Análise e Comparação de Estratégias de Implementação Física de Multiplicadores Matriciais

Rafael R. de Aguiar[1], **Isaías Felzmann**[2]

[1]Faculdade de Engenharia de Computação – Pontifícia Universidade Católica de Campinas

[2]Instituto de Computação – Universidade Estadual de Campinas

rafael.ra3@puccampinas.edu.br, ibf@unicamp.br

Abstract. *This work presents an experimental comparison of three architectures of 4x4 matrix multiplication units: parallel, multicycle and pipeline, implemented in RTL and synthesized for an FPGA Cyclone IV board. Motivated by the importance of matrix multiplication for AI applications and dissatisfied with the results of non-specialized systems, we aim to investigate alternatives for a model dedicated to matrix operations. We have included an explanation on the design of the models and a comparison of the synthesis results, focusing on area and maximum frequency obtained with the intention to evaluate the performance of each of the models in a constant stream of operations. Our results show that the parallel model utilizes 31.91% more logic elements than the multicycle with a reduction of 16.37% on operating frequency, demanding 60% of the bandwidth needed for the pipeline model. This suggests a better balance between area, latency and bandwidth of the parallel model in comparison to the others.*

Resumo. *Este trabalho apresenta uma comparação experimental de três arquiteturas de unidades de multiplicação de matrizes 4×4: paralela, multiciclo e pipeline, implementadas em RTL e sintetizadas para uma placa FPGA Cyclone IV. Motivados pela importância da multiplicação matricial para aplicações de IA e insatisfeitos com resultados de sistemas não especializados, visamos investigar alternativas para um modelo dedicado a operações matriciais. Incluímos uma explicação do funcionamento dos modelos e uma comparação dos resultados da síntese com foco em área e frequência máxima obtidos a fim de avaliar o desempenho de cada um dos modelos em um fluxo contínuo de operações. Nossos resultados mostram que o modelo paralelo utiliza 31,91% mais elementos lógicos que o multiciclo com uma redução de 16,37% na frequência de operação, exigindo 60% da largura de banda de dados necessária para o modelo pipeline. Isso sugere um melhor equilíbrio entre área, latência e largura de banda do modelo paralelo em comparação aos demais.*

1. Introdução

O uso pessoal e comercial de Inteligência Artificial tem crescido de forma substancial nos últimos anos [McKinsey & Company 2024]. Um limitante de desempenho nesses sistemas é a multiplicação de vetores e matrizes, que possui complexidade computacional custosa e lida com grande volume de dados [Stothers 2010]. Devido à sua característica de processamento vetorial, unidades de processamento gráfico (GPUs) são comumente

empregadas para acelerar esse tipo de operação [Song et al. 2022]. Apesar de GPUs serem eficientes na aceleração de operações matriciais [Ansari and Ansari 2025], através de uma interpretação vetorial das matrizes, ainda há uma busca por maior eficiência através de sistemas especializados nessas operações [Giasemis et al. 2025].

Escolher os mecanismos adequados para multiplicação é delimitante para o desempenho e a eficiência de um sistema especializado de processamento matricial. Diferentes modelos requerem quantidades de recursos diferentes. Enquanto um modelo pode possuir uma área menor, ela é compensada pelo maior tempo necessário para a conclusão da operação. De modo semelhante, onde a velocidade seja a maior prioridade, dimensões em área devem aumentar. É importante a comparação entre diferentes modelos para buscar o modo mais adequado de concluir a operação.

A arquitetura da unidade de multiplicação de dados matriciais pode ser implementada de formas distintas: em paralelo, em que as multiplicações ocorrem simultaneamente para todos os elementos da matriz, em um único ciclo; em multiciclo, que possui área reduzida, e resultados parciais são acumulados a cada ciclo; e sequenciando um pipeline, no qual os cálculos parciais são sequencializados em uma estrutura que permite diversas operações ao mesmo tempo, separadas por estágios.

O objetivo desse trabalho é experimentar as possibilidades de implementação dessa unidade e comparar os resultados entre os três modelos de implementação paralelo, multiciclo e pipeline. Para isso, eles foram implementados em RTL, sintetizados em FPGA e avaliados quanto suas características de frequência máxima e área, de modo a entender padrões do projeto. Esse experimento traz as seguintes contribuições fundamentais: (a) um levantamento de alternativas para um módulo de multiplicação de matrizes; (b) a avaliação de eficiência em área e latência desses modelos; e (c) um estudo sobre as características de projeto que influenciam a eficiência.

Os resultados demonstram que, muito embora a implementação paralela tenha maior área, o modelo multiciclo não traz uma diminuição suficiente em área para compensar o aumento significativo do tempo de conclusão da operação. O modelo pipeline, apesar de parecer promissor pelo número maior de multiplicações completas a longo prazo, necessita de um aumento substancial de largura de banda de memória além de um aumento na área e utilização de registradores, assim pontuando o equilíbrio encontrado na utilização do modelo paralelo para a dimensão que estudamos.

2. Fundamentos de multiplicação matricial

Uma multiplicação matricial se define pela multiplicação de uma matriz A, com M linhas e K colunas, e uma matriz B, com K linhas e N colunas, resultando em uma matriz de dimensões [M][N]. A representação da Figura 1 exemplifica a operação para matrizes de dimensões $M = N = K = 2$, onde cada elemento [i][j] da matriz resultante representa a soma dos produtos dos elementos [i][k] da matriz A e [k][j] da matriz B, com k percorrendo a dimensão dos vetores:

$C_{ij} = \sum_{k=0}^{K-1} A_{ik} B_{kj}$.

Dois métodos dessa operação serão utilizados:

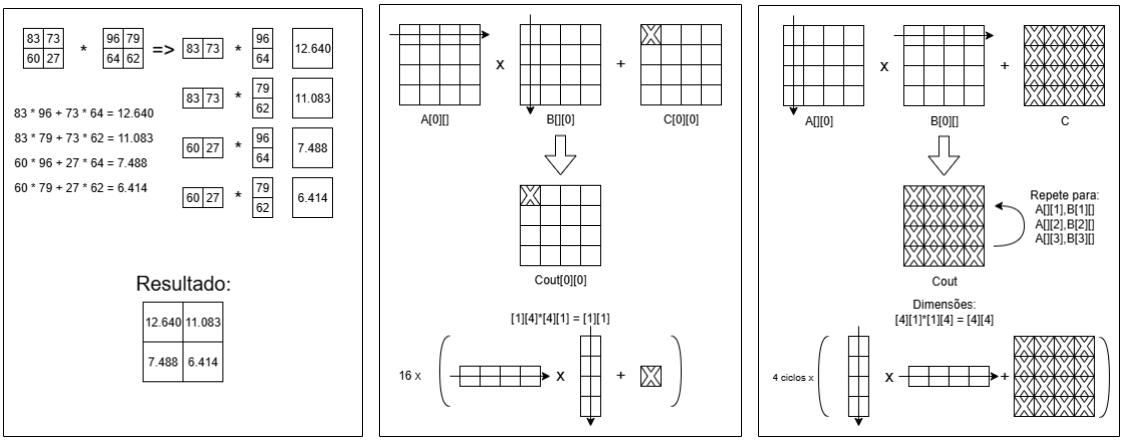


Figura 1. Métodos de multiplicação de matrizes: (a) Exemplo ilustrativo; (b) Produto interno; (c) Produto externo.

2.1. Produto Interno (*Inner Product*)

Nesse método, um elemento C_{ij} da matriz resultado é computado de cada vez. A dimensão *k* é percorrida inteiramente, e cada elemento de uma linha *i* da matriz A é multiplicado por um elemento da coluna *j* da matriz B, ambos na mesma posição em relação ao começo do "vetor" utilizado para essa operação (linha ou coluna da matriz). Esses produtos são acumulados e resultam no elemento C_{ij} da matriz resultado. No caso da operação MAC (multiplicação e acumulação) o elemento C_{ij} é somado ao produto, como visível na Figura 1(b). Esse processo acontece para cada um dos elementos da matriz, assim ocorrendo 16 vezes em uma matriz 4x4 como a exemplificada na Figura, com 4 multiplicações escalares por elemento, resultando em um total de 64 multiplicações.

2.2. Produto Externo (*Outer Product*)

Nesse outro método, os elementos C_{ij} da matriz resultado são computados em partes. A dimensão *k* é mantida fixa, e cada elemento *ik* da matriz A é multiplicado por cada elemento *kj* da matriz B, formando um elemento da matriz resultante parcial C_{ij}^{k}. O processo se repete para todos os valores de *k* e suas matrizes resultantes parciais são acumuladas e somadas com a matriz C, como descrito na Figura 1(c). Como o processo, no modelo que utilizamos, possui 16 multiplicações escalares para cada matriz resultante parcial e 4 parciais, são efetuadas 64 multiplicações escalares no total, assim como no caso do *inner product*.

3. Trabalhos relacionados

Diversos trabalhos têm sido feitos na área de multiplicação de matrizes, dando destaque a [Kelefouras et al. 2016], que de forma similar ao nosso projeto, explora formas de tornar mais eficiente a operação de multiplicação matricial, mas essa pesquisa tem um grande foco em acessos de memória e configuração de cache, enquanto nossa pesquisa está centrada nos processos dos cálculos sendo feitos em hardware.

Tiramos inspiração na construção da arquitetura dos modelos do núcleo de RISC-V do [LowRISC 2025], que apesar de não possuir uma unidade matricial, sua arquitetura de operações vetoriais e multiplicadores multiciclo serviu de inspiração para construção

dos modelos e levantou questionamentos sobre a melhor forma de aplicar a interação dos elementos da unidade de multiplicação.

A pesquisa de [Liu et al. 2024] também se trata da implementação de um multiplicador de matrizes em uma placa FPGA, porém com as dificuldades e especificidades do foco dado à pesquisa de matrizes esparsas e seu funcionamento único. Em contrapartida, como seguimos um caminho generalizado, tivemos mais espaço e foco para preocupação da eficiência, funcionamento e performance dos modelos em um nível mais físico.

4. Modelagem dos multiplicadores

Desenvolvemos multiplicadores de matrizes de tamanho 4x4 elementos inteiros de 32 bits, resultando em matrizes de 512 bits, compatível com unidades de processamento matricial como Intel AVX-512. Além disso, as matrizes quadradas facilitam no particionamento do cálculo, onde os operandos são subdivididos, como acontece com algoritmos como BLAS [Pirova et al. 2026]. Utilizamos o Intel Quartus 20.1 para simulação e síntese dos modelos em RTL, escritos em Systemverilog e sintetizados para a FPGA Altera Cyclone IV EP4CE115F29C7, sem *register retiming*. O período alvo foi estipulado de forma que o *timing slack* correspondesse a 10-15% do período, proporcionando uma margem de segurança para possíveis oscilações. Os modelos e suas especificações são:

4.1. Implementação puramente paralela

O modelo paralelo é único que utiliza do método *inner product*. Diferentemente dos demais, o resultado das operações acontece em apenas um ciclo: as 64 multiplicações necessárias para a operação acontecem paralelamente, seguidas de uma árvore de redução, também com 64 elementos de soma. Assim, a árvore de redução torna o resultado das 4 multiplicações parciais em um único elemento. O resultado da redução é adicionado ao devido elemento da matriz C para completar o MAC. O processo está ilustrado na Figura 2.

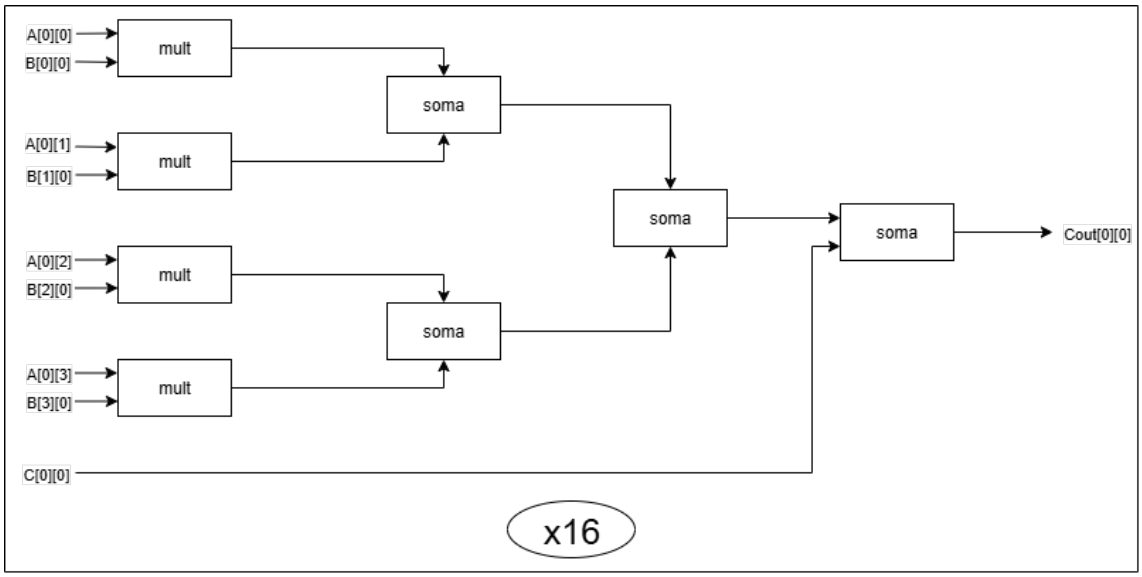


Figura 2. Implementação do multiplicador em paralelo.

A simplicidade desse modelo é um dos motivos para a sua consideração na implementação de unidades de multiplicação matricial. No entanto, seu caminho crítico

pode se provar muito longo, o que torna o modelo mais lento do que desejado. Para resolver essa situação, os outros modelos foram desenvolvidos procurando uma eficiência maior em questão de multiplicações matriciais concluídas por período de tempo.

4.2. Implementação multiciclo

O modelo de multiciclo utiliza de *outer product* para completar a multiplicação e acumulação de forma particionada. A Figura 3 representa o diagrama de funcionamento para a coluna 0 de uma matriz 4x4.

O sinal colA é responsável por selecionar a coluna de A e a linha de B que estão sendo calculadas no momento. Além disso, também é utilizado para definir o elemento a ser somado com o resultado da multiplicação durante o ciclo, a matriz C no primeiro ciclo e o valor acumulado em ciclos subsequentes.

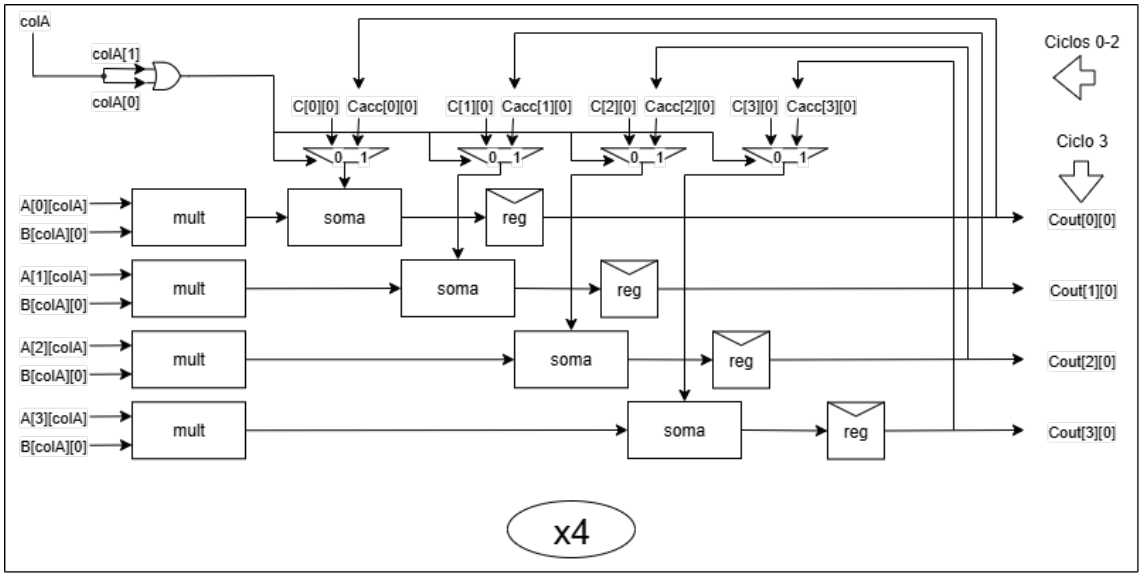


Figura 3. Implementação do multiplicador multiciclo.

Esse modelo necessita de 4 ciclos para computação da matriz resultante C. No entanto, sua área é menor devido aos 16 multiplicadores e somadores, em comparação aos 64 multiplicadores e somadores do modelo paralelo e pipeline, concluindo apenas 1/4 das operações por ciclo em comparação aos outros modelos.

4.3. Implementação pipeline

De forma similar ao multiciclo, cada estágio do pipeline possui uma multiplicação parcial também utilizando do *outer product*, onde o primeiro estágio soma a multiplicação da coluna 0 de A pela linha 0 de B com a matriz C original. Estágios subsequentes somam os valores da coluna N de A e linha N de B com a matriz acumulada recebida pelos registradores entre estágios (sendo N o número do estágio), como na Figura 4.

A vantagem desse modelo, é que operações sequenciais preenchem os estágios durante os ciclos iniciais, garantindo que resultados dessas operações aconteçam a cada ciclo a partir do 4°. Isso torna o modelo mais eficiente com um número maior de operações subsequentes. Assim como o paralelo, ele completa 64 multiplicações e somas a cada clock, mas estão agrupadas nesse modelo em grupos de 16 por estágio.

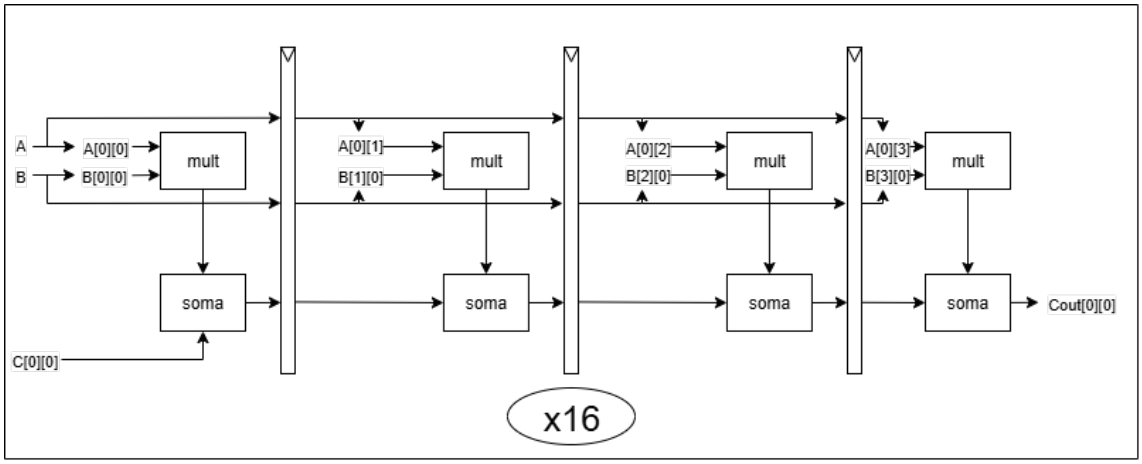


Figura 4. Implementação do multiplicador em pipeline.

5. Avaliação e Resultados

Após garantir que os resultados dos modelos em RTL estavam de acordo com o esperado, a síntese foi realizada para encontrar os valores de área (Figura 5) e frequência máxima (Figura 5) de cada modelo. Para contexto, adicionamos ao gráfico de frequência máxima a informação reportada pelo fabricante como sendo a frequência máxima suportada pelos DSPs da FPGA utilizada [Altera Corporation 2016].

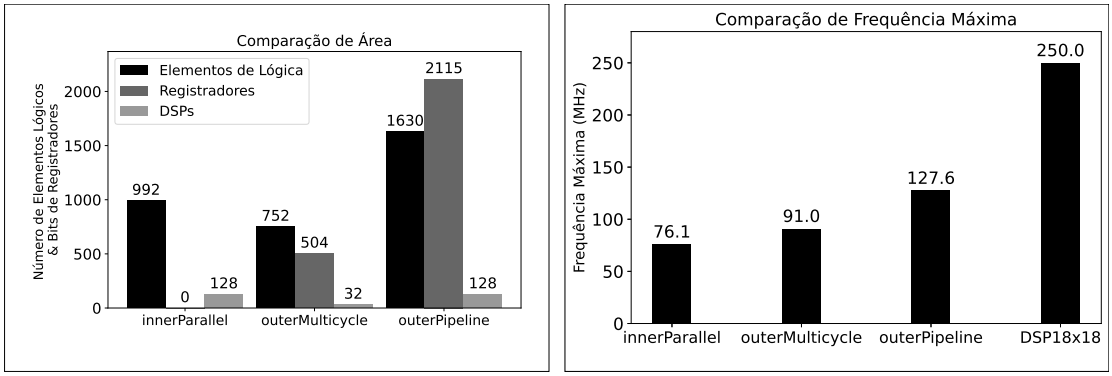
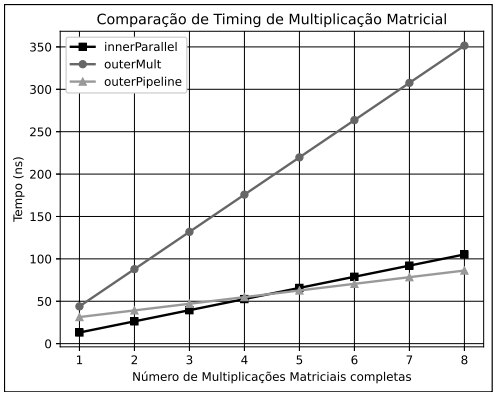


Figura 5. Comparação de (a) área utilizada e (b) frequência máxima entre os multiplicadores paralelo, multiciclo e pipeline.

Como esperado, o menor valor de frequência encontrado pertence ao modelo paralelo, pelo seu maior caminho crítico. O multiciclo, apesar de possuir frequência máxima mais elevada, tem área significativamente alta em comparação com o pipeline, produzindo um ganho pouco vantajoso. A redução de 25,2% de utilização de elementos de lógica da FPGA do paralelo para o multiciclo resultou em apenas uma redução de 16,5% do período, além do fato que o multiciclo necessita de 4 ciclos para completar a operação, causando um aumento de 234,2% no tempo de conclusão de uma única operação.

O pipeline, por sua vez, teve um aumento de uso de elementos de lógica de 64,3%, com uma redução de 40,4% do período mínimo, o que resulta em uma maior eficiência a partir da quinta operação consecutiva. Isso fica evidenciado pelo resultado da Figura 6, que mostra o tempo necessário para executar um número crescente de multiplicações consecutivas em cada um dos modelos, considerando a sua frequência máxima de operação.

O tempo inicial no qual o pipeline está sendo “preenchido” pelas primeiras operações é responsável pelo atraso dos resultados iniciais do pipeline em comparação com o modelo paralelo.



	paralelo	multiciclo	pipeline
Latência por operação (ns)	13,148	43,94	31,356
Frequência Máxima (MHz)	76,1	91,0	127,6
Elementos de Lógica	992 (0,87%)	752 (0,66%)	1630 (1,42%)
DSPs	128 (24,1%)	32 (6,0%)	128 (24,1%)
Largura de banda (Gbps)	117	35	196

Figura 6. Resumo de resultados de (a) tempo para concluir multiplicações matriciais consecutivas e (b) comparativo de latência, frequência máxima, área e vazão dos multiplicadores.

No entanto, para o funcionamento do modelo como esperado, a cada ciclo, a unidade precisa ter acesso a 3 matrizes para operar os cálculos, ou um total de 1536 bits por ciclo. Com um período mínimo calculado de 7,84 ns, isso significa uma largura de banda necessária de 196 Gbps. Em comparação aos outros modelos, o paralelo requer 117 Gbps (ciclos de 13,1 ns) e o multiciclo 35 Gbps (ciclos de 11 ns) [Figura 6]. A redução signficativa de largura de banda necessária para o multiciclo se deve pela divisão da operação ao longo de 4 ciclos, permitindo que os dados sejam parcialmente carregados.

Na Figura 5 é possível também visualizar a diferença de utilização de registradores e de DSPs dos modelos. Como a utilização de DSPs é proporcional à quantidade de multiplicações realizadas, o pipeline e o paralelo mantiveram valores idênticos de DSPs, 128, enquanto o multiciclo contém um quarto desse valor. Quanto aos registradores, apesar do multiciclo ser menor que o paralelo em questão de Elementos de Lógica, há uma quantidade significativa de registradores utilizados pelo multiciclo, enquanto o paralelo não possui registradores. O pipeline, além de sua área maior de elementos de lógica, possui uma quantia de registradores aproximadamente 4 vezes maior do que a quantia utilizada no multiciclo, devido à necessidade de transportar informação utilizada em cada um dos estágios.

Com o tamanho atual de matrizes, os modelos paralelo, multiciclo e pipeline utilizaram respectivamente, 0,87%, 0,66% e 1,42% de elementos de lógica disponíveis na placa e o multiciclo ocupou 6,01% dos DSPs, enquanto os outros modelos ocuparam 4 vezes isso, ou 24,1%. Não foi possível a análise de matrizes maiores com a placa utilizada devido à quantidade de recursos disponíveis na placa de desenvolvimento utilizada. A próxima dimensão quadrada e potência de dois, 8x8, já não pode ser sintetizada para o dispositivo alvo com apenas 532 DSPs.

6. Conclusão

A multiplicação de matrizes é um elemento crucial para o desempenho de diversas aplicações computacionais, como computação científica e aprendizado de máquina.

Unidades de multiplicação matricial podem ser aplicadas para resolução desse problema. Nesse trabalho, comparamos a área e frequência de operação de três modelos de implementação de multiplicadores matriciais. Os resultados mostram que uma arquitetura paralela, embora custosa em termos de área, pode apresentar um bom custo-benefício devido a sua menor latência. O modelo em pipeline apresenta um desempenho promissor, porém traz desafios adicionais de largura de banda de memória para manter a unidade utilizada, o que pode não escalar para matrizes maiores. Essa característica pode ser mitigada através da divisão da própria multiplicação escalar, embutida dentro dos módulos matriciais, em estágios, ao invés da divisão das matrizes. Em trabalhos futuros, pretendemos estudar o desempenho de arquiteturas que explorem esse quesito, bem como unidades de multiplicação maiores, além de analisar outros aspectos de relevância para a aplicação prática desses modelos, como eficiência energética.

Agradecimentos

Este trabalho recebeu apoio do Instituto de Pesquisas Eldorado e da Softex.

Referências

Altera Corporation (2016). *Cyclone IV Device Handbook, Volume 3*. Altera Corporation, San Jose, CA. Altera Publication Number: CYIV-53001-2.1.

Ansari, M. Q. and Ansari, M. Q. (2025). Racing to idle: Energy efficiency of matrix multiplication on heterogeneous cpu and gpu architectures.

Giasemis, F. I., Lončar, V., Granado, B., and Gligorov, V. V. (2025). Comparative analysis of fpga and gpu performance for machine learning-based track reconstruction at lhcb.

Kelefouras, V., Kritikakou, A., Mporas, I., and Kolonias, V. (2016). A high-performance matrix–matrix multiplication methodology for cpu and gpu architectures. *The Journal of supercomputing*, 72(3):804–844.

Liu, Y., Chen, R., Li, S., Yang, J., Li, S., and da Silva, B. (2024). Fpga-based sparse matrix multiplication accelerators: From state-of-the-art to future opportunities. *ACM Transactions on Reconfigurable Technology and Systems*, 17(4):1–37.

LowRISC (2025). Ibex risc-v core documentation. Version latest (as of 2025-08-06).

McKinsey & Company (2024). The State of AI in Early 2024. Report, McKinsey & Company.

Pirova, A., Vodeneeva, A., Kovalev, K., Ustinov, A., Kozinov, E., Liniov, A., Volokitin, V., and Meyerov, I. (2026). Performance optimization of blas algorithms with band matrices for risc-v processors. *Future Generation Computer Systems*, 174:107936.

Song, L., Chi, Y., Guo, L., and Cong, J. (2022). Serpens: A high bandwidth memory based accelerator for general-purpose sparse matrix-vector multiplication. In *Proceedings of the 59th ACM/IEEE design automation conference*, pages 211–216.

Stothers, A. J. (2010). *On the Complexity of Matrix Multiplication*. PhD thesis, The University of Edinburgh. Doctoral thesis, School of Mathematics.