprensión del documento, donde se exponen diversas definiciones y explicación de los conceptos. Además se exponen distintos trabajos relacionados en el tema que desarrollan los esfuerzos anteriormente realizados en el caso específico de multiplicación de matrices aproximada.

2.1. Computación aproximada

La computación aproximada ha surgido como un paradigma de computación transformador destinado a mejorar la eficiencia energética de los sistemas digitales permitiendo una pérdida controlada de precisión en los cálculos. Este enfoque aprovecha la resistencia a errores inherente a muchas aplicaciones, sobre todo en multimedia y análisis de datos, donde la percepción humana puede tolerar imprecisiones [9][10].

Al sacrificar precisión por ahorro de energía, las técnicas de computación aproximada pueden reducir significativamente el consumo de energía, lo que las hace ideales para aplicaciones de bajo consumo, como los dispositivos de internet de las cosas (IoT) [11].

Estudios recientes indican que una parte sustancial de las tareas computacionales, aproximadamente el 83 %, puede funcionar eficazmente con requisitos de precisión relajados [10]. Además, la integración de la computación incremental con métodos aproximados puede optimizar la utilización de recursos y acelerar los tiempos de procesamiento, al mejorar así el rendimiento general del sistema [12]. Esta sinergia representa una vía prometedora para futuras investigaciones y aplicaciones en el ámbito de la computación energética mente eficiente [13].

2.1.1. Cuantización y truncamiento de datos

La cuantización en procesamiento de señales implica la conversión de valores continuos a un conjunto discreto de niveles, que produce un error debido a la diferencia entre el

valor real y el valor aproximado. Este proceso se usa comúnmente en sistemas digitales para representar señales analógicas en formato digital o en computación para realizar operaciones algebraicas aproximadas [14].

Además, se pueden aplicar estrategias de truncamiento y redondeo en la cuantización para representar los valores; el truncamiento descarta bits menos significativos, mientras que el redondeo selecciona el valor más cercano al valor real. Aunque el truncamiento es sencillo, el redondeo suele ser preferido ya que tiende a mantener el error de cuantización con media cero y un comportamiento menos sesgado [15].

Esta selección de técnicas permite un balance entre precisión y complejidad en los sistemas digitales, lo que asegura una representación adecuada de las señales analógicas en diversos niveles de resolución. La Figura 2.1 representa la operación de cuantización en un número binario de ocho bits.

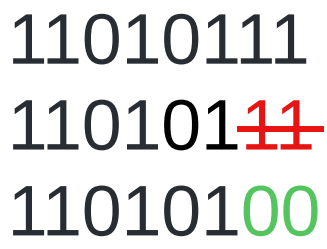


Figura 2.1: Cuantización adaptativa para un valor binario

El truncamiento es un método en el procesamiento de señales y computación digital que consiste en eliminar los bits menos significativos de un valor numérico para reducir su longitud de representación, lo que es útil en sistemas de hardware o software donde la memoria y los recursos computacionales son limitados. A diferencia del redondeo, el truncamiento simplemente descarta los bits sin ajustar el valor al número más cercano, lo que puede llevar a un sesgo sistemático en el error, ya que siempre reduce el valor de la representación numérica en lugar de aproximarlo [14][16].

El truncamiento es, por lo tanto, una técnica efectiva cuando se prioriza la velocidad y simplicidad sobre la precisión, y es comúnmente empleada en sistemas embebidos y en tareas de procesamiento de señales donde el costo computacional debe mantenerse bajo [16]. La Figura 2.2 representa la operación de truncado para un valor binario de ocho bits de longitud.

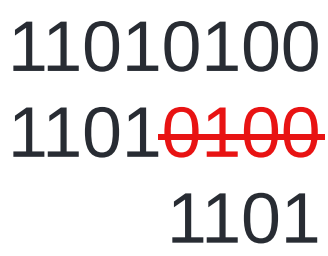


Figura 2.2: Truncamiento para un valor binario

2.1.2. Computación aproximada a nivel de software

La computación aproximada a nivel de software es un paradigma que mejora la eficiencia energética y el rendimiento al permitir imprecisiones controladas en los cálculos. Este enfoque es especialmente beneficioso en el diseño VLSI de bajo consumo, donde las técnicas de cálculo aproximado pueden reducir significativamente el consumo de energía manteniendo niveles aceptables de precisión para aplicaciones tolerantes a errores [17].

El éxito de la implementación requiere un esfuerzo de colaboración entre la pila hardware-software, donde los compiladores y el *middleware* desempeñan un papel crucial en la optimización del código para los métodos de aproximación [18].

Además, las estrategias de conservación de energía a nivel de software, como el compilador ecológico distribuido, pueden mejorar aún más la eficiencia optimizando el uso de recursos durante la transformación del código [19].

Técnicas como la regresión bayesiana para la corrección de errores en convertidores analógico-digitales también ejemplifican innovaciones a nivel de software que mejoran la precisión al tiempo que aprovechan la computación aproximada [20].

2.2. Arquitectura de computadoras heterogéneas

Las arquitecturas de computadoras heterogéneas son cada vez más vitales para abordar las demandas de la computación moderna, particularmente en campos como la inteligencia artificial y el big data. Estas arquitecturas integran diversas unidades de procesamiento, como CPU, GPU y aceleradores especializados, para optimizar el rendimiento y la eficiencia energética, superando las limitaciones planteadas por los sistemas homogéneos tradicionales [21] [22].

La evolución de estos sistemas incluye innovaciones como arquitecturas componibles que permiten la asignación dinámica de recursos y la desagregación de memoria, al mejorar la utilización y flexibilidad del sistema [21]. Persisten desafíos como la complejidad de programación y las cargas de recursos desequilibradas, lo que requiere modelos de programación unificados y estrategias inteligentes de asignación de tareas [22].

Las aplicaciones abarcan sistemas de conducción autónoma, que se benefician del diseño conjunto de hardware/software, hasta sistemas de imagen avanzados que aprovechan los FPGA para el preprocesamiento [23][24]. El futuro de la computación heterogénea promete avances significativos en diversas industrias, impulsando el crecimiento económico y la innovación tecnológica [22] [25].

2.3. Algoritmo de multiplicación de matrices

El algoritmo de multiplicación de matrices es un método fundamental para combinar dos matrices y obtener una nueva matriz resultante. El proceso consiste en calcular cada entrada de la matriz resultado como el producto escalar de una fila de la primera matriz con una columna de la segunda matriz. Esto se formaliza matemáticamente mediante la Ecuación (2.1) [26].

$$C[i][j] = \sum_{k=1}^{n} A[i][k] \times B[k][j] \tag{2.1}$$

donde A es una matriz de dimensión $m \times n\times$, B de dimensión $n \times p$, y C es la matriz resultante de dimensión $m \times p$. El algoritmo tiene una complejidad de tiempo de $\mathcal{O}(m \times n \times p)$, lo que lo convierte en un proceso costoso para matrices de gran tamaño [26].

Para optimizar el cálculo, se han desarrollado otros algoritmos como el algoritmo de Strassen, que reduce la complejidad a $\mathcal{O}(n^{2,81})$ al dividir las matrices en submatrices más pequeñas, aprovechando recursividad y simplificaciones algebraicas [27], [28]. Este algoritmo es clave en campos que requieren grandes cálculos matriciales, como gráficos por computadora, procesamiento de imágenes y redes neuronales [27].

2.3.1. Producto punto

El producto punto o producto escalar es una operación matemática fundamental en álgebra lineal. Es una operación entre dos vectores que devuelve un escalar, calculado como la suma del producto de los componentes correspondientes de los vectores. Dados dos vectores $a = (a_1,a_2,a_3,...,a_n)$ y $b = (b_1,b_2,b_3,...,b_n)$, la Ecuación (2.2) lo define de forma algebraica [29].

$$\mathbf{a} \cdot \mathbf{b} = \sum_{i=1}^{n} a_i b_i = a_1 b_1 + a_2 b_2 + \cdots + a_n b_n \tag{2.2}$$

Este concepto es ampliamente utilizado en geometría, física y ciencias de la computación, donde el producto punto permite medir la proyección de un vector sobre otro, y determinar si son ortogonales entre sí [30].

2.3.2. Desenrollado temporal y espacial

El desenrollado temporal (*temporal unrolling*) es una técnica de optimización que se utiliza principalmente en el contexto de sistemas con paralelismo, como en el diseño de hardware, FPGAs o procesamiento paralelo. En lugar de procesar todas las operaciones de un ciclo en una misma unidad de tiempo, se distribuyen en diferentes ciclos de reloj para maximizar

el uso del hardware en diferentes momentos. Esto mejora la eficiencia al permitir que las tareas que dependen de los resultados de pasos previos continúen en paralelo en el tiempo [31].

En esta técnica, el "desenrollado"no se hace duplicando las instrucciones en espacio (como en el *loop unrolling* clásico), sino permitiendo que el ciclo avance en múltiples unidades de tiempo, distribuyendo el trabajo en diferentes ciclos temporales para aprovechar mejor los recursos disponibles [31]. Por consecuente, el *spatial unrolling* distribuye las operaciones del ciclo en espacio, al ejecutar diferentes iteraciones del ciclo en paralelo en diferentes unidades de hardware. En lugar de procesar una sola iteración en un ciclo, varias iteraciones se ejecutan simultáneamente, aprovechando unidades de procesamiento independientes o paralelismo de datos. Esta técnica se usa comúnmente en procesadores vectoriales, FPGAs y GPU, donde se puede replicar el hardware para ejecutar múltiples operaciones al mismo tiempo [32].

2.4. Área total

En el diseño de circuitos de hardware, el área total es una métrica fundamental, ya que determina el tamaño físico del chip y, en última instancia, su costo de fabricación. Al minimizar el área, permite reducir el costo y mejorar la eficiencia del diseño, especialmente en aplicaciones de dispositivos embebidos y sistemas integrados [33].

El área total de un circuito digital incluye tres componentes principales, el área de lógica (A_{logic}) que se refiere al espacio ocupado por los elementos lógicos, como puertas y multiplexores. Luego, el área de memoria (A_{memory}) que incluye la cantidad de espacio físico asignado a las celdas de memoria, esenciales en sistemas con almacenamiento integrado. Por último el área de interconexión ($A_{interconnect}$) que corresponde al área necesaria para conectar los distintos bloques y asegurar la comunicación entre ellos. Su tamaño aumenta con la complejidad del diseño [34]. La Ecuación (2.3) define el área total en función a los componentes anteriormente descritos.

$$A_{total} = A_{logic} + A_{memory} + A_{interconnect} \tag{2.3}$$

Cada uno de estos términos puede variar significativamente según el tipo de aplicación y los requisitos específicos del sistema. Para el caso de este informe, se tomó en cuenta otros aspectos importantes para el cálculo del área total que se pueden ver en el Capítulo 4.

2.5. Potencia total

La potencia total en un circuito de hardware se refiere a la cantidad de energía consumida por unidad de tiempo. Entender este consumo es crucial en el diseño de circuitos digitales

y sistemas embebidos, especialmente cuando se busca eficiencia energética y reducción de calor [35].

La potencia total se descompone comúnmente en tres componentes principales [36]:

- **Potencia dinámica (P_{dyn})**: La potencia consumida cuando los circuitos están cambiando de estado. Depende de la frecuencia de operación, la capacitancia de carga y el voltaje de operación.
- **Potencia estática o de fuga (P_{static})**: La potencia consumida debido a corrientes de fuga cuando el circuito está en reposo.
- **Potencia corta (P_{short})**: La potencia disipada brevemente cuando ambos transistores en una puerta lógica están conduciendo durante la conmutación.

La fórmula general para la potencia total está relacionada a la Ecuación (2.4).

$$P_{total} = P_{dyn} + P_{static} + P_{short} \tag{2.4}$$

Para calcular específicamente la potencia dinámica se usa la expresión de la Ecuación (2.5):

$$P_{dyn} = \alpha \cdot C_L \cdot V_{DD}^2 \cdot f \tag{2.5}$$

donde α es el factor de actividad, que representa la proporción de tiempo en que el circuito conmutará. C_L es la capacitancia de carga del circuito. V_{DD} es el *voltaje de operación* y f es la frecuencia de conmutación del circuito [37].

Para el caso de las pruebas diseñadas para este informe, el factor de actividad y la capacitancia fue propuesto por la herramienta en modo definido, ya que de esta forma permite que se calcule específicamente para cada caso.

2.6. Ruta crítica

La ruta crítica se refiere a la secuencia de tareas u operaciones que determinan el tiempo mínimo de finalización de un proyecto o proceso computacional, particularmente cuando se trata de datos inciertos o imprecisos. El método de ruta crítica (CPM) se emplea comúnmente para identificar estas rutas en diversas aplicaciones, incluyendo la simulación de circuitos, donde algoritmos como *Critical Path Tracing* (CPT) mejoran la eficiencia al simplificar las simulaciones de fallas [38].

Para el caso de este informe, el flujo de síntesis registra todos estos datos en dos tipos distintos de tiempo, llamados *hold timing* y *setup timing*, siendo el reporte del tiempo de espera y de configuración respectivamente.

2.7. Distancia de error promedio (MED)

La distancia de error medio en la computación aproximada se refiere a la estimación del error al interpolar datos aproximados, particularmente en el contexto de funciones de base radial. Este concepto es crucial ya que aborda las discrepancias entre las funciones fluidas y sus interpolantes, especialmente cuando los datos exhiben una rugosidad significativa [39].

La fórmula en la Ecuación (2.6) se basa en primero obtener la diferencia entre el dato real y el aproximado, luego sumar todos los valores de esta diferencia y por último dividir entre la cantidad de datos que se registran.

$$MED = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i| \tag{2.6}$$

Para este proyecto, se diseñó un ambiente de pruebas completo auto verificable, mediante el diseño por capas con aleatorización controlada para así obtener los valores de error, y todo se realiza dentro de este mismo archivo de *test*. Específicamente se ingresan matrices aleatorias de tamaño definido $8x8x8$, con el objetivo de obtener el archivo *.csv* con los datos de error calculados en el mismo test.

2.8. SNAX cluster

Como trabajo relacionado el clúster *SNitch Acceleration eXtension* (SNAX), es un ambiente que proporciona las necesidades fundamentales para un acelerador, que se compone para este caso de un núcleo gestor (núcleo Snitch) utilizado para controlar el acelerador, una memoria a la que los aceleradores pueden enviar y recibir datos, donde el número de puertos para la memoria de datos estrechamente acoplada (TCDM) conectados al acelerador es parametrizable y un ambiente de interconexión que puede conectarse a otros clusters SNAX o Snitch [40].

Estos componentes básicos permiten a los usuarios crear cómodamente su propia arquitectura heterogénea [40]. Este ambiente o *shell*, se añadió un *Certificate Signing Request (CSR)* para el control general, puertos del acelerador Snitch para controlar el acelerador y puertos TCDM a los puertos de memoria del acelerador, específicos para el OpenGeMM [41].

2.8.1. Micro-arquitectura de SNAX clúster

Esta sección define los conceptos específicos que componen a la micro arquitectura del clúster de SNAX, donde se observa cada dispositivo que lo compone. La Figura 2.3,

muestra los bloques que definen la estructura del ambiente completo, para tomar como referencia en el estudio de las unidades.

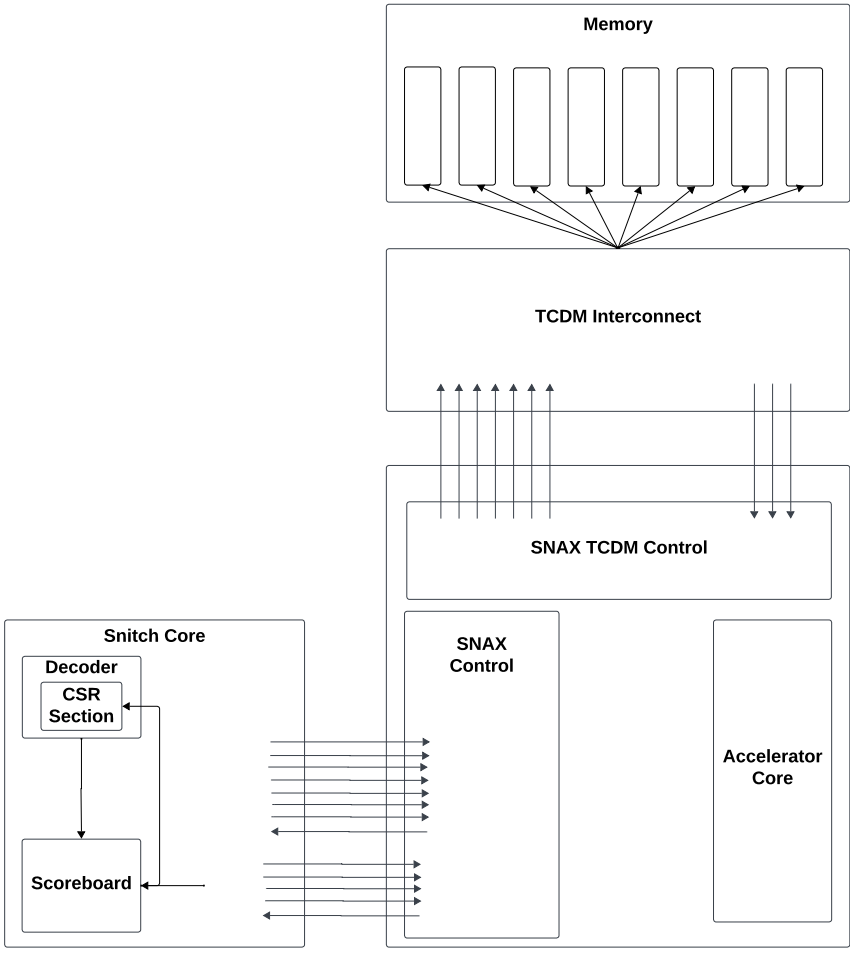


Figura 2.3: Micro arquitectura SNAX [41]

2.9. RISC-V snitch core

El proyecto Snitch es una investigación sobre hardware RISC-V de código abierto de la ETH de Zúrich y la Universidad de Bolonia que persigue la máxima eficiencia energética posible. El sistema está diseñado en torno a un núcleo versátil y de pequeños números enteros, al que llamamos Snitch. Se espera que el sistema sea altamente parametrizables y adecuado para muchos casos de uso, desde pequeños núcleos sólo de control hasta grandes sistemas multi núcleo para el cálculo de números en el ámbito de la computación de alto rendimiento [41].

Snitch es un núcleo RISC-V (RV32I o RV32E) de una sola etapa, una sola emisión y en orden, ajustado para ofrecer simplicidad y una huella de área mínima. Además, es altamente configurable y puede utilizarse en multitud de aplicaciones diferentes. Además contiene el módulo definido como *Score Board* que se encarga que obtener información y generar control interno de las instrucciones y datos de salida o entrada [40].

El núcleo dispone de una interfaz de acelerador opcional que puede utilizarse para controlar y descargar instrucciones RISC-V. La interfaz de carga/almacenamiento es de doble canal, con un canal de petición y otro de respuesta independientes [40].

En la Figura 2.3 relacionada a la arquitectura se puede identificar como un bloque llamado Snitch Core quien tiene embebido el decodificador en conjunto con el registro de desplazamiento, esencial para el manejo de instrucciones del procesador y el *scoreboard* que permite llevar un conteo y verificación de los datos que entran y salen de la unidad.

2.10. Acelerador OpenGeMM

OpenGeMM engloba un acelerador general de multiplicación de matrices (GeMM) parametrizado y codificado en Chisel, un procesador RISC-V ligero y una memoria *tightly coupled multi-banked scratchpad* [5]. La utilización del núcleo GeMM y la eficiencia del sistema se potencian mediante tres mecanismos base: precarga de configuración, precarga de entrada con búfer de salida y acceso programable a la memoria [5].

Los resultados experimentales muestran que OpenGeMM puede lograr una utilización del hardware que oscila del 81,89 % al 99,34 % en diversas cargas de trabajo de redes neuronales convolucionales (CNN) y transformadores de dichas redes [5].

Al realizar un acelerador aproximado el objetivo es que permita reducir los distintos factores de área, potencia, y rendimiento, con un factor de error intrínseco que no afecte en magnitud a los resultados. La Figura 2.4 muestra la estructura de la operación matricial.

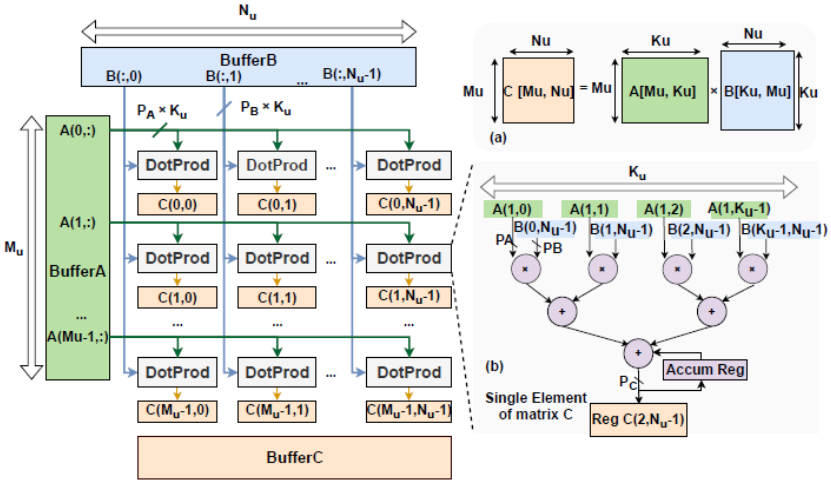


Figura 2.4: Diagrama de producto punto matricial. (a) Multiplicación de matrices que el acelerador GeMM procesa en un ciclo con un resenrollado espacial en 3D. (b) Micro arquitectura de producto punto (tomada de [5])

El acelerador GeMM procesa espacial mente una matriz de colas (*tile*) A' de tamaño (M_u,K_u) y una matriz de colas (*tile*) B' de tamaño (K_u,N_u) para producir una matriz

de colas C' de tamaño (M_u,N_u), como se muestra en la Figura 2.4(a). Para maximizar la reutilización espacial, es importante reutilizar cada elemento de datos obtenido de A' y B' tanto como sea posible. Para garantizarlo, la ruta de datos de la matriz GeMM se conceptualiza como una matriz 3D MAC (Multiplipy-Accumulate), como en la Figura 2.4. La matriz MAC 3D se organiza como una malla (Mesh) del tamaño (M_u,N_u) de K_u unidades de producto vectorial por puntos (DotProd, como se detalla en la en la Figura 2.4(b) para desenrollar espacialmente todas las dimensiones de las matrices A',B' y C' [5].

La matriz MAC 3D está adaptada para que coincida con las 3 *spartial unrolling* anidados para el procesamiento GeMM. En concreto, los vectores de A' y los vectores de la matriz B' se transmiten horizontalmente y verticalmente entre el arreglo *DotProd*, al maximizar la reutilización espacial. Dentro de un *DotProd*, los resultados de la multiplicación K_u se acumulan combinatoriamente para obtener un resultado de C' [5].

OpenGeMM soporta configuraciones en tiempo de diseño del tamaño del arreglo *DotProd* y el tamaño de cada unidad *DotProd*. Esta característica permite una generación flexible de una amplia gama de aceleradores, como unidades de producto punto, unidades de producto punto exterior, aceleradores de multiplicación vector-matriz o aceleradores de multiplicación matriz-matriz. Por lo tanto, se pueden implementar diversos desenrollamientos espaciales optimizados basados en estos parámetros (M_u,K_u,N_u) para adaptarse a los diversos requisitos computacionales de las distintas aplicaciones [5].

2.10.1. Algoritmo de cálculo utilizado en GeMM

El proceso de cálculo consiste en dividir matrices grandes en colas que pueden ejecutarse en un ciclo en el acelerador GeMM. En OpenGeMM este proceso se representa como 6 bucles anidados que pueden clasificarse en desenrollamientos espaciales y temporales.

Los desenrollamientos espaciales (los 3 bucles más internos de la Figura 2.5) representan el paralelismo de cálculo espacial dentro de un único ciclo de reloj [5]. Por otra parte, los desenrollamientos temporales (los 3 bucles más externos de la Figura 2.5) reflejan el orden de procesamiento secuencial de las distintas colas, óptimos para maximizar la reutilización de datos locales y minimizar las paradas mediante un ordenamiento adecuado de los bucles [5].

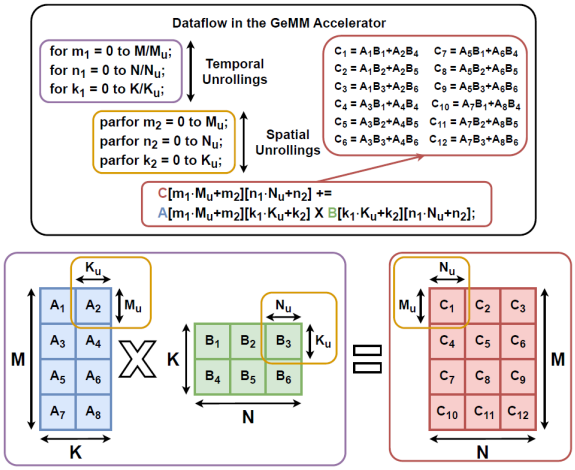


Figura 2.5: Representación de flujo de datos en el generador de hardware para acelerador GeMM (tomada de [5])

2.10.2. Unidad de Control y registro de estados (CSR)

Un registro de desplazamiento de control (*CSR*) es un componente que proporciona un desplazamiento síncrono de datos dentro y fuera de un registro paralelo. El *CSR* puede utilizarse para crear funciones de nivel superior, como en este caso para enviar instrucciones y datos a un acelerador, proveniente de un procesador [42].

El set de instrucciones RISCV (ISA) tiene un estándar para las instrucciones en el registro de desplazamiento de control (CSR). En SNAX, se asume que cada acelerador personalizado tiene sus propios *CSR*. En las instrucciones CSR de RISCV, sólo especificamos la dirección CSR de destino y el valor que queremos escribir en esa dirección. Utilizar CSRs es un enfoque más general y configurable que escribir instrucciones personalizadas. Hay que tener en cuenta que estas instrucciones tratan los CSR a través de entradas y salidas mapeadas al registro [40].

La Figura 2.3 ilustra como el registro de desplazamiento de control está ubicado dentro del bloque del procesador Snitch, esto me permite ubicar la región de trabajo del mismo y ver qué dispositivos tiene conectado entre sí.

2.10.3. Memoria de datos estrechamente acoplada (TCDM)

Los sistemas de memoria de datos estrechamente acoplados (TCDM) se caracterizan por la estrecha integración de las unidades de memoria y procesamiento, lo que facilita el rápido acceso y procesamiento de datos esenciales para aplicaciones de alto rendimiento, como la computación en tiempo real y los sistemas embebidos [43]. Un componente crítico de estos sistemas es la memoria caché compartida, que permite que múltiples núcleos de procesador accedan a la misma estructura de memoria simultáneamente, mejorando el intercambio de datos y la eficiencia de la comunicación [43].

Para el caso en el clúster de SNAX, el OpenGeMM tiene una unidad de control previa que ajusta los datos a ser enviados a la memoria principal, lo que permite una interfaz eficaz en el manejo de los datos de forma bidireccional. La Figura 2.6 muestra un diagrama de su interconexión.

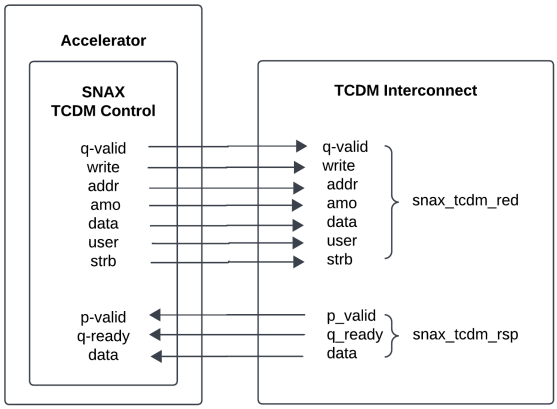


Figura 2.6: Conexión a TCDM

Capítulo 3

Diseño de acelerador OpenGeMM aproximado

Como fue descrito anteriormente, el acelerador OpenGeMM (*General Matrix Multiplication*) es una arquitectura encargada de la multiplicación de matrices de múltiples dimensiones, específicamente para datos enteros provenientes del conjunto de procesadores RISC-V Snitch Core, los cuales se encuentran en una configuración paralela de forma que cumplen con la descripción de una arquitectura heterogénea [5].

OpenGeMM es una pieza de suma importancia en lo que se definió como clúster SNAX, ya que sin el la arquitectura pierde las características para el cual está diseñado, como alto rendimiento y bajo consumo energético para procesos matemáticos complejos.

La Figura 3.1 muestra el flujo estándar de multiplicación de matrices entre los módulos que componen la microarquitectura y el nivel del diseño implementado para aproximar esta operación. El flujo comienza desde la ejecución de la instrucción en el microprocesador. Posteriormente, el resultado de estas instrucciones es enviado al acelerador, que convierte los datos binarios secuenciales en matrices del tamaño especificado. Estas matrices son analizadas y descompuestas para ser enviadas al módulo *Mesh*, encargado de distribuir las filas hacia el módulo *Tile*.

El módulo *Tile* recibe estas filas, las descompone en valores por columna y ejecuta la multiplicación correspondiente al algoritmo de multiplicación de matrices. En este momento, se aplican las aproximaciones previas a la operación.

Los valores resultantes se suman para formar la matriz final, que luego se envía a la interfaz de conexión a memoria para su almacenamiento. Este proceso puede ser bidireccional, y las aproximaciones funcionan en ambos sentidos.

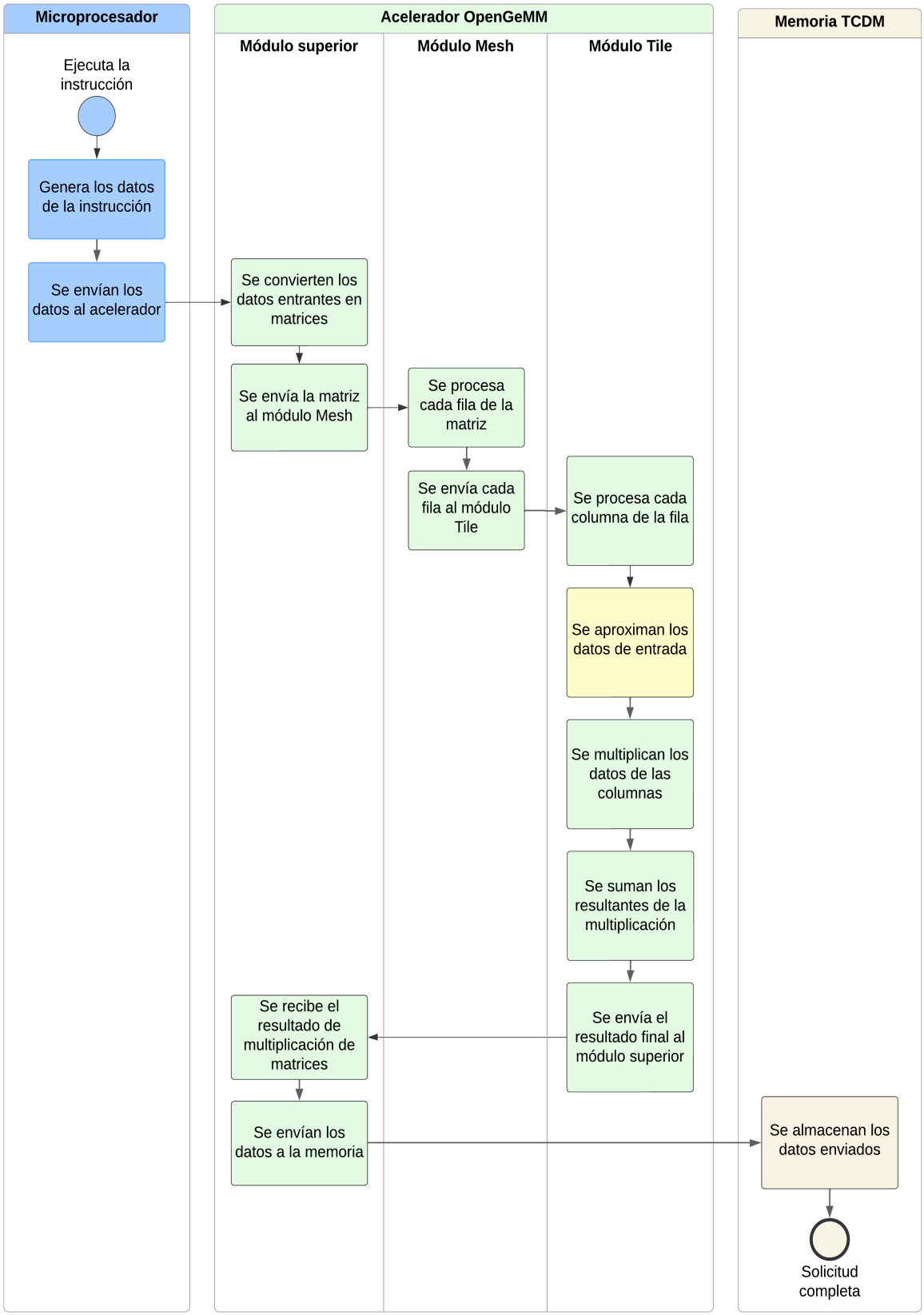


Figura 3.1: Flujo del proceso de la micro arquitectura

Seguidamente se describirán los diseños realizados como solución para tres distintos tipos de aproximaciones. En cada uno de ellos se podrá encontrar una breve definición del código realizado en Chisel, como ejemplo de la implementación realizada.

3.1. Diseño de acelerador OpenGeMM truncado

Al utilizar el conjunto de aproximaciones descrito en el capítulo 2, la idea es obtener una reducción de los módulos que operan los datos, como lo son el multiplicador y los sumadores. Desde el enfoque de diseño de *Very Large Scale Integration* (VLSI), los sumadores son unidades que tienen la característica de su amplio tamaño en comparación con otros operadores, por lo que es importante a tomar en cuenta en el análisis del área equivalente para esta unidad aproximada.

Al aplicar la técnica, el diseño reduce el tiempo de duración de los datos en calcularse, al disminuir la cantidad de bloques de operación que existen para un solo resultado. En concordancia con lo anterior, se espera una reducción en la potencia que consume el dispositivo final. Donde al permitir cierto nivel de error, es posible mejorar distintos factores de rendimiento en la arquitectura.

La Figura 3.2 muestra la reducción de unidades aritméticas por cada *Tile* que se ejecuta para realizar el proceso de multiplicación. Se puede ver que se reducirán dos módulos sumadores y dos módulos de multiplicación al utilizar la aproximación de truncamiento.

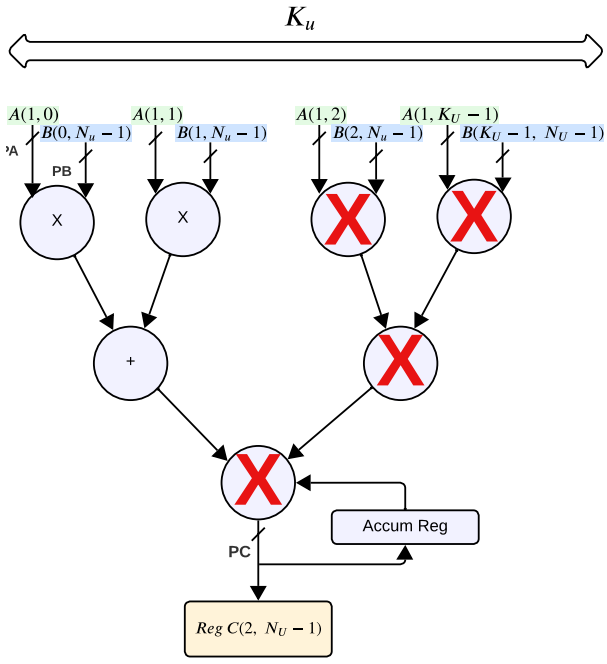


Figura 3.2: Unidad aritmética después de aproximación de truncamiento

Primeramente se decidió realizar las pruebas para profundidades estáticas, por lo que el código en el lenguaje Chisel, el Listado 3.1 muestra el diseño para parámetros definidos en

función del tamaño de los datos de entrada. El caso representado es para truncamiento, y obtiene el valor del ancho de la trama de bits para realizar la operación.

```
1    class Tile(params: GemmParams) extends Module with RequireAsyncReset
     {
2        val desp_a = (io.data_a_i.head.getWidth / 2).U
3        val desp_b = (io.data_b_i.head.getWidth / 2).U
4
5        for (i <- 0 until params.tileSize) {
6            val data_a_approx = (data_a_i_subtracted(i) >> desp_a).
     asUInt
7            val data_b_approx = (data_b_i_subtracted(i) >> desp_b).
     asUInt
8
9            // Multiplicar los valores aproximados
10           val mul_result = data_a_approx * data_b_approx
11           mul_add_result_vec(i) := (mul_result).asSInt
12       }
13   }
14
```

Listing 3.1: Encabezado de módulo Tile, específicamente el diseño para truncamiento de la mitad de bits

Luego, para hacer el diseño de diferentes contextos, se realizó un la modificación de la profundidad de cada aproximación, con el fin de obtener las variaciones, en relación con los factores que se están midiendo y poder obtener comportamientos en los resultados obtenidos.

Al realizar estos diseños adicionales fue posible darle más completitud a los comportamientos observados y poder confirmar las características específicas de cada aproximación.

3.2. Diseño de acelerador OpenGeMM cuantizado

La cuantización es el proceso en el cual los bits menos significativos de la trama se redondean al número menor siguiente, es por ello que logra simplificar con el hecho de agregar valores de cero en la sustitución. La aplicación es un proceso conservador y proporciona reducción menor en las unidades de hardware.

La Figura 3.3 muestra el resultado de cuantización en la unidad de multiplicación del acelerador.

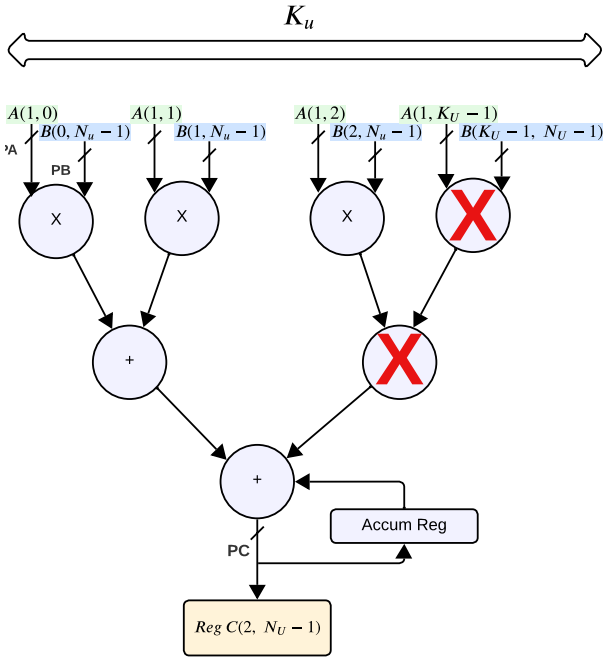


Figura 3.3: Unidad aritmética después de aproximación de cuantización

El código relacionado al Listado 3.2, muestra el diseño de la etapa de multiplicación cuantizada, donde es posible observar las características de redondeo de los bits menos significativos.

```
    class Tile(params: GemmParams) extends Module with RequireAsyncReset
    {
        val desp_a = (io.data_a_i.head.getWidth / 4).U
        val desp_b = (io.data_b_i.head.getWidth / 4).U

        for (i <- 0 until params.tileSize) {
        val data_a_approx = (data_a_i_subtracted(i) >> desp_a).asUInt
            val data_b_approx = (data_b_i_subtracted(i) >> desp_b).
asUInt

            // Multiplicar los valores aproximados
            val mul_result = data_a_approx * data_b_approx

            mul_add_result_vec(i) := (mul_result << (2.U * desp_a)).
asSInt
        }
    }

```

Listing 3.2: Encabezado de módulo Tile, específicamente el diseño de cuantización

Luego, para hacer el diseño de diferentes contextos, se realizó un la modificación de la profundidad de cada aproximación, con el fin de obtener las variaciones, en relación con

los factores que se midieron y poder obtener comportamientos en los resultados obtenidos. La Figura 3.2, muestra el código de este caso específico.

Este diseño se realizó para cada variación de bits, es por ello que se obtuvo los resultados de 1 bit, 2 bits, 3 bits y 4 bits, internamente en el proceso de multiplicación de la cola (*tile*).

3.3. Diseño de acelerador OpenGeMM conjunto

Al utilizar ambas aproximaciones como una conjunta se obtuvo las mejores características de cada caso. El Listado 3.3 representa el código de diseño realizado para la multiplicación matricial. Donde se puede observar como el código relacionado a truncamiento y cuantización forman el complemento, por lo que se decidió realizar los test al diseño e identificar los resultados, para discernir si es una aproximación a tomar en cuenta en este caso.

```
1     class Tile(params: GemmParams) extends Module with RequireAsyncReset
      {
2         val desp_a = (io.data_a_i.head.getWidth / 2).U
3         val desp_b = (io.data_b_i.head.getWidth / 2).U
4
5         for (i <- 0 until params.tileSize) {
6             // Aproximacion de los datos desplazados a la derecha
7             val data_a_approx = (data_a_i_subtracted(i) >> desp_a).
      asUInt
8             val data_b_approx = (data_b_i_subtracted(i) >> desp_b).
      asUInt
9
10            // Multiplicar los valores aproximados
11            val mul_result = data_a_approx * data_b_approx
12            mul_add_result_vec(i) := (mul_result << (2.U * desp_a)).
      asSInt
13        }
14    }
15
```

Listing 3.3: Encabezado de módulo Tile, específicamente el diseño conjunto

3.4. Diseño de prueba aleatoria auto verificable

El procedimiento de test de nuestro diseño, el Listado 3.4, muestra el código del diseño del *testbench* de pruebas aleatorias, realizado para medir el Mean Error Distance (MED) y otros factores.

```
    import chisel3._
    import chiseltest._
    import org.scalatest.flatspec.AnyFlatSpec
    import snax_acc.gemm.{BlockGemm, DefaultConfig}
    import scala.util.Random
    import scala.concurrent.duration._
    import java.io._

    //Desarrollado por Carlos Cerdas Mora
    //Correo: cacerdasm@estudiantec.cr

    class BlockGemmTest extends AnyFlatSpec with ChiselScalatestTester {
      "BlockGemm" should "process input data correctly with 2x2 matrix
    for 1000 random cases" in {
        // Crea o abre el archivo CSV para escribir los resultados
        val writer = new PrintWriter(new File("test_results.csv"))
        // Escribe la cabecera del CSV
        writer.write("TestNumber,ElapsedTime(ms),Error(%),Cycles\n")
        var cycles: BigInt = 0
        // Generar 1000 pruebas con diferentes datos aleatorios
        for (testNum <- 1 to 1000) {
          test(new BlockGemm(DefaultConfig.gemmConfig)) { dut =>
            val numMat: Int = 8
            val random = new Random()
            // Continuar con el codigo de la prueba...
          }
        }
        // Cerrar el archivo CSV despues de escribir todos los
    resultados
        writer.close()
      }
    }

```

Listing 3.4: Encabezado de prueba aleatoria autoverificable

Este *testbench* está conectado directamente al código del módulo superior (*top*), con todas sus señales de control necesarias para activar la máquina de estados que se encuentra a un nivel inferior y por consecuente la estructura de arreglo de colas o *tile array*, que me permite el proceso de multiplicación.

Para el diseño del banco de pruebas aleatorias se siguió el formato de prueba relacionado a la Figura 3.4, donde muestra que trabaja con el bloque general del acelerador, de forma aislada a la arquitectura, al hacer uso de las señales de entrada, control y salida.

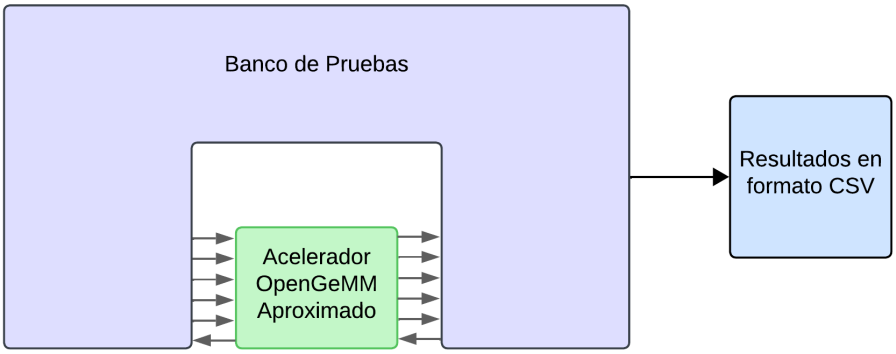


Figura 3.4: Diseño para banco de pruebas

Además este módulo tiene la capacidad de exportar los resultados obtenidos de cada una de las pruebas como un solo archivo en formato *csv*.

3.5. Flujo de ejecución y síntesis de hardware

Para esta sección se buscó explicar los conceptos relacionados al flujo de ejecución del diseño de pruebas auto verificables y aproximaciones en el acelerador de Multiplicación de Matrices. Además de ampliar el flujo de síntesis de hardware en la herramienta Fushion Compiler con la tecnología de la empresa *Taiwan Semiconductor Manufacturing Company Limited* (TSMC) $65nm$.

La Figura 3.5, muestra un diagrama de alto nivel del proceso realizado para obtener los valores finales, relacionados a latencia, porcentaje de error, área y potencia de consumo del acelerador con y sin aproximaciones. Además es posible analizar que los resultados relacionados a error y latencia de la unidad fue posible obtenerlo con el diseño del Test auto verificable específico para el acelerador.

Por otra parte, los resultados de área y potencia se obtuvieron por los reportes generados en el algoritmo de síntesis de la herramienta Fushion Compiler, al ser específico en la sección de diseño de *floor plan*.

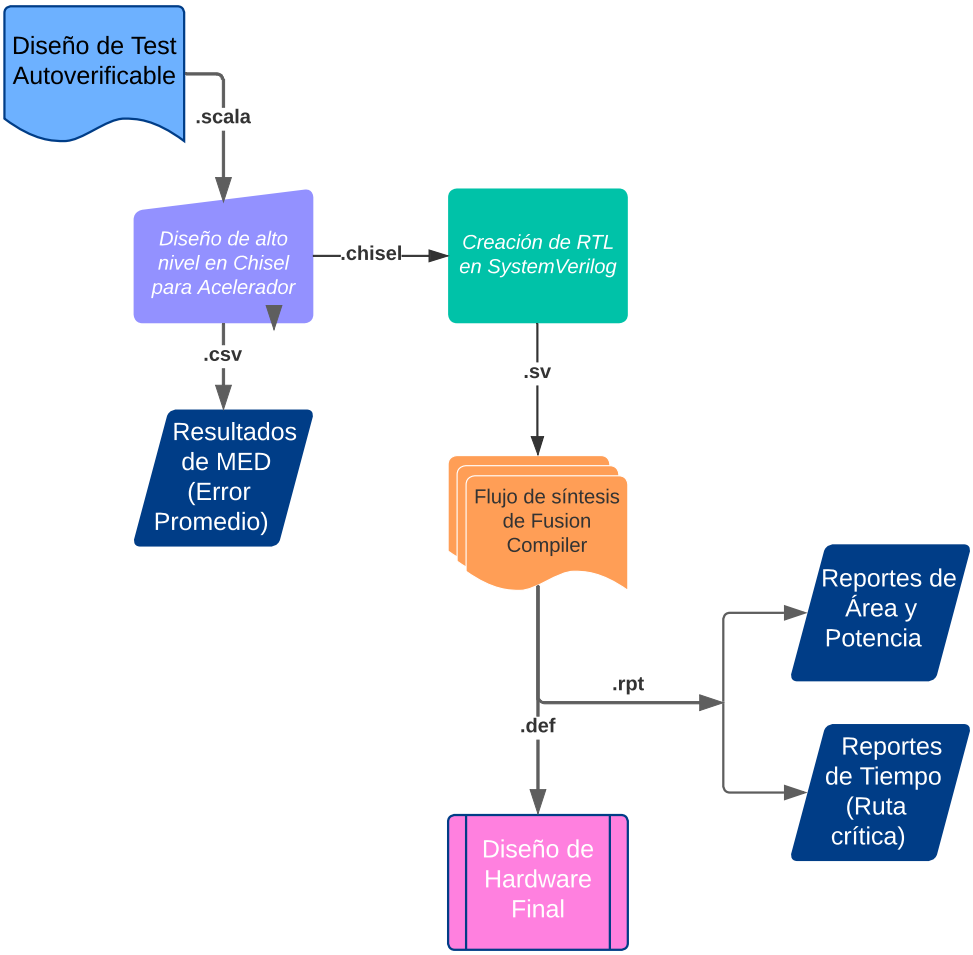


Figura 3.5: Flujo general para ejecución y síntesis

Capítulo 4

Resultados obtenidos

En esta sección se presentan los resultados obtenidos al completar el flujo de síntesis del software Fushion Compiler, con la tecnología desarrollada por TSMC de $65nm$, ya que es una de las herramientas disponibles por la escuela en conjunto con el DCILab B.3, quienes amablemente brindaron acceso para realizar el flujo de compilación y los recursos computacionales.

4.1. Parámetros de configuración generales

Para todos los casos de síntesis se utilizó los mismos parámetros de configuración de la herramienta de Fusion Compiler, esto con el objetivo de que los resultados finales tengan consistencia, ya que se busca realizar el contraste entre cada aproximación implementada.

La Tabla 4.1, indica los parámetros generales que se utilizó para realizar la síntesis lógica.

Tabla 4.1: Parámetros de configuración general

Parámetro	**Valor**
Frecuencia de reloj	200 MHz
VDD	0,625 V
VCC	0,625 V
Ancho de celdas estándar	0,42 nm
Puerto de reloj asignado	clock

4.2. Acelerador OpenGeMM exacto

El acelerador OpenGeMM en la investigación, es el personaje principal, ya que se tomó como referencia para punto de comparación, por lo que en la sección de resultados obtenidos adquiere relevancia para realizar las conclusiones y además lograr definir el avance

del proyecto.

Originalmente el acelerador OpenGeMM está desarrollado en la tecnología de TSMC de $16nm$, que se encuentra a la vanguardia de las nuevas micro arquitecturas actuales. El acceso a este tipo de programas es complejo y restringido, por lo que se resintetizó, tal y como se brindó por MICAS Lab, en la opción disponible en la Escuela de Ingeniería Electrónica que se discutió en la sección 4.1.

Los beneficios brindados por el uso del acelerador permitió que se descubrieran distintas limitaciones en el programa que mejoró el proceso que se lleva a cabo en el DCILab. Además esto ayudó en que se pudiera sintetizar los nuevos diseños de aproximaciones bajo las mismas consideraciones técnicas.

Para cada uno de los respectivos diseños se registraron los parámetros de **Área, Potencia, Ruta crítica y Error promedio**, definidas como las variables principales para todos los casos. Específicamente para el Acelerador Exacto no se realizó estudio de error ya que no aplican las condiciones al ser una operación algebráica definida.

La Tabla 4.2, resume los resultados de las cuatro variables ya que representan un punto de comparación para analizar con las demás soluciones.

Tabla 4.2: Resultados para OpenGeMM exacto

Factor	Valor
Área total	0,553 mm^2
Potencia total	191,05 mW
Ruta crítica	7,532 μs
Mean Error Distance	0

4.3. Acelerador con aproximación de truncamiento

En esta sección se diseñó la aproximación de truncamiento dentro del acelerador, justo en la operación de multiplicación, con el objetivo de obtener los resultados y analizar las variaciones que se producen. Para hacer un estudio más completo de las implicaciones que tiene cada ajuste de profundidad de los bits, se decidió realizar un recorrido desde el truncamiento agresivo hasta el otro extremo conservador.

Se considera que una aproximación es profunda cuando se supera la mitad de los bits que representan un dato, ya que cada vez se pierde más precisión [8].

La Tabla 4.3, contiene los resultados por cada tipo de análisis de profundidad, de forma que es posible observar las variables en función al nivel de aproximación.

Tabla 4.3: Resultados para OpenGeMM con 4 niveles de truncamiento

Factor	Profundidad			
	1 bit	**2 bits**	**3 bits**	**4 bits**
Área total (mm^2)	0,517	0,481	0,436	0,410
Potencia total (mW)	175,29	165,05	143,76	128,00
Ruta crítica (μs)	6,753	6,752	6,559	6,494
Mean Error Distance	95.165,96	115.803,89	137.487,74	144.965,65

4.4. Acelerador con aproximación de cuantización

Para esta sección se diseñó la aproximación de cuantización dentro del acelerador, justo en la operación de multiplicación, con el objetivo de obtener los resultados y analizar las variaciones que se producen. Para hacer un estudio más completo de las implicaciones que tiene cada ajuste de profundidad de los bits, se decidió por realizar un recorrido desde la cuantización menos invasiva hasta la cuantización más p rofunda.

La Tabla 4.4, contiene los resultados por cada tipo de análisis de profundidad, de forma que es posible observar las variables en función al nivel de aproximación.

Tabla 4.4: Resultados para OpenGeMM con 4 niveles de cuantización

Factor	Profundidad			
	1 bit	**2 bits**	**3 bits**	**4 bits**
Área total (mm^2)	0,548	0,542	0,537	0,532
Potencia total (mW)	177,78	164,525	151,262	138,00
Ruta crítica (μs)	7,275	7,018	6,761	6,505
Mean Error Distance	992,95	3.030,68	7.090,85	15.416,92

4.5. Acelerador con aproximación conjunta

En esta sección, de diseño en conjunto se refiere a utilizar las dos aproximaciones anteriormente descritas, para obtener resultados en un balance con todas las variables a medir. Al igual que los anteriores diseños, se realizó el diseño de ajuste justo con el multiplicador de cada valor binario.

La Tabla 4.5, contiene los resultados del recorrido por los distintos niveles de profundidad en los bits de la aproximación, para matrices de entrada de 8 bits.

Tabla 4.5: Resultados para OpenGeMM con 4 niveles de aproximación conjunta

Factor	**Profundidad**			
	1 bit	**2 bits**	**3 bits**	**4 bits**
Área total (mm^2)	0,546	0,544	0,516	0,462
Potencia total (mW)	177,79	164,524	149,98	131,74
Ruta crítica (μs)	7,273	7,012	6,839	6,504
Mean Error Distance	1.096,53	3.174,86	9.757,29	14.553,34

Capítulo 5

Análisis de resultados

Se realizó el análisis de los datos obtenidos en el capítulo 4 y en esta sección se cumple uno de los objetivos específicos del proyecto, el cual consiste en contrastar las variables medidas y determinar el mejor diseño en función de los datos del acelerador exacto.

También se trabajaron distintos análisis específicos de los resultados obtenidos, con el objetivo de justificar el comportamiento en función de la teoría investigada.

5.1. Resultados de acelerador OpenGeMM exacto

Para el caso base del acelerador exacto se obtuvo resultados de área, potencia, ruta crítica y Mean Error Distance en función a lo que se esperaba. Según el *paper* [5], relacionado al diseño base de la unidad de multiplicación, se encuentran similitudes en algunos aspectos, pero no es posible compararlo de forma exacta, ya que las herramientas empleadas para los test y síntesis no son las mismas a las desarrolladas.

Esto se debe a que no se tiene acceso a este tipo de herramientas de alta vanguardia, pero se logró el diseño lo que permitió el uso de los test de herramientas para obtener un resultado similar.

Los factores de área total, potencia total y ruta crítica de la tabla 4.2, permitieron observar los recursos que este necesitará, además de que sienta la base en la mejora de rendimiento general del acelerador.

5.2. Análisis de resultados para acelerador OpenGeMM con distintos diseños de aproximaciones

En esta sección se creó los gráficos relacionados a los datos obtenidos del capítulo 4, con el objetivo de tener una manera visual más simple para analizar los datos obtenidos en

función del barrido de profundidad en bits para cada método de aproximación y poder concluir y determinar la mejor opción en términos de las variables analizadas.

5.2.1. Análisis de área para distintos diseños de aproximaciones

La Fig. 5.1, presenta los resultados relacionados al área, agrupados según el método de aproximación implementado y los colores definen la profundidad de bits asignados en la aproximación.

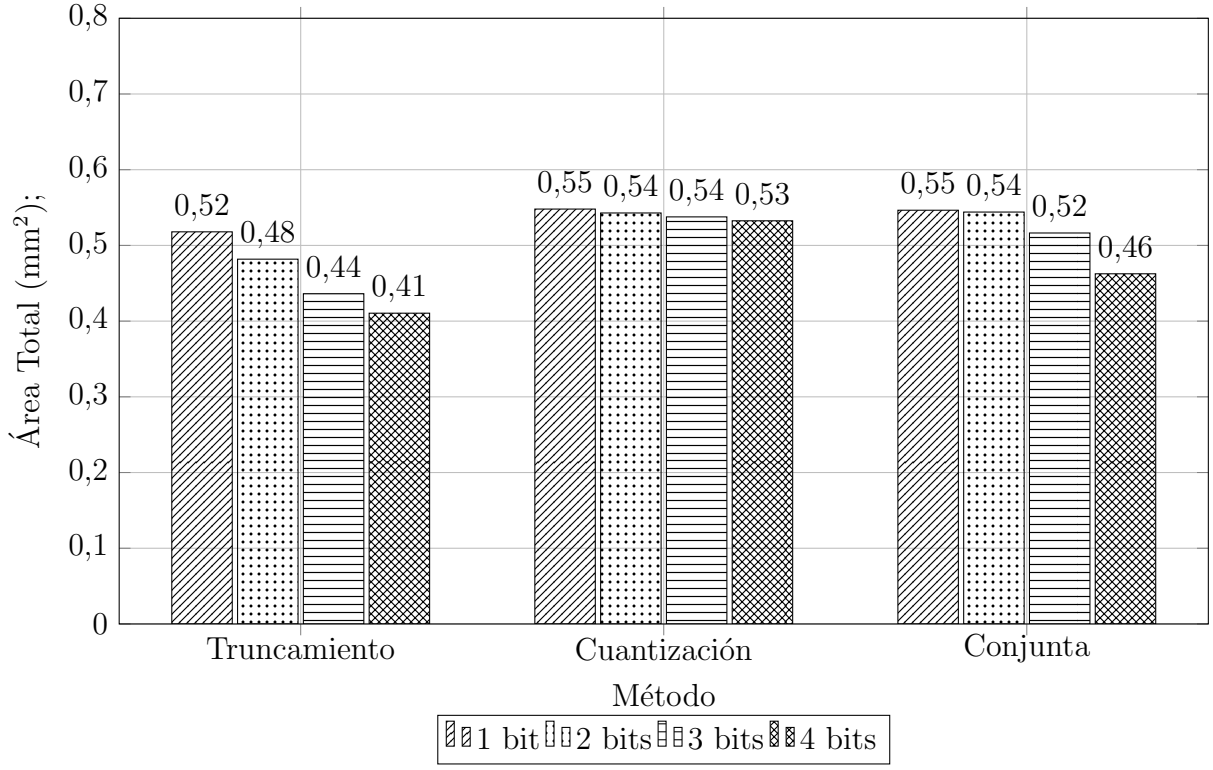


Figura 5.1: Área Total por Método y Nivel de Bits

Con Figura 5.1 se analizó el comportamiento de variar la profundidad de bits en cada dato y la forma en la que reduce el área según la aproximación. Como se pudo comprobar, la aproximación de truncamiento fue la que dio menores valores, esto ya que el método reduce la cantidad de bits de cada dato de forma abrupta. Y como era de esperarse el método de cuantización presentó una menor disminución de los valores de área debido a sus características.

Es por ello que se diseñó la aproximación conjunta de forma adicional, donde los datos de área son similares a cuantización para los primeros bits de reducción, pero para los mayores, presentó una mayor disminución y generó un balance intermedio en este diseño.

En el análisis se encontró adicionalmente algunos valores de área que no cambiaron según la cantidad de bits reducidos y esto se debe a que el compilador de síntesis de alto nivel, necesita ciertos módulos para ejecutar la operación específica. Vemos el comportamiento

que se encontró en truncamiento y es relacionado al mismo motivo, de forma inversa, ya que este diseño sí es más profundo en la reducción de bits.

Si se considera solo la variable de área el mejor caso es para la aproximación conjunta de 4 bits de profundidad, el cual se encuentra en un punto intermedio entre los dos resultados obtenidos, así mismo comparten las características de cada opción.

5.2.2. Análisis de potencia para distintos diseños de aproximaciones

La Fig. 5.2, representa los datos de potencia total para cada tipo de aproximación en función a la profundidad de bits en cada una de ellas.

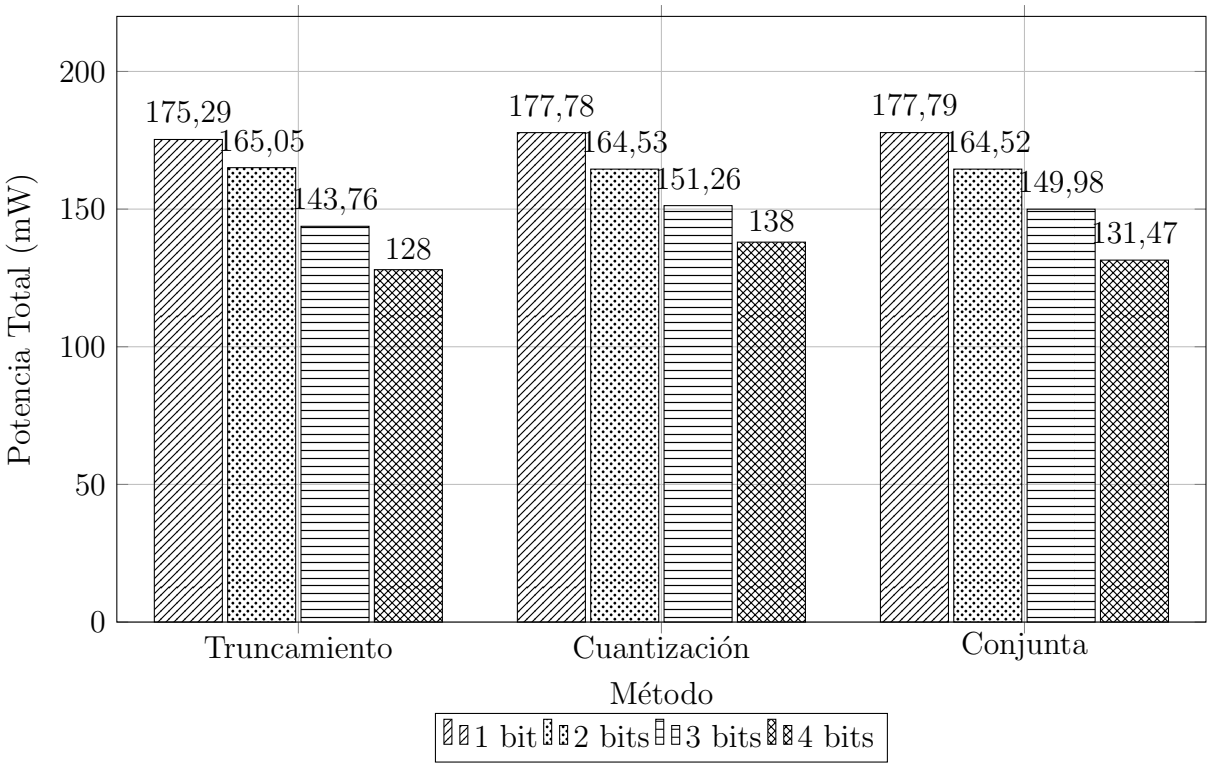


Figura 5.2: Potencia Total por Método y Nivel de Bits

Al analizar los resultados de potencia de la Figura 5.2, estos se encuentran totalmente relacionados con los datos obtenidos de área, además este se calcula en función de los factores adicionales que le brindan exactitud. Pero en general se observó el mismo comportamiento del área, un cambio significativo para truncamiento y la profundidad de 4 bits, el mejor resultado en magnitud.

Por consiguiente, se obtuvo resultados de cuantización que se encuentran en el otro extremo, equivalen a la mayoría con una relación de comportamiento para cada caso de profundidad. La tesis se comprueba que al realizar una aproximación conjunta se obtuvo un resultado en balance a las dos primeras opciones.

Entonces, para la variable de potencia el mejor caso es de aproximación conjunta para 4 bits, ya que representa el dato intermedio entre los dos extremos obtenidos.

5.2.3. Análisis de ruta crítica para distintos diseños de aproximaciones

La Fig. 5.3, presenta los resultados del análisis múltiple de bits en función de la aproximación para la variable de ruta crítica. Esta variable permitió analizar los aspectos relacionados al tiempo para cada aproximación, desde el análisis de la ruta crítica en el resultado de sintetizar cada aproximación.

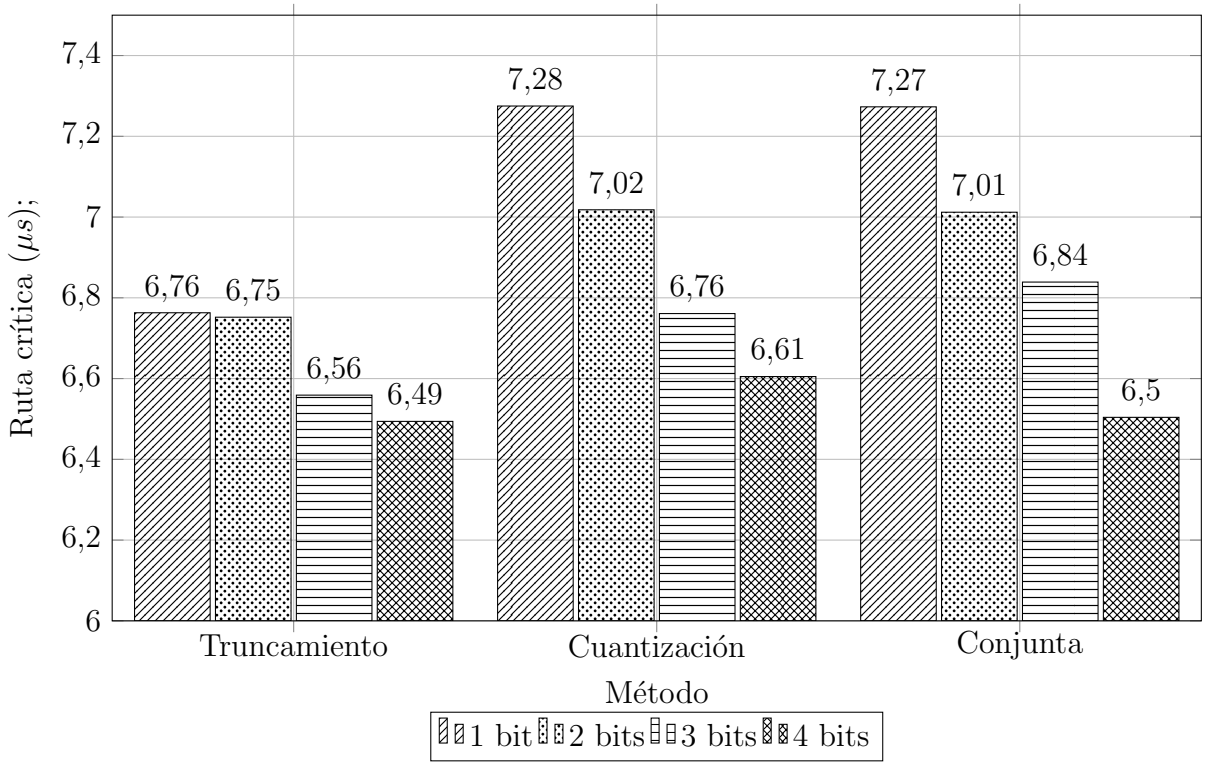


Figura 5.3: Ruta crítica por Método y Nivel de Bits

Los resultados obtenidos con la herramienta de síntesis para la ruta crítica en cada diseño, dieron como resultado que el truncamiento fue el que presentó los valores más bajos y la cuantización lo contrario. Donde en este caso el diseño de la aproximación conjunta permitió obtener un balance entre estos dos.

Además, se analizó que para algunos casos de profundidad la ruta crítica no varió, como lo es para truncamiento de 1 bit y 2 bit respectivamente, lo que es relacionado a que la ruta crítica en este caso no fue modificada y por ende no se observa variación.

Otro caso, con el mismo comportamiento, pero ahora entre diseños fue para cuantización y conjunta ambos en la profundidad de 2 bits. Lo que se le atribuye a la misma razón de la ruta crítica, sin modificarse por el algoritmo de síntesis.

De la misma forma si analizamos los resultados de forma aislada de ruta crítica el que presenta mejor desempeño es la aproximación conjunta de 4 bits, ya que es un resultado intermedio en comparación con las otras dos disponibles.

5.2.4. Análisis de error promedio para distintos diseños de aproximaciones

La Fig. 5.4, muestra los resultados obtenidos para la variable de Mean Error Distance en cada método de aproximación y cada profundidad de bits específica.

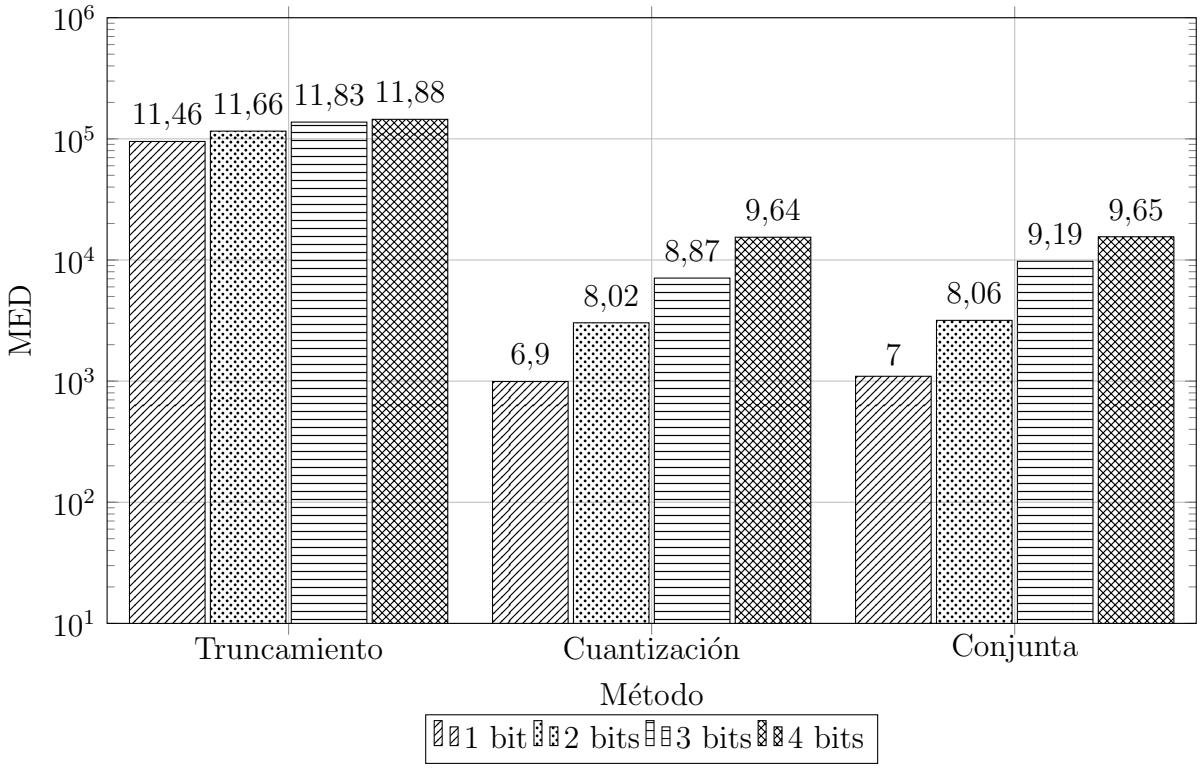


Figura 5.4: Comparación de MED para diferentes métodos y niveles de bits

En la Figura 5.4 se observa el truncamiento, donde en este caso presentó los valores máximos de error, los cuales son tan distantes que se representó mediante una escala logarítmica para hacer posible un análisis correcto de los demás diseños. Consecuentemente, al aumentar la cantidad de bits de profundidad el error aumentó, siento un factor de suma importancia a tomar en consideración en el uso de este diseño.

Para el caso de cuantización se obtuvo un factor de MED menor, lo cual es ideal para una buena aproximación, pero con el diseño conjunto, es posible obtener un error cercano con una correlación entre variables más óptima en el proceso de selección de aproximación.

El resultado general relacionado al MED, donde aumenta en función de los bits de profundidad fue el esperado, ya que cada vez, se reduce la precisión entre cada dato de entrada

y el error se expande de manera exponencial entre los operandos. Al dar como indicador que una taza menor de error es lo buscado en este caso.

Por la naturaleza del proceso algebráico los valores de MED se visualizan como un valor de alta magnitud, pero es importante recordar la manera en la que esta variable se calcula y el entorno en el que se encuentra, donde cada dato es un vector de 8 bits de extensión.

En función de los resultados de la variable de error, el mejor diseño se le atribuye nuevamente a la aproximación conjunta, ya que el truncamiento sobre pasa los límites de funcionamiento básico en error y la cuantización me permite un error que se puede despreciar, por lo que al realizar la conjunción de ambas características se obtuvo una solución más balanceada.

Se analizó los resultados obtenidos de error y para poder clasificarlo entre un valor bajo o alto se generó la referencia teórica de los límites de error. Para el valor mínimo se define a cero y para el dato máximo se realizó el análisis de cuando se multiplican dos matrices de tamaño $8 \times 8 \times 8$ bits, cada elemento del resultado se calcula al multiplicar los elementos correspondientes de las matrices y al sumar los resultados. Para cada elemento en la matriz resultante, se tiene la Ecuación 5.1.

$$\mathrm{R}_{ij} = \sum_{k=1}^{8} A_{ik} \times B_{kj} \tag{5.1}$$

Donde A_{ik} y B_{kj} son los elementos de las matrices de entrada.

El valor máximo que puede tomar un elemento de cada matriz es 255, ya que estamos trabajando con matrices de 8 bits. Por lo tanto, el valor máximo para el producto de dos elementos de 8 bits se representa en la Ecuación 5.2

$$255 \times 255 = 65{,}025 \tag{5.2}$$

Como cada elemento de la matriz resultante es la suma de 8 productos, el valor máximo de un elemento en la matriz resultante está relacionado a la Ecuación 5.3

$$65{,}025 \times 8 = 520{,}200 \tag{5.3}$$

Esto significa que, sin tener en cuenta el manejo de desbordamientos o límites de tipo de dato, el valor máximo para cualquier elemento en la matriz resultante es 520,200.

Además, al analizar los resultados de error obtenidos, se obtuvo el porcentaje de error según la Ecuación 5.4 y se generó la Tabla 5.1 donde se registró el valor para cada aproximación.

$$\%\ \text{Error} = \left(\frac{\text{Valor actual}}{\text{Valor máximo}} \right) \times 100 \tag{5.4}$$

Tabla 5.1: Resultados de porcentaje de error para OpenGeMM con diferentes niveles de aproximación

Aproximación	Factor	Profundidad			
		1 bit	**2 bits**	**3 bits**	**4 bits**
Truncamiento	MED	95,165.96	115,803.89	137,487.74	144,965.65
	% Error	18.29	22.26	26.43	27.87
Cuantización	MED	992.95	3,030.68	7,090.85	15,416.92
	% Error	0.19	0.58	1.36	2.96
Conjunta	MED	1,096.53	3,174.86	9,757.29	14,553.34
	% Error	0.21	0.61	1.88	2.80

Para lograr clasificar estos valores de porcentaje de error obtenidos, se generó la distribución donde

- Bajo: 0 % - 33 % del valor máximo
- Medio: 33 % - 66 % del valor máximo
- Alto: 66 % - 100 % del valor máximo

Al plantear una aproximación de máximo 4 bits, se obtuvo en la mayoría de error una clasificación de error bajo, con algunos valores cercanos a error medio, como lo es el caso de truncamiento. Por lo que para aproximaciones superiores a 4 bits el error estará entre rangos medio y alto.

5.3. Compendio de resultados

En esta sección se realizó la síntesis del análisis previo, pero en conjunto con todas las variables medidas, de forma que resultó en la selección del mejor diseño de aproximación para el acelerador OpenGeMM.

5.3.1. Selección del mejor diseño de aproximación

Al analizar el resultado por separado en cada caso se obtuvo que la aproximación ideal es el diseño realizado en conjunto, donde este mezcla los beneficios que se obtienen del truncamiento y la cuantización, y se logra un balance entre área y potencia total, ruta crítica y Mean Error Distance.

Se realizó el análisis en el cual de las cuatro opciones disponibles para aproximación conjunta es la ideal, donde se tuvo en cuenta el mejor caso para cada variable. Para la opción de área, potencia y ruta crítica, la profundidad de 4 bits fue la que tuvo un área

menor, pero en el valor del *Mean Error Distance* obtuvo máximo de esta variable, lo cual representa la razón de perdida en función de la ganancia en los demás aspectos.

En conclusión, el diseño de aproximación conjunta de 4 bits de profundidad fue la que presentó el mejor comportamiento en las variables estudiadas.

Consecuentemente es importante analizar, los resultados del mejor diseño de aproximación seleccionado, en función con los resultados del acelerador exacto. La Tabla 5.2 muestra la comparación en conjunto de las variables investigadas para el diseño de acelerador exacto y el aproximado conjunto con profundidad de 4 bits.

Como se analizó, la reducción en las variables de área, potencia total y ruta crítica, permite despreciar el factor de MID. Por lo que es posible utilizar este tipo de micro arquitecturas en aplicaciones donde los resultados no son comprometidos por el factor de error y se quiera obtener un diseño más eficiente en aspectos de energía, tamaño y rendimiento.

Factor	**Aproximación**	**Exacto**	**Diferencia**	%
Área total (mm^2)	0,462514	0,553170	0,090	16,38
Potencia total (mW)	131,74	191,05	59,313	31,04
Ruta crítica (μs)	6,504	7,532	1,02	13,64

Tabla 5.2: Resultados de comparación entre OpenGeMM aproximado y exacto

Se redujo aproximadamente 90 μm^2 de área total, 59,3 mW de potencia total, y se aumentó 1,02 μs de rendimiento en la ruta crítica. Esta ganancia viene ligada a un error de Mean Error Distance, lo que comprobó que el factor de pérdidas que se obtiene compensa la ganancia obtenida en las demás variables.

El nivel de error es posible lograr despreciarlo para aplicaciones que no sea necesaria la exactitud en el resultado, por ejemplo, la creación de imágenes, audio y video, o para asistentes de voz, o chats artificiales. En este caso se ignora para el análisis ya que la comparación es contra un modelo exacto de valor cero.

Vemos que esta aproximación permitió una disminución del 31 % de la potencia total del acelerador exacto y el aproximado. De la misma forma para área total y ruta crítica con un 16 % y 13 % respectivamente y un error bajo de 2.80 %.

Capítulo 6

Conclusiones y trabajo futuro

En esta sección se detallan las conclusiones relacionadas a los diseños realizados y el análisis de los resultados obtenidos. Seguido de la sección de trabajo futuro, en donde se amplían puntos de vista acordes con el tema, así como ideas nuevas para realizar en nuevas investigaciones.

6.1. Conclusiones

El truncamiento en multiplicación ofrece mejoras significativas en el rendimiento de los sistemas, ya que permite reducir la complejidad computacional y acelerar las operaciones. Sin embargo, esta mejora en la eficiencia viene a costa de un aumento notable en los errores, lo que hace adecuado solo en aplicaciones que no sea esencial un alto grado de precisión, como en ciertos algoritmos de aprendizaje automático o procesamiento de señales, donde la tolerancia al error es más alta.

Por otro lado, la combinación de truncamiento y cuantización proporciona un balance más óptimo entre rendimiento y precisión. Esta técnica híbrida no solo mejora la eficiencia, sino que también mitiga los errores generados por el truncamiento al hacer ajustes adaptativos a los datos, lo que resulta en una mayor calidad en los resultados sin una gran penalización en el rendimiento.

En particular, la aproximación conjunta con 4 bits ha mostrado ser la mejor opción entre todas las configuraciones evaluadas. Esta combinación maximiza la eficiencia computacional y mantiene un nivel de error aceptable, lo que la convierte en una opción ideal para diversas aplicaciones que requieren un buen equilibrio entre rendimiento y precisión, como en sistemas embebidos o dispositivos de bajo consumo.

Finalmente, la técnica híbrida de truncamiento y cuantización es especialmente útil en campos como el procesamiento de señales y redes neuronales, tanto la eficiencia energética como la tolerancia al error son factores clave para el éxito de las aplicaciones, especialmente cuando se trabaja con hardware limitado.

6.2. Trabajo futuro

En trabajos futuros, se podría explorar la modificación de la arquitectura del acelerador para optimizar aún más el rendimiento y reducir el consumo energético en la multiplicación de matrices. Este enfoque puede incluir la introducción de unidades específicas para manejar operaciones con precisión ajustada, y adaptar el acelerador a las necesidades de la aplicación. Otro aspecto a considerar es el diseño de bajo nivel del acelerador, con enfoque en optimizar los circuitos para aprovechar al máximo las técnicas de aproximación. Un diseño detallado de esta naturaleza permitiría un control más preciso de las operaciones aritméticas, al ajustar cada etapa de la multiplicación y disminuir la acumulación de errores.

Se podrían investigar otras técnicas de aproximación, tales como el uso de filtros adaptativos o la reducción de precisión selectiva en partes específicas del cálculo, al permitir un ajuste más preciso del error en función del contexto y del tipo de datos procesados.

Incorporación de algoritmos de reconstrucción de pérdidas sería una mejora interesante para el sistema de aceleración. Técnicas de redondeo o recuperación de datos perdidos podrían aplicarse para corregir los errores introducidos por la aproximación, que mejoran la precisión general del cálculo sin aumentar significativamente la carga computacional. Además, se podría implementar una asignación dinámica de la profundidad de bits en función de la carga del acelerador.

Bibliografía

[1] M. S. Moreira, D. de Tarso Da Silva y T. C. Pimenta, «1. The Influence of Artificial Intelligence in Society,» 2023. DOI: 10.1109/icm60448.2023.10378900.

[2] N. Manoj y K. T. Liji, «3. Artifical Intellegnce in Business and Society,» *2*, 2024. DOI: 10.46632/rmc/5/2/16.

[3] N. Tukanov, R. Srinivasaraghavan, J. E. Moreira y T. M. Low, «Modeling Matrix Engines for Portability and Performance,» 2022, págs. 1173-1183. DOI: 10.1109/ipdps53621.2022.00117.

[4] J. R. Cary, D. Abell, G. I. Bell et al., «Select Advances in Computational Accelerator Physics,» *IEEE Transactions on Nuclear Science*, vol. 63, n.º 2, págs. 823-841, 2016. DOI: 10.1109/TNS.2015.2500686.

[5] X. Yi, R. Antonio, J. Dumoulin et al., «OpenGeMM: A Highly-Efficient, Customizable GeMM Accelerator with Lightweight RISC-V Control and Tight Memory Coupling,» *ASP-DAC*, 2024.

[6] E. P. Iniciative. dirección: https://www.european-processor-initiative.eu/accelerator/.

[7] CONVOLVE, *Seamless design of Smart Edge Processors*. dirección: https://convolve.eu/.

[8] S. Bosio Ménard, *Approximate Computing Techniques: From Component - to Application-Level*. Springer, 2022.

[9] J. Han y M. Orshansky, «An emerging paradigm for energy-efficient design,» 2013. DOI: 10.1109/ETS.2013.6569370.

[10] K. Roy y A. Raghunathan, «An Energy-Efficient Computing Technique for Error Resilient Applications,» 2015. DOI: 10.1109/ISVLSI.2015.130.

[11] K. S. Mohamed, «Approximate Computing: Towards Ultra-Low-Power Systems Design,» en 2019. DOI: 10.1007/978-3-030-37224-8_5.

[12] D. L. Quoc, D. R. Krishnan, P. Bhatotia, C. Fetzer y R. Rodrigues, «Incremental Approximate Computing,» en 2019. DOI: 10.1007/978-3-319-63962-8_151-1.

[13] A. Raghunathan y K. Roy, «Energy-efficient computing with good-enough results,» 2013. DOI: 10.1109/IOLTS.2013.6604092.

[14] M. I. of Technology, *Quantization in Digital Signal Processing*, Accedido en noviembre de 2024, 2023. dirección: https://dspace.mit.edu/bitstream/handle/1721.1/108951/Quantization_DSP.pdf.

[15] A. V. Oppenheim y R. W. Schafer, *Discrete-Time Signal Processing*, 3rd. Pearson, 2010, Capítulo 6, Procesamiento digital y cuantización.

[16] D. R. B. III, *Quantization Basics*, Accedido en noviembre de 2024, 2014. dirección: https://spinlab.wpi.edu/courses/ece503_2014/5-2quantization.pdf.

[17] A. Gorantla, R. Kothapalli y T. spandana, «Developments of Approximate Computing: From Algorithm Level to System Level,» 2021. DOI: 10.1109/ic3p52835.2022.00020.

[18] W.-F. Wong, P. Roy, R. Ray y N.-M. Ho, «Compilation and Other Software Techniques Enabling Approximate Computing,» en 2018. DOI: 10.1007/978-3-319-99322-5_22.

[19] F. Fakhar, B. Javed, R. U. Rasool, O. A. Malik y K. Zulfiqar, «Software level green computing for large scale systems,» *Journal of Cloud Computing*, 2012. DOI: 10.1186/2192-113X-1-4.

[20] K. Tatsumi y T. Matsuoka, «A Software Level Calibration Based on Bayesian Regression for a Successive Stochastic Approximation Analog-to-Digital Converter System,» *IEEE Transactions on Systems, Man, and Cybernetics*, 2019. DOI: 10.1109/TCYB.2018.2795238.

[21] C. Hagleitner, F. Auernhammer, J. Sexton et al., «An Architecture for Heterogeneous High-Performance Computing Systems: Motivation and Requirements,» 2023. DOI: 10.1109/jva60410.2023.00013.

[22] C. Song, «Analysis on Heterogeneous Computing,» 2021. DOI: 10.1088/1742-6596/2031/1/012049.

[23] W. Han y J. Ning, «Heterogeneous computing architecture camera system and image processing method,» 2021.

[24] I. Alvarez, M. P. Andrew, O. Carlos et al., «Heterogeneous compute architecture hardware/software co-design for autonomous driving,» 2020.

[25] A. M. Bruaset, X. Cai, F. Desprez y M. Parsons, «Editorial: Heterogeneous computing in physics-based models,» *Frontiers of Physics in China*, 2023. DOI: 10.3389/fphy.2023.1320450.

[26] G. H. Golub y C. F. Van Loan, *Matrix computations*, 4th edition. JHU press, 2013, ISBN: 9781421407944.

[27] T. H. Cormen, C. E. Leiserson, R. L. Rivest y C. Stein, «Introduction to algorithms,» 2009.

[28] L. G. León-Vega, A. Chacón-Rodríguez, E. Salazar-Villalobos y J. Castro-Godínez, «Acceleration of Fully Connected Layers on FPGA using the Strassen Matrix Multiplication,» en *2023 IEEE 5th International Conference on BioInspired Processing (BIP)*, 2023, págs. 1-6. DOI: 10.1109/BIP60195.2023.10379257.

[29] H. Anton y C. Rorres, *Elementary linear algebra: Applications version*, 10th edition. John Wiley & Sons, 2009, ISBN: 9780470432051.

[30] G. Strang, *Introduction to linear algebra*, 5th edition. Wellesley-Cambridge Press, 2016, ISBN: 9780980232776.

[31] B. List, L.-W. Chen, K. Bali y N. Thuerey, «How Temporal Unrolling Supports Neural Physics Simulators,» *arXiv.org*, 2024. DOI: 10.48550/arxiv.2402.12971.

[32] H. Kojima, K. Yamada e Y. Tanaka, «Restoration of Time-Varying Graph Signals using Deep Algorithm Unrolling,» 2023. DOI: 10.1109/icassp49357.2023.10094838.

[33] A. Sangiovanni-Vincentelli, *IC Design and Optimization: Theory and Practice*. Springer, 2012, ISBN: 978-1461414068.

[34] M. Smith, *Application-Specific Integrated Circuits*. Addison-Wesley, 2020, ISBN: 978-0201500221.

[35] R. J. Baker, *CMOS: Circuit Design, Layout, and Simulation*, 3rd. Wiley-IEEE Press, 2011, ISBN: 978-0470881323.

[36] J. M. Rabaey, A. Chandrakasan y B. Nikolić, *Digital Integrated Circuits: A Design Perspective*, 2nd. Prentice Hall, 2011, ISBN: 978-0130909961.

[37] D. M. Harris y S. L. Harris, *Digital Design and Computer Architecture*, 2nd. Morgan Kaufmann, 2013, ISBN: 978-0123944245.

[38] F. Smarandache, M. Mohamed, Y. Zhou y M. Abdel-Baset, *A Critical Path Problem Using Triangular Neutrosophic Number*, 2017.

[39] R. A. Brownlee, «Error estimates for interpolation of rough data using the scattered shifts of a radial basis function,» *Numerical Algorithms*, 2005. DOI: 10.1007/S11075-004-3620-2.

[40] KULeuven, *Snitch cluster*. dirección: https://kuleuven-micas.github.io/snax_cluster/index.html.

[41] E. Zurich, *Snitch Documentation*. dirección: https://pulp-platform.github.io/snitch/.

[42] J. L. Hennessy y D. A. Patterson, *Computer Architecture: A Quantitative Approach*. Elsevier, 2011.

[43] U. A. Acar, G. E. Blelloch, M. Fluet, S. K. Muller y R. Raghunathan, *2. Coupling Memory and Computation for Locality Management*, 2014. DOI: 10.4230/LIPICS.SNAPL.2015.1.

[44] KULeuven, *Snitch cluster*. dirección: https://www.kuleuven.be/english/kuleuven.

[45] E. Zurich, *ETH Zurich*. dirección: https://ethz.ch/en.html.

[46] M. Lab, *Microelectronics and Computer Architecture Laboratory (MICAS Lab)*, Universidad Católica de Lovaina, KULeuven, Accedido: noviembre 2024, 2024. dirección: https://www.esat.kuleuven.be/micas/.

Apéndice A

A.1. CONVOLVE

CONVOLVE refuerza la posición de la UE en el diseño y desarrollo de procesadores de borde inteligentes, de modo que pueda convertirse en un actor dominante en el mercado mundial de procesamiento de borde. Con el auge del aprendizaje profundo (DL), nuestro mundo se prepara para la Inteligencia Artificial (IA) en todos los dispositivos periféricos, lo que crea una necesidad urgente de hardware de procesamiento periférico de IA. A diferencia de las soluciones existentes, este hardware debe soportar un procesamiento de IA de alto rendimiento, fiable y seguro a un consumo ultrabajo (ULP), con un plazo de comercialización muy corto.

Con su sólido legado en soluciones de vanguardia y plataformas de procesamiento abiertas, la UE está en una posición ideal para convertirse en líder de este mercado de inteligencia artificial de vanguardia. Sin embargo, para que la UE se sitúe a la cabeza de estas tecnologías, es necesario cumplir ciertos requisitos: Los procesadores de borde deben ser 100 veces más eficientes energéticamente; su complejidad exige un diseño automatizado con una reducción del tiempo de diseño 10 veces mayor; deben ser seguros y fiables para ser aceptados; por último, deben ser flexibles y potentes para soportar el dominio DL.

CONVOLVE da respuesta a estas demandas y, de este modo, hace posible el liderazgo de la UE en Edge-AI. Amplie más al visitar la referencia [7].

A.2. KU Leuven

KULeuven (Katholieke Universiteit Leuven) es una universidad ubicada en la ciudad de Lovaina, en Bélgica. Es una de las universidades más prestigiosas y antiguas de Europa, fundada en 1425. KULeuven es conocida por su excelencia en investigación y educación, especialmente en áreas como la ingeniería, la ciencia, la medicina y las humanidades.

La universidad tiene una fuerte orientación hacia la investigación, con numerosos laboratorios y centros de investigación que colaboran con instituciones internacionales y la industria para desarrollar innovaciones tecnológicas y científicas. Además, KULeuven tiene

una notable presencia en el ámbito académico global, con una amplia oferta de programas de grado y posgrado en diversas disciplinas. [44]

A.3. ETH Zúrich

ETH Zurich (Eidgenössische Technische Hochschule Zürich) es una de las universidades más reconocidas del mundo en el ámbito de la ingeniería, la tecnología y las ciencias naturales. Fundada en 1855, se encuentra en Zurich, Suiza, y es una de las principales universidades técnicas del país.

ETH Zurich es conocida por su excelencia en investigación y formación en áreas como la ingeniería eléctrica, la informática, la física, la biotecnología y la arquitectura. Ha producido numerosos premios Nobel y otros galardonados por sus contribuciones científicas. Su enfoque interdisciplinario en la investigación y la educación la ha posicionado como líder global en innovación y tecnología. [45]

Apéndice B

B.1. MICAS Lab

MICAS Lab (Microelectronics and Computer Architecture Laboratory) es un laboratorio de investigación que se centra en el diseño y desarrollo de sistemas de microelectrónica y arquitectura computacional. El laboratorio está vinculado a la Universidad Católica de Lovaina (KU Leuven), una de las universidades más importantes de Bélgica, con una fuerte reputación en ingeniería y ciencias aplicadas.

MICAS Lab realiza investigaciones avanzadas en áreas como circuitos integrados, sistemas electrónicos de alto rendimiento, diseño de microprocesadores, arquitecturas computacionales y técnicas de optimización para mejorar el rendimiento de sistemas electrónicos y computacionales. Los proyectos desarrollados en MICAS Lab suelen abordar desafíos en computación de alto rendimiento, eficiencia energética y diseño de hardware especializado para aplicaciones como la inteligencia artificial, redes de comunicaciones y otros campos tecnológicos. [46]

B.2. ECAS Lab

El ECAS Lab es un laboratorio de investigación para el diseño de computación aproximada y procesos de síntesis aproximada en FPGA de la escuela de Ingeniería Electrónica del ITCR. Su tarea es la investigación en proyectos relacionados a High Performace Computing y diseño de hardware aproximado.

B.3. DCI Lab

El DCILab consiste en el laboratorio de diseño de circuitos integrados de la escuela de ingeniería electrónica del ITCR. Este laboratorio tiene la tarea de llevar a cabo el flujo completo de VLSI (very large scale integration) para el diseño y pruebas de circuitos integrados.