NINA: Processador RISC-V para Tiny Machine Learning

Guilherme Moreno Silva

Orientador: Alfredo Goldman

Instituto de Matemática e Estatística - Universidade de São Paulo

Introdução

A desaceleração da Lei de Moore tem ameaçado o ritmo de inovação em capacidade de processamento. Nesse contexto, *hardware-software co-design*, a abordagem de projetar processadores especificamente para algoritmos e vice-versa, surge como uma alternativa para superar essas limitações. Investigamos a eficácia de *co-design* com o NINA, um microcontrolador RISC-V otimizado para redes neurais pequenas. Comparamos o desempenho do NINA e outras duas variantes: uma sem otimizações e outra com otimizações gerais, não focadas em aplicações específicas.

Objetivos e Desafios

- Objetivos e Requisitos
  - Avaliar o sucesso e aplicabilidade de *co-design*
  - Necessário encontrar aplicação simples e que estresse a CPU
- Software
  - Inferência em redes *feedforward* satisfazem os requisitos
  - Camadas densas são o produto escalar entre entradas e pesos
  - Multiplicação de matrizes estressam fator tradicionalmente limitante nas CPUs: desvios condicionais (`branches`)
- Hardware
  - CPUs utilizam *pipelining* para paralelizar instruções
  - Desvios condicionais são não-determinísticos (devo desviar?)
  - Prevê-los (desviar ou não) aumenta a complexidade da CPU, utilizando mais transistores
  - Não prever impede que outras instruções sejam processadas

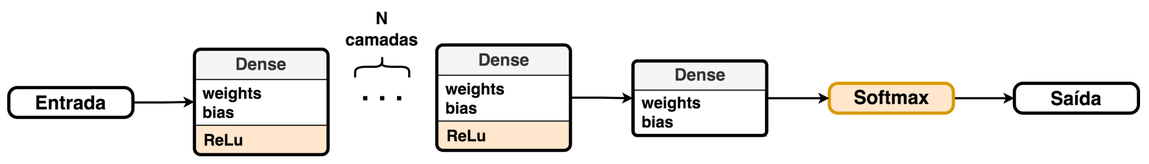
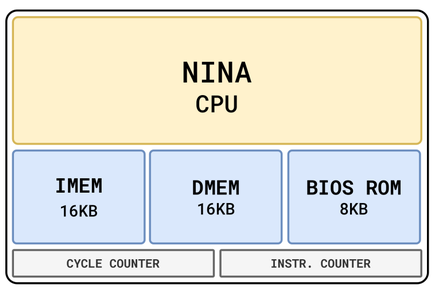


Figura 1: *Arquitetura de rede neural usada como benchmark*

NINA (NINA Is Not ARM)



- Implementa o RISC-V RV32I
- Três estágios de *pipeline*
- Reduz *stalls* resolvendo desvios no primeiro estágio

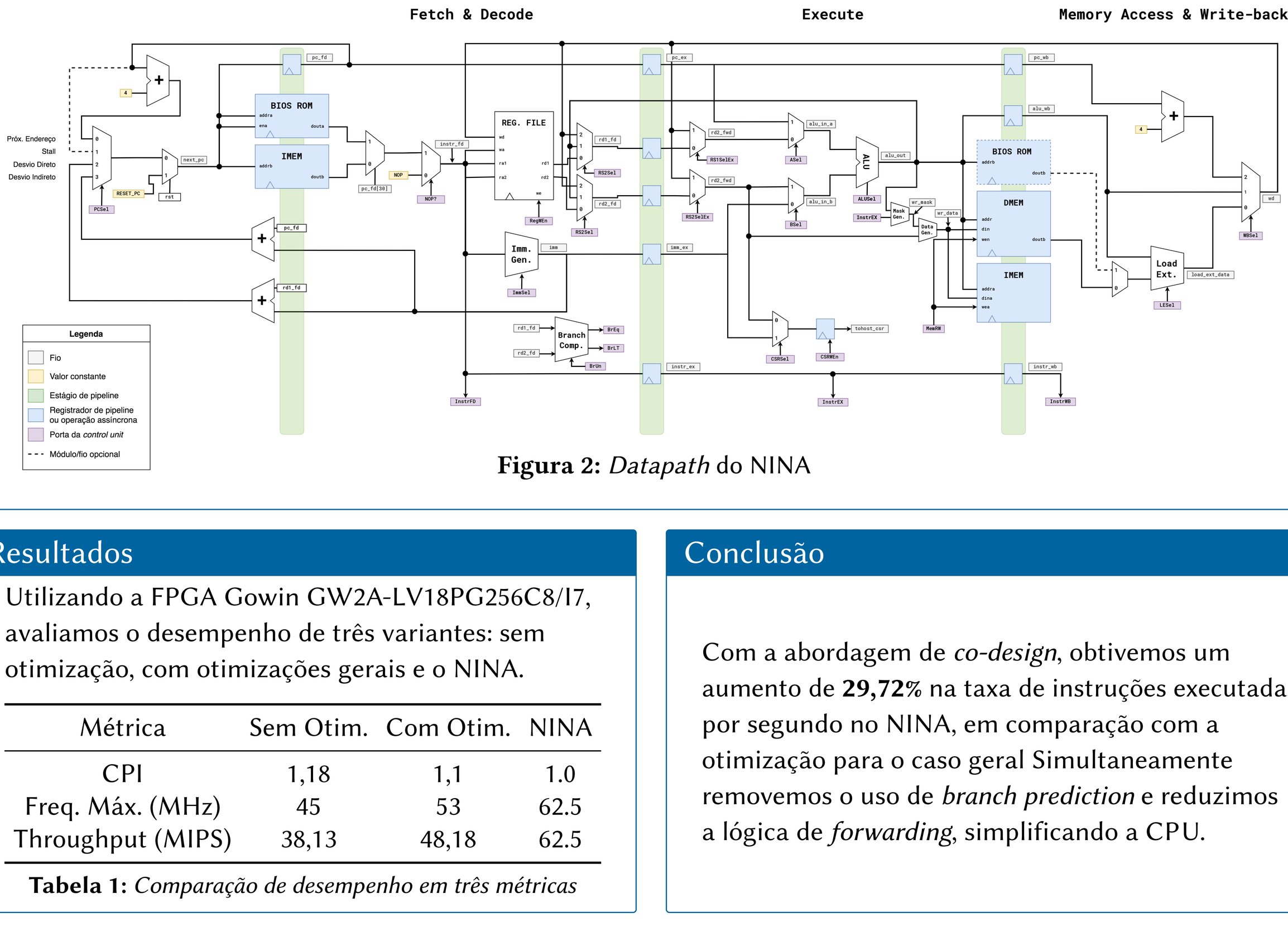


Figura 2: *Datapath* do NINA

Resultados

Utilizando a FPGA Gowin GW2A-LV18PG256C8/I7, avaliamos o desempenho de três variantes: sem otimização, com otimizações gerais e o NINA.

Métrica	Sem Otim.	Com Otim.	NINA
CPI	1,18	1,1	1.0
Freq. Máx. (MHz)	45	53	62.5
Throughput (MIPS)	38,13	48,18	62.5

Tabela 1: *Comparação de desempenho em três métricas*

Conclusão

Com a abordagem de *co-design*, obtivemos um aumento de **29,72%** na taxa de instruções executadas por segundo no NINA, em comparação com a otimização para o caso geral Simultaneamente removemos o uso de *branch prediction* e reduzimos a lógica de *forwarding*, simplificando a CPU.