Aceleração de hardware em sistemas embarcados para aprendizado de máquina utilizando KNN em FPGA

Wanderson Berbert*
wberbert@id.uff.br
Universidade Federal Fluminense
Rio das Ostras, Rio de janeiro, Brasil

Luciano Bertini
lbertini@id.uff.br
Universidade Federal Fluminense
Rio das Ostras, Rio de janeiro, Brasil

Alessandro Copetti
alessandro_copetti@id.uff.br
Universidade Federal Fluminense
Rio das Ostras, Rio de janeiro, Brasil

ABSTRACT

Aprendizado de máquina tem se tornado uma ferramenta essencial para qualquer sistema de tomada de decisão. Devido a limitações de performance impostas por arquiteturas tradicionais que utilizam *Central Processing Unit*s (CPUs), para aplicações mais críticas, métodos de aceleração com *Graphical Processing Unit* (GPU) e *Application Specific Integrated Circuit* (ASIC) têm sido empregados. No entanto, quando aplicadas a sistemas embarcados, estas apresentam limitações relacionadas a tamanho físico e complexidade. Para resolver estes problemas, a utilização da tecnologia *Field Programmable Gate Array* (FPGA) tem se mostrado promissora devido a sua grande eficiência, paralelismo real, reconfigurabilidade e flexibilidade. Diante disso, este estudo tem como objetivo, além de fazer uma revisão aprofundada da bibliografia, apresentar arquiteturas projetadas em FPGA que buscam minimizar tais limitações, maximizando a eficiência, sem perda de performance significativa e de modo a viabilizar sua utilização em sistemas embarcados. Resultados mostram ganhos em performance acima de 95% quando utilizando um hardware especialista desenvolvido em FPGA utilizando o algoritmo de aprendizado de máquina *K-Nearest Neighbor* (KNN).

KEYWORDS

FPGA, Aprendizado de máquina, Aceleração de Hardware e Sistemas Embarcados.

1 INTRODUÇÃO

A necessidade de miniaturização de dispositivos com o intuito de atender premissas impostas por sistemas embarcados como os utilizados em Internet das coisas (IoT) e a demanda crescente por sistemas inteligentes cada vez mais independentes, como os exigidos na indústria 4.0, veículos autônomos, e-health, smart cities, entre outros, é um grande desafio e uma oportunidade para a implementação eficiente de algoritmos de classificação, como, por exemplo, os algoritmos de aprendizado de máquina.

Tecnologias atuais buscam executar o processamento de diferentes algoritmos em um espaço de tempo muito curto, permitindo sua utilização em aplicações de tempo real em sistemas embarcados com baixos recursos. Um exemplo desses algoritmos são os utilizados nas técnicas de *Aprendizado de Máquina* (AM) ou aprendizado de máquina. Estas aplicações necessitam de uma alta acurácia e executam algoritmos computacionalmente complexos, resultando na necessidade de hardware de alta performance [Du and Du 2018].

Atualmente, a maioria das aplicações que envolvem AM é processada em arquiteturas híbridas utilizando CPUs de uso geral e/ou GPUs. Em um datacenter, onde virtualmente não há limitações de recursos (energia, espaço físico, memória e processamento), tal arquitetura atende aos requisitos. No entanto, devido às limitações dos sistemas embarcados o uso desta arquitetura se torna limitado.

Neste trabalho vamos investigar os possíveis ganhos em eficiência na utilização do FPGA em sistemas embarcados, através da utilização ou não de um algoritmo de aprendizado de máquina acelerado em hardware, no próprio FPGA. O FPGA é uma tecnologia promissora que oferece ganhos em performance, aliados ao baixo consumo de energia em uma arquitetura flexível. O estudo visa responder algumas perguntas sobre a performance de aplicações em um hardware com limitação de recursos, comparando-os com as mesmas aplicações porém sem a utilização dos recursos de aceleração do FPGA. O método escolhido para testes foi o método KNN de aprendizado de máquina, constantemente utilizado em sistemas embarcados, por ser um algoritmo relativamente simples porém com alto custo computacional.

2 CONCEITOS BÁSICOS E TRABALHOS RELACIONADOS

Nesta seção apresentamos os conceitos básicos das tecnologias envolvidas no trabalho, mesclados em uma revisão bibliográfica aprofundada nas áreas de aplicações do FPGA em sistemas embarcados e aprendizado de máquina. Nesse contexto, foram avaliados trabalhos relacionados a IoT, ASIC e implementação de processadores em FPGA.

2.1 FPGA e a Internet das Coisas (IoT)

FPGA são circuitos integrados reprogramáveis que contêm blocos de lógica programável. As principais características de um chip FPGA são sua flexibilidade, confiabilidade e paralelismo. Permitem aos designers desenvolverem uma lógica digital customizável em campo [Moore 2017] e que podem ser eletricamente programados para se tornar quase qualquer tipo de sistema ou circuito digital [Kuon et al. 2008].

O paradigma FPGA aliado a evolução dos sistemas embarcados em sistemas *System on Chip* (SoC) trouxe novas possibilidades para sua utilização, sendo uma delas em sistemas relacionados a IoT.

Circuitos lógicos eletrônicos são descritos na tecnologia FPGAs através das linguagens denominadas *Hardware Description Language* (HDL), que permitem programar a estrutura, operação e design dos circuitos. A linguagem HDL possui uma descrição textual composta de operadores, expressões, estruturas e operações de entrada e saída. Ao invés de gerar um *assembly* executável no computador, o HDL provê um mapa de portas, algumas vezes mencionados como *bitstream*. O mapa de portas obtido é carregado no dispositivo FPGA com objetivo de verificar sua operação. A

*

linguagem permite descrever qualquer circuito digital de forma estrutural e comportamental.

Um exemplo de aplicação de FPGA em sistemas embarcados é o apresentado por Rodríguez et al. [2018], que desenvolvem um framework utilizado no desenvolvimento de sistemas cyber-physical para computação na borda (Edge Computing). A arquitetura apresentada oferece um hardware dinamicamente adaptável com alto desempenho em sistemas embarcados explorando o paralelismo a níveis de tarefa e de dados. Nela é possível ajustar os recursos de hardware em tempo de execução de modo a atender as mudanças de requisitos de energia, performance e tolerância a falhas. Concluiu-se que utilizando o framework o processamento ficou 13 vezes mais rápido e 9,5 vezes mais eficiente energeticamente do que sua versão equivalente puramente baseada em software.

O uso de IoT na indústria tem se tornado cada vez mais uma realidade. Rupani and Sujediya [2016] propuseram uma implementação para sensoriamento de sistemas industriais baseados em FPGA utilizando o paradigma IoT. Foi apresentado um conceito de utilização em monitoramento com o objetivo de criar um sistema de baixo custo, robusto, amigável e de monitoramento 24 por dia em tempo real. Os autores conclui que o FPGA é uma tecnologia apropriada para monitoramento e controle em tempo real para sistemas industriais, suportando um grande número de dispositivos.

2.2 ASIC versus FPGA

Aplicações que executam tarefas específicas de alta performance onde é necessário o máximo de eficiência geralmente utilizam um hardware ASIC específico para sua execução. Com o surgimento do FPGA estudos surgiram com o intuito de mostrar sua viabilidade em utilizá-los em substituição ao ASIC.

Hou et al. [2012] apresentaram uma arquitetura de um divisor de frequência de alta acurácia tendo sua implementação feita tanto de ASIC quanto em FPGA. O estudo relacionou os recursos necessários e os desafios para cada tecnologia. Foram utilizados como hardware FPGA o XC3S400 da Xilinx e para o ASIC tecnologias de 180 nm e 90 nm. O estudo concluiu que ambas apresentam uma excelente acurácia. No entanto, para o ASIC foi mencionada a necessidade de uma área física muito maior devido a dificuldade em implementar operações de ponto flutuante, necessitando de muitos *gates*, e que devido a esse fato essa tecnologia pode se tornar inapropriada em algumas situações especiais.

Uma comparação das tecnologias FPGA e ASIC para o desenvolvimento de circuitos foi apresentado por Greggain [1990]. Os autores analisaram o processo de desenvolvimento, a metodologia, o design e os custos envolvidos para ambos. Foram implementados quatro circuitos em ASIC e em FPGA, utilizando como hardware FPGA o Xilinx, Altera e Actel. O estudo mostrou que dos quatro circuitos, dois eram impossíveis de serem implementados em FPGA devido a limitações de hardware e os outros dois foram de sete a nove vezes mais lentos que seu equivalente em ASIC. Segundo o autor, tal fato se deve às limitações do circuito lógico e dificuldades no roteamento. O autor menciona também a metodologia de implementação bottom-up e top-down que requer muito mais disciplina do que a metodologia interativa utilizada com o FPGA. O trabalho concluiu que, embora seja muito difícil comparar tecnologias tão diferentes, o ASIC leva vantagem sobre o FPGA, pois a performance do circuito é melhorada em termos de velocidade, consumo de energia, redução de custos e confiabilidade do circuito. Embora o ano de publicação seja antigo, o estudo é importante pois mostra a inviabilidade do FPGA na época e nos permite constatar o enorme salto dado por essa tecnologia nos dias atuais. Novos dispositivos híbridos combinando CPU e GPU com FPGA estão sendo lançados, além de soluções que minimizam suas restrições.

Por exemplo, em uma tentativa de mitigar algumas restrições dos circuitos FPGA relacionados principalmente à sua performance e tempo de compilação, Podobas et al. [2020] apresentam o conceito de Coarse-Grained Reconfigurable Architectures (CGRA). Segundo a metodologia de prototipação CGRA, aumentando-se a granularidade do circuito programado é possível obter-se sistemas digitais maiores e mais especializados além de um aumento na performance. O trabalho concluiu que o FPGA pode superar as GPUs se levarmos em consideração sua performance relacionada a sua eficiência energética, porém, o estudo ressalta que o escopo não abrange arquiteturas com alto grau de complexidade sendo necessário mais tempo de pesquisa.

Abid and Izeboudjen [2015] compararam o FPGA com o ASIC em aplicações de baixo consumo de energia. Foram utilizados para o estudo hardwares FPGA da Xilinx e Altera mais recentes para a época e ASIC utilizando tecnologias de 0.13 µm. O estudo analisou o consumo de energia estática e dinâmica dos componentes para as duas arquiteturas em dois projetos diferentes, um circuito contador e um circuito de transformação de imagens. O estudo mostra que o consumo de energia utilizando ASIC é muito menor que o FPGA, porém o estudo chama a atenção para que o autor descreve como *Embedded FPGA* (e-FPGA), onde o FPGA Core é incluído em uma arquitetura SoC visando flexibilidade e trazendo melhor eficiência no consumo de energia. Os experimentos mostraram que o menor consumo de energia foi obtido utilizando o chip Cyclone FPGA, porém o mesmo foi 20 vezes maior que seu equivalente em ASIC. O estudo conclui que o FPGA nunca irá competir com o ASIC no quesito consumo de energia devido a arquitetura intrínseca do ASIC, porém o estudo deixou a possibilidade para utilização do FPGA na área de prototipação de circuitos ou quando a necessidade de flexibilidade é mais importante que o consumo de energia. Nesses casos, tal como na área de sistemas embarcados, a utilização do FPGA é viável.

Kuon and Rose [2007] propuseram uma nova metodologia para mensurar de forma quantitativa as diferenças entre circuitos implementados em FPGA e seu equivalente em ASIC nos quesitos, área ocupada, consumo de energia e performance. O estudo utilizou para comparação tecnologia FPGA e ASIC de 90 nm. A metodologia empregou o uso de benchmarks, escritos em HDL, selecionados de forma a assegurar que sejam sintetizados similarmente em ambas as arquiteturas. Para efeitos de comparação foi utilizado o hardware FPGA Altera Stratix II. Adotou-se na comparação o pior e o melhor cenário de acordo com a velocidade do hardware FPGA utilizado. Verificou-se que o FPGA ocupou em seu pior caso 54 vezes mais área que um ASIC e no melhor caso 9,5 vezes mais. Em relação à performance o FPGA foi em seu pior caso 6,7 vezes mais lento que o ASIC e em seu melhor caso 2,5 vezes mais lento. Em relação ao consumo de energia o FPGA foi 52 vezes maior que o ASIC no pior caso e no melhor caso 5,3 vezes. O estudo concluiu que arquiteturas em FPGA utilizando apenas o que o autor chamou de elementos

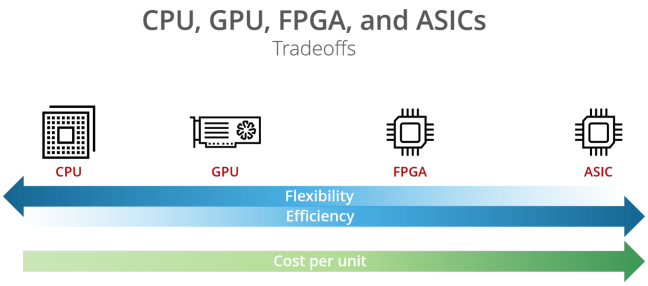

Figure 1: CPU, GPU, FPGA, ASIC Tradeoffs. Fonte: [Shimoni 2018]

lógicos *Look-up Table* (LUT) são em média 35 vezes maiores e de 3,4 a 4,6 vezes mais lentos que sua versão em ASIC.

2.3 Sistemas híbridos ASIC e FPGA

Na Computação, aceleração de hardware é a utilização de uma arquitetura feita exclusivamente para atender a um propósito específico de forma a executar as operações mais eficientemente do que se a mesma fosse feita utilizando um software para uma CPU de uso geral. Um exemplo clássico de aceleração é a utilizada para cálculo de ponto flutuante. Operações de ponto flutuante são muito mais rápidas utilizando *Floating Point Unit* (FPU) do que se feitas exclusivamente pela CPU.

A Figura 1 apresenta o compromisso existente entre as tecnologias na hora de se projetar mecanismos de aceleração de hardware. Quanto mais eficiente o hardware menos flexível ele é, ou seja, mudanças são mais difíceis de serem implementadas. Por outro lado, quanto mais flexível o hardware mais fácil é a implementação de mudanças, porém, ao custo de eficiência.

Sistemas com processadores FPGA possuem como principal característica a flexibilidade, porém, quando a premissa é performance, sistemas ASIC são implementados em sacrifício à flexibilidade. Quando a performance é necessária, mas, sem abrir mão totalmente da flexibilidade, sistemas híbridos podem ser a solução. Associando processadores desenvolvidos em hardware ASIC com partes desenvolvidas em FPGA, pode-se aliar performance e eficiência sem abdicar completamente da flexibilidade.

Também com base no hardware FPGA Stratix-II, Brzoza-Woch and Nawrocki [2015] utilizaram Aplicação Orientada a Serviço - *Service Oriented Application* (SOA), em um conceito experimental de web service utilizando FPGA em conjunto com Microcontrolador - *Microcontroller Unit* (MCU). O estudo criou uma arquitetura que pode ser reconfigurável com o objetivo de executar atividades computacionalmente intensivas de forma mais eficiente.

A ideia de utilizar sistemas híbridos não é nova. Salcic [1997] elaborou um sistema híbrido composto de MCU e FPGA e o chamou de PROTOS. O objetivo dessa arquitetura foi de aliar a flexibilidade de um software à reconfigurabilidade do FPGA a fim de atender as premissas impostas pelas aplicações embarcadas. Como exemplo de aplicação da arquitetura foi implementado um semáforo onde o controle foi executado pelo FPGA e a configuração e reconfiguração foi executada pela MCU. O estudo concluiu que, utilizando a reprogramabilidade do microcontrolador e a reconfigurabilidade do FPGA, o poder conjunto das tecnologias pode ser utilizado eficientemente.

Vavouras et al. [2016] propuseram a utilização de um circuito híbrido FPGA e ASIC de um pâncreas artificial. Tal sistema, segundo o estudo, é muito mais confiável e muito mais tolerante a falhas que os sistemas atuais por justamente utilizar o FPGA em sua elaboração. Quando uma falha permanente é detectada as técnicas tradicionais permitem que o módulo seja isolado do restante do circuito. Utilizando FPGA, tal falha poderia ser corrigida reconfigurando a região do chip onde a falha ocorreu ou realocando a funcionalidade para outra área do chip caso a falha ainda persista após a reconfiguração. O pâncreas artificial foi projetado utilizando *Very High Speed Integrated Circuit Hardware Description Language* (VHDL) com hardware FPGA Xilinx ML605 Virtex 6. O estudo concluiu que no circuito empregado a Probabilidade de Falhas por Hora - *Probability of Failures per Hour* (PFH) foi 5100 vezes menor para falhas permanentes com um acréscimo de 2,4 vezes na área utilizada e a PFH foi 85 vezes menor para falhas transientes com um acréscimo de 1,6 na área.

2.4 FPGA e processadores SoftCores

O termo SoftCore refere-se a CPUs implementadas exclusivamente utilizando a tecnologia FPGA. Com base nessa premissa muitos fornecedores desenvolveram suas próprias versões, como o NIOS2, LEON3 e PicoBlaze. Pesquisadores utilizam esses processadores ou criam suas próprias versões em seus estudos, principalmente devido a sua flexibilidade. Processadores desenvolvidos em FPGA permitem ser otimizados para atender as particularidades de um projeto. Alguns processadores desenvolvidos por terceiros vêm com o código fonte em linguagens HDL, o que permite ao pesquisador modificá-los conforme a necessidade.

Prado [2019] em seu estudo comparou microcontroladores com processadores SoftCore no que diz respeito a performance, custo, flexibilidade e consumo de energia. Foi sintetizado para o estudo processadores SoftCores capazes de executar as mesmas instruções de 8 bits dos microcontroladores da família 16F84 da Microchip. O processador foi comparado com os microcontroladores PIC16F84 e PIC16F877. Os resultados do estudo mostraram que o SoftCore foi em todos os aspectos mais eficiente que os microcontroladores originais. Os mais avançados processos de *Integração de Larga Escala - Very Large Scale Integration* (VLSI) aliado a flexibilidade oferecida pelo FPGA foram papéis chaves para velocidade e eficiência energética, conferindo uma grande vantagem em relação aos microcontroladores. A única desvantagem mencionada foi a necessidade de projetos mais complexos e um tempo maior de verificação mas que poderiam se tornar insignificantes caso estes SoftCores fossem disponibilizados pelos fornecedores. O FPGA foi 6,9 vezes mais rápido e 28 vezes mais energeticamente eficiente com praticamente o mesmo custo.

2.5 Aprendizado de máquina em FPGA

A aceleração de hardware através do FPGA tem sido utilizada com sucesso em sistemas embarcados onde os recursos são limitados, necessitando assim o máximo de eficiência, um exemplo disso é sua aplicabilidade em aprendizado de máquina. Estudos mostram que o ganho de eficiência obtido com o FPGA tem tornado essa

plataforma mais atrativa para desenvolvimento, uma vez que as perdas impostas por limitações de hardware podem ser em grande parte eliminadas adotando uma utilização mais eficiente dos recursos fornecidos pelo FPGA em uma batalha constante entre performance e eficiência [Diniz et al. 2017]. A aceleração de hardware fornecida pelo FPGA também tem sido aplicada com sucesso na aceleração de algoritmos de *Deep Learning* (DL) em particular *Covolutional Neural Network* (CNN) em aplicações de reconhecimento de imagens relacionados ao *Computer Vision* (CV) [Shawahna et al. 2019].

Possa et al. [2011] apresentaram uma metodologia para utilização de aceleração de hardware com FPGA. Foram analisados diversas arquiteturas de hardware bem como exploradas diversas interfaces de comunicação com a CPU, analisando sua performance, recursos utilizados, consumo de energia e métodos de implementação. O estudo concluiu que a escolha da arquitetura e interfaces têm papel fundamental na performance final. Com a utilização de aceleração de hardware é possível conseguir uma redução significativa no consumo de energia e aumento de performance, tornando possível sua utilização em sistemas embarcados.

Afifi et al. [2015] mostram o ganho que é possível obter utilizando a aceleração de hardware fornecida pelo FPGA em aprendizado de máquina com o algoritmo *Suport Vector Machine* (SVM), inclusive comparando-as com as GPUs com resultados promissores para o FPGA. Segundo os autores, FPGA é poderoso o suficiente para executar o SVM pois é altamente paralelizável, reconfigurável e muito eficiente na utilização dos recursos computacionais disponíveis.

Com um FPGA Zilinx operando a 337 Mhz máquina Siddiqui et al. [2019] mostraram um ganho de 8 vezes em clusterização e 9,6 vezes em reconhecimento de imagens utilizando aprendizado de máquina pelo método K-means em comparação a um processador ARM equivalente. O mesmo apresentou uma eficiência energética 1,7 vezes maior que outras arquiteturas equivalentes (ARM Cortex-A7 CPU, nVIDIA GeForce GTX980 GPU e ARM Mali-T628).

Zhao et al. [2017] apresentaram um modelo mostrando os ganhos na utilização do FPGA na aceleração de algoritmos de machine learning CNN e SVM. Esse mesmo trabalho o comparou com processadores ARM e GPU, além de procurar desenvolver uma arquitetura padrão de implementação otimizado que pudesse ser utilizada facilmente. O estudo conclui que a aceleração de hardware proveniente da utilização do FPGA aumenta consideravelmente a performance de algoritmos de *Aprendizado de Máquina* (AM).

Frances-Villora et al. [2016] apresentaram o conceito de *Exterme Machine Learning* (EML) e sua implementação em FPGA aplicados a sistemas em tempo real. O trabalho avaliou três arquiteturas selecionando a que melhor atendia as premissas de processamento em tempo real, concluindo que o FPGA é uma plataforma eficiente, pequena e flexível de modo a trazer novas possibilidades para implementação de aprendizado de máquina em sistemas do tipo SoC.

Cambre et al. [2008] relacionaram a eficiência energética de um processador desenvolvido em FPGA com o tamanho atribuído ao cache de memória. O estudo concluiu que tamanhos otimizados de cache de memória resultam em menores consumos de energia.

Estudo semelhante foi realizado por Xu et al. [2012], porém o mesmo analisou a performance de sistema multiprocessados baseados em FPGA. O estudo concluiu que o ganho de performance em processadores dual-core aumenta juntamente com a complexidade computacional, quando comparados ao cálculo de 64-bits ou mais o tempo de processamento foi reduzido em torno de 50% no sistema dual-core.

Sun and Cheng [2012] propuseram uma arquitetura para sistemas embarcados baseado em *Artificial Neural Networks* (ANN) utilizando FPGA para classificação de doenças cardíacas, tendo com base as informações fornecidas pelo eletrocardiograma (ECG). A implementação em FPGA foi 5000 vezes mais rápida e gastou 4000 vezes menos energia que sua implementação feita utilizando smartphones. O autor conclui também que é possível uma redução de 98,7% de utilização de área, 99,1% de economia de energia dependendo da arquitetura utilizada, porém com o sacrifício do tempo de resposta.

Como se pode notar em todos esses trabalhos, devido às suas características, a aplicação do FPGA em aprendizado de máquina tem sido promissora, principalmente para os algoritmos que podem se beneficiar do paralelismo, como por exemplo o SVM e KNN. O hardware apresenta uma grande eficiência também para as arquiteturas de DL, citando alguns exemplos o CNN e ANN.

Essas afirmações são evidenciada por Nurvitadhi et al. [2017] onde se discute a eficiência do FPGA e sua possível aplicação para AM. Recentemente, Mittal [2020] onde o autor discute sobre a aceleração em algoritmos que utilizam CNN, citando o FPGA como uma tecnologia promissora e Abdelrahman [2020] onde fala sobre a aceleração de aprendizado de máquina utilizando FPGA e CPU.

Wei et al. [2019] em seu estudo focaram na otimização do projeto de hardware de um CNN para utilização em operações com inteiros com baixa largura de bits utilizando quantização. O estudo conclui que com a utilização de números inteiros, CNNs podem ser implementados eficientemente em hardware de baixo consumo de energia com apenas uma pequena perda na acurácia.

3 ARQUITETURAS PROPOSTAS

Apresentaremos a seguir o desenvolvimento de 2 (duas) arquiteturas implementadas em FPGA com o projeto em HDL: a primeira foi denominada SoftCore e a segunda SoftCore com hardware KNN especialista. O objetivo será avaliar a viabilidade dessas arquiteturas em aplicações embarcadas, cujo emprego de algum tipo de aceleração de hardware seja imprescindível. Já na Seção 4, as arquiteturas serão avaliadas, utilizando implementações do algoritmo KNN.

A arquitetura nomeada SoftCore não possui aceleração de hardware e será utilizada principalmente para comparações, enquanto a arquitetura SoftCore com hardware KNN especialista utiliza aceleração de hardware e tem como objetivo avaliar o ganho em performance e eficiência. Nessa arquitetura toda a informação a ser classificada é enviada ao hardware especialista, desenvolvido em HDL, o qual realiza o processamento e retorna a classe predita.

Para o desenvolvimento das arquiteturas foi utilizada a DE2-115, originalmente desenvolvido pela Altera e atualmente pertencente a Intel, que possui o chip Cyclone IV.

O FPGA DE2-115 possui uma quantidade bastante limitada de elementos lógicos. Esses elementos permitem armazenar as configurações do circuito desenvolvido e as entradas e saídas, para interface com os elementos internos da arquitetura e o software desenvolvido. Os elementos lógicos em particular limitam a quantidade de circuitos que podem ser configurados na linguagem HDL.

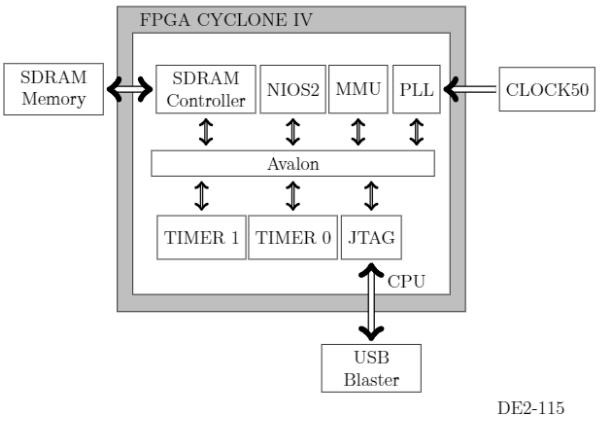

Figure 2: Arquitetura Softcore. Fonte: Desenvolvido pelo Autor

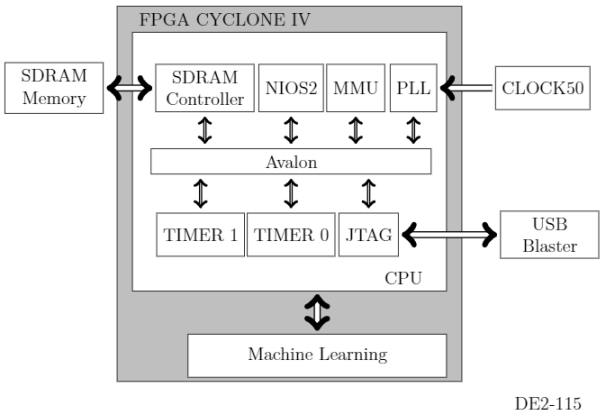

Figure 3: Softcore com hardware especialista. Fonte: Desenvolvido pelo Autor

Devido a quantidade de elementos lógicos disponíveis no hardware, algumas limitações foram impostas à arquitetura SoftCore com hardware KNN especialista. Uma delas foi o número máximo de exemplos utilizado para treinamento do modelo que teve que ser reduzido a no máximo 18.

3.1 Arquitetura Softcore

A arquitetura SoftCore está apresentada na Figura 2. Ela contém um único processador descrito por software. O propósito do SoftCore é executar os experimentos de aprendizado de máquina sem utilizar a aceleração de hardware. O SoftCore executará o algoritmo em sua totalidade, tanto a aplicação quanto também executar a parte do algoritmo responsável pelo aprendizado de máquina.

Além do processador SoftCore, esta arquitetura é composta por:

- *Memory Management Unit* (MMU). Uma MMU é um dispositivo de hardware que traduz endereços virtuais em endereços físicos. É geralmente implementada como parte da CPU, mas pode também estar na forma de um circuito integrado separado.
- *Phase Locked Loop* (PLL). Um PLL é um sistema de controle que gera um sinal de saída que está em fase com o sinal de entrada. Existem diferentes tipos, o mais simples é um circuito eletrônico composto de um oscilador variável e um detector de fase em loop retro-alimentado. O oscilador gera um sinal periódico, o detector de fase compara a fase do sinal com a fase do sinal de entrada, ajustando o oscilador para manter a fase em sincronismo.
- NIOS2 é a CPU responsável pelo processamento das instruções. O NIOS 2 é um processador 32-bits de arquitetura embarcada feita especificamente pela Altera para utilização com FPGA.
- O SDRAM Controller permite designers criarem sistemas em dispositivos FPGA que facilmente se conectam a SDRAM Memory.
- Avalon é o barramento de dados responsável pela interconexão dos componentes dentro da CPU.
- TIMER 0 e TIMER 1 são os temporizadores responsáveis nas operações críticas que envolvem sincronismo, liberando a CPU para realizar outras tarefas.
- *Joint Test Access Group* (JTAG) é uma interface de programação e teste de circuitos digitais, padronizada com o IEEE 1.149,1. Originalmente desenvolvido para a programadores lógicos, o JTAG também é frequentemente utilizado para microcontroladores.
- CLOCK50 é o oscilador que produz um sinal de 50 MHz que é utilizado como referência para o Chip FPGA.
- USB Blaster é a conexão física entre a placa de desenvolvimento DE2-115 e o computador.
- *Synchronous dynamic random-access memory* (SDRAM) é o chip de memória *Random Access Memory* (RAM) disponível na placa de desenvolvimento DE2-115.

3.2 Arquitetura Softcore com hardware KNN especialista

Esta segunda arquitetura desenvolvida, representada pela Figura 3, executará os mesmos experimentos de aprendizado de máquina realizados na primeira arquitetura. Porém, esta utilizará de aceleração de hardware para a execução da parte do algoritmo responsável pelo aprendizado de máquina. A arquitetura é idêntica a arquitetura SoftCore, porém com adição de um hardware especialista nomeado "Machine Learning" e responsável pelo aprendizado de máquina. Parte do algoritmo será executado em uma arquitetura SoftCore e a parte do algoritmo responsável pelo aprendizado de máquina será executado em uma arquitetura de hardware FPGA.

O objetivo do hardware especialista é acelerar o processo de classificação dos dados, tornando-o mais eficiente. Utilizando o paralelismo e a eficiência do FPGA em uma arquitetura completamente definida por software utilizando HDL espera-se que o tempo necessário de execução diminua consideravelmente.

4 EXPERIMENTOS

Com base nas arquiteturas desenvolvidas foram conduzidos experimentos com o objetivo de aferir a performance do aprendizado de

Característica do Dataset:	Multivariável
Número de instâncias:	150
Área:	Vida
Característica dos atributos:	Real
Número de atributos:	4
Data de doação:	1988-07-01
Tarefas associadas:	Classificação
Valores faltantes?	Não

Table 1: Dataset Iris

Fonte: https://archive.ics.uci.edu/ml/datasets/iris

máquina com e sem aceleração de hardware. Foram realizados dois experimentos, um para cada arquitetura projetada.

Todos os experimentos tiveram a performance do algoritmo de aprendizado de máquina KNN testada utilizando o conjunto de dados (dataset) Iris. A tabela 1 mostra as características do conjunto de dados utilizado.

O método de seleção de amostras (instâncias) para treinamento será o sequencial balanceado, de forma que a base de treinamento tenha sempre o mesmo número de amostras para cada classe presente no dataset. Neste modo será selecionado um elemento de cada classe de forma sequencial até que o número de exemplos selecionados para treinamento seja igual ao número desejado, respeitando as restrições de cada projeto.

O conjunto de dados foi dividido em treinamento e teste. Por motivos de comparação, o número de exemplos de treinamento para esse experimento foi limitado a 18

4.1 Avaliação da arquitetura Softcore

Este experimento utiliza a arquitetura SoftCore e tem como objetivo aferir a performance e a acurácia do aprendizado de máquina sem a utilização do módulo especialista KNN. Todo o processo de classificação é executado em um arquitetura SoftCore sem nenhum hardware especialista, desenvolvida utilizando a linguagem C.

A Figura 4 mostra a acurácia do algoritmo de classificação KNN para valores de k entre 1 e 18. O intervalo de confiança da predição foi calculado com 95% de confiabilidade.

Para o cálculo do intervalo de confiança foi levado em conta o intervalo de confiança para a proporção, mostrado na Equação 1 e 2 [Jain 1991].

$$f = \frac{x}{n} \tag{1}$$

, onde n é o total de amostras e x o total de acertos na predição da classe.

$$I.C.(1-\alpha)\ \textit{será}: f - Z_{\frac{\alpha}{2}}\sqrt{\frac{f(1-f)}{n}} \leq p \leq f + Z_{\frac{\alpha}{2}}\sqrt{\frac{f(1-f)}{n}} \tag{2}$$

, onde f é a estimativa da proporção p.

A Figura 5 mostra os mesmo experimento porém com ênfase nos tempos de execução. Para cada valor de K foram mostrados a acurácia e o tempo médio de classificação de cada exemplo do conjunto

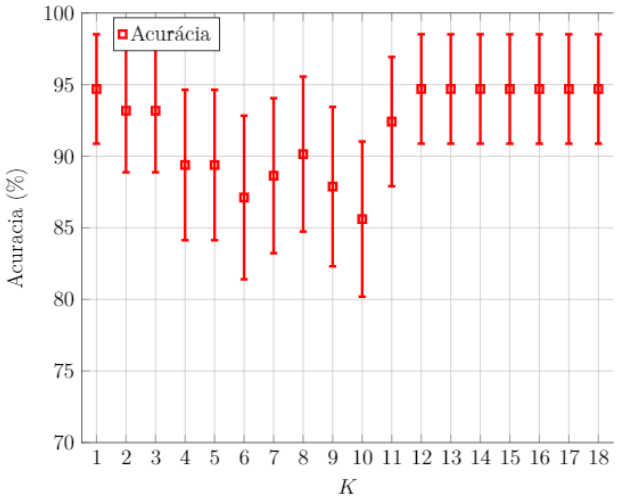

Figure 4: Acurácia do KNN em Software. Fonte: Desenvolvido pelo Autor

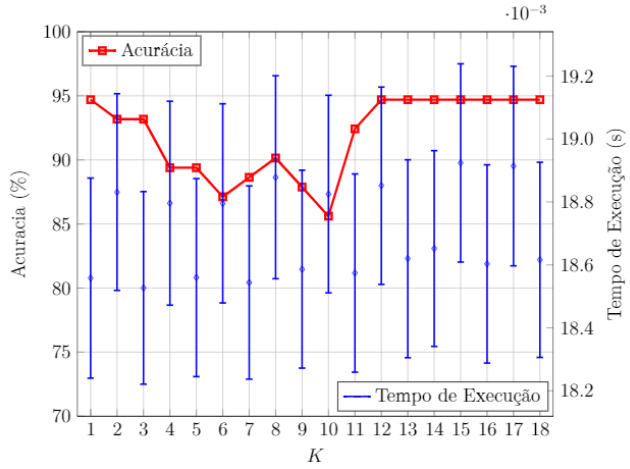

Figure 5: Acurácia e Tempo de Execução do KNN em Software. Fonte: Desenvolvido pelo Autor

de dados Iris bem como o intervalo de confiança com 95% de confiabilidade pelo método bootstrapping, método de reamostragem proposto por Bradley Efron em 1979 [Hesterberg 2011].

4.2 Avaliação da arquitetura Softcore com hardware KNN especialista

Este experimento utiliza a arquitetura SoftCore com KNN em hardware especialista e tem como objetivo aferir a performance e a acurácia do aprendizado de máquina com a utilização do módulo especialista KNN.

Nessa arquitetura o SoftCore é responsável pela execução do software, enquanto o módulo especialista, responsável pela execução do KNN. A função do software se reduz a apenas ler as informações,

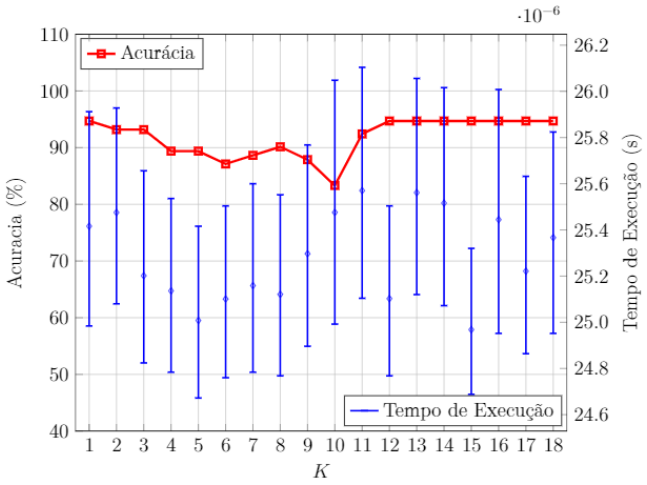

Figure 6: Acurácia e Tempo de Execução do KNN em Hardware. Fonte: Desenvolvido pelo Autor

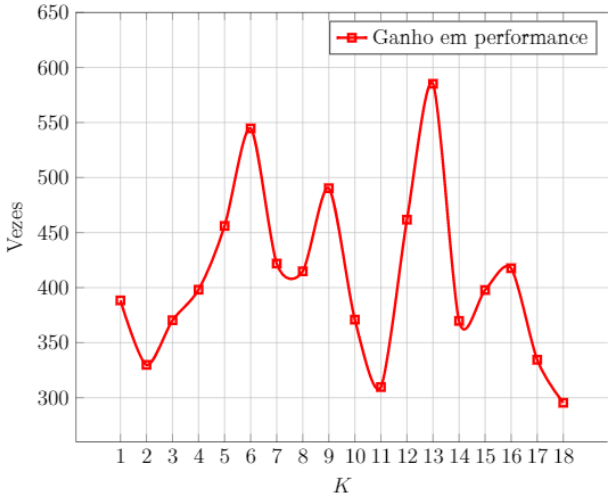

Figure 7: Ganho em performance no tempo de execução pelo Hardware. Fonte: Desenvolvido pelo Autor

enviar ao modulo especialista e obter os resultados. Já o módulo especialista "Machine Learning" fará todo o trabalho de classificação da informação e retorno para tratamento por parte do software, poupando assim tempo de processamento.

O experimento foi executado uma vez para cada valor de K, sendo que a acurácia mostrou-se idêntica a arquitetura SoftCore apresentado na Figura 4. Já a Figura 6 mostra os tempos de execução e a acurácia para cada valor de K e seus intervalos de confiança. Para cada valor de K foram mostrados a acurácia e o tempo médio de classificação de cada exemplo do conjunto de dados Iris bem como o intervalo de confiança com 95% de confiabilidade utilizando o método bootstrapping.

A Figura 7 evidencia o ganho de performance entre o classificador KNN sem aceleração de hardware que utiliza a arquitetura SoftCore e o classificador KNN que utiliza a arquitetura SoftCore com hardware KNN especialista. O ganho máximo obtido foi de aproximadamente 600 vezes para K=13 em relação ao SoftCore.

O tempo médio individual de classificação foi de aproximadamente 18 milissegundos sem aceleração e 25 micro-segundos com aceleração de hardware, sendo o experimento com aceleração de hardware 99,86% mais eficiente que seu equivalente sem a aceleração.

Ainda na Figura 7 é evidenciado o ganho em performance entre as arquiteturas SoftCore com hardware KNN especialista em relação ao SoftCore. O resultado do experimento mostra um ganho no tempo de execução entre 300 e 600 vezes em relação a arquitetura SoftCore.

5 CONCLUSÃO

O FPGA é um paradigma que abre muitas possibilidades de desenvolvimento para sistemas embarcados e IoT. Sua reconfigurabilidade permite que a funcionalidade do dispositivo seja adequada às mudanças de regras de negócio. Mesmo depois do dispositivo configurado, o paralelismo real permite que os processos não sejam concorrentes pelos mesmos recursos e sua reconfiguração parcial permite que o dispositivo seja modificado sem que afete o sistema atual em execução. Essas características são o grande diferencial desta tecnologia.

O FPGA, em muitos casos, é utilizado para prototipar sistemas ASIC até o projeto estar exaustivamente testado para ser disponibilizado definitivamente em chips com tecnologia padrão, diminuindo os custos de produção. A eficiência energética e performance quando comparado com os ASIC são inferiores, porém, a possibilidade de reconfiguração e adequação do hardware ao processo, torna a solução atrativa economicamente. Sua deficiência, bem conhecida pelos engenheiros, é compensada por sua flexibilidade.

Comparando os dois experimentos realizados verificou-se um ganho de performance de 99,98%, sem alteração na acurácia, entre a arquitetura que possuía o módulo especialista "Machine Learning" e o que não possuía, sendo este um ganho bastante significativo.

O estudo também realizou uma revisão da literatura com o intuito de verificar o estado da arte da tecnologia FPGA no contexto de sistemas embarcados e aprendizado de máquina. Verificou-se que sua eficiência, reconfigurabilidade, flexibilidade e paralelismo são uma ferramenta ideal para aplicações com aprendizado de máquina, considerando os algoritmos que podem ser representados com total paralelismo em hardware.

O desenvolvimento do hardware especialista em FPGA para o algoritmo de aprendizado de máquina KNN foi bastante desafiador, principalmente devido as restrições de recursos no hardware, que diminuíram o número máximo de exemplos suportados pelo módulo especialista. Essas restrições serão melhor exploradas em um trabalho futuro, onde pretende-se desenvolver algoritmos em software por divisão e conquista, através de execuções parciais, para atingir a meta de se resolver problemas maiores, em uma abordagem híbrida de software e hardware. Pretende-se também expandir o estudo para outros algoritmos de aprendizado de máquina.

REFERENCES

Li Du and Yuan Du. Hardware accelerator design for machine learning. In Hamed Farhadi, editor, *Machine Learning*, chapter 1. IntechOpen, Rijeka, 2018. doi: 10.5772/intechopen.72845. URL https://doi.org/10.5772/intechopen.72845.

Andrew Moore. *FPGAs For Dummies®, 2nd Intel® Special Edition*. John Wiley & Sons, Inc., 2017.

I. Kuon, R. Tessier, and J. Rose. *FPGA Architecture: Survey and Challenges*. 2008. URL https://ieeexplore.ieee.org/document/8187326.

Alfonso Rodríguez, Juan Valverde, Jorge Portilla, Andrés Otero, Teresa Riesgo, and Eduardo De la Torre. Fpga-based high-performance embedded systems for adaptive edge computing in cyber-physical systems: The artico3 framework. *Sensors*, 18(6), 2018. ISSN 1424-8220. doi: 10.3390/s18061877. URL http://www.mdpi.com/1424-8220/18/6/1877.

Ajay Rupani and Gajendra Sujediya. A review of fpga implementation of internet of things. *International Journal of Innovative Research in Computer and Communication Engineering*, 3297, 09 2016. doi: 10.15680/IJIRCCE.2016.0409098.

L. Hou, T. Zhang, J. Wang, and Y. Li. A high-accuracy design of frequency divider with fpga and asic implementation. In *2012 International Conference on Industrial Control and Electronics Engineering*, pages 1653–1656, 08 2012. doi: 10.1109/ICICEE.2012.437.

L. Greggain. Cost benefit tradeoffs for asic versus programmable logic device. In *Third Annual IEEE Proceedings on ASIC Seminar and Exhibit*, pages P1/5.1–P1/5.4, 09 1990. doi: 10.1109/ASIC.1990.186091.

A. Podobas, K. Sano, and S. Matsuoka. A survey on coarse-grained reconfigurable architectures from a performance perspective. *IEEE Access*, 8:146719–146743, 2020. doi: 10.1109/ACCESS.2020.3012084.

F. Abid and N. Izeboudjen. Technology-independent approach for fpga and asic implementations. In *2015 4th International Conference on Electrical Engineering (ICEE)*, pages 1–4. IEEE, 12 2015. doi: 10.1109/INTEE.2015.7416610.

I. Kuon and J. Rose. Measuring the gap between fpgas and asics. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 26(2):203–215, 02 2007. ISSN 0278-0070. doi: 10.1109/TCAD.2006.884574.

Arnon Shimoni. A gentle introduction to hardware accelerated data processing, 2018.

Robert Brzoza-Woch and Piotr Nawrocki. Reconfigurable fpga-based embedded web services as distributed computational nodes. pages 159–164. Annals of Computer Science and Information Systems, Volume 6, 10 2015. doi: 10.15439/2015F37.

Zoran Salcic. Protos— a microcontroller/fpga-based prototyping system for embedded applications. *Microprocessors and Microsystems*, 21:249–256, 12 1997. doi: 10.1016/S0141-9331(97)00041-0.

M. Vavouras, R. P. Duarte, A. Armato, and C. Bouganis. A hybrid asic/fpga fault-tolerant artificial pancreas. In *2016 International Conference on Embedded Computer Systems: Architectures, Modeling and Simulation (SAMOS)*, pages 261–267, 07 2016. doi: 10.1109/SAMOS.2016.7818356.

Daniel Francisco Gómez Prado. Embedded microcontrollers and fpgas soft-cores. *Department of Electrical and Computer Engineering, University of Massachusetts, Amherst, USA*, 04 2019.

Wendell Diniz, Vincent Fremont, Isabelle Fantoni, and Euripedes Nobrega. An fpga-based architecture for embedded systems performance acceleration applied to optimum-path forest classifier. *Microprocessors and Microsystems*, 52, 06 2017. doi: 10.1016/j.micpro.2017.06.013.

A. Shawahna, S. M. Sait, and A. El-Maleh. Fpga-based accelerators of deep learning networks for learning and classification: A review. *IEEE Access*, 7:7823–7859, 2019. ISSN 2169-3536. doi: 10.1109/ACCESS.2018.2890150.

P. Possa, D. Schaillie, and C. Valderrama. Fpga-based hardware acceleration: A cpu/accelerator interface exploration. In *2011 18th IEEE International Conference on Electronics, Circuits, and Systems*, pages 374–377, 12 2011. doi: 10.1109/ICECS.2011.6122291.

Shereen Afifi, Hamid Gholamhosseini, and Roopak Sinha. Hardware implementations of svm on fpga: Astate-of-the-art review of current practice. volume 2, pages 733–752, 11 2015.

Fahad Siddiqui, Sam Amiri, Umar Ibrahim Minhas, Tiantai Deng, Roger Woods, Karen Rafferty, and Daniel Crookes. Fpga-based processor acceleration for image processing applications. *Journal of Imaging*, 5(1), 2019. ISSN 2313-433X. doi: 10.3390/jimaging5010016. URL http://www.mdpi.com/2313-433X/5/1/16.

R. Zhao, W. Luk, X. Niu, H. Shi, and H. Wang. Hardware acceleration for machine learning. In *2017 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, pages 645–650, 07 2017. doi: 10.1109/ISVLSI.2017.127.

Jose V. Frances-Villora, A. Rosado-Muñoz, José M. Martínez-Villena, Manuel Bataller-Mompean, Juan Fco. Guerrero, and Marek Wegrzyn. Hardware implementation of real-time extreme learning machine in fpga: Analysis of precision, resource occupation and performance. *Computers & Electrical Engineering*, 51:139 – 156, 2016. ISSN 0045-7906. doi: 10.1016/j.compeleceng.2016.02.007. URL http://www.sciencedirect.com/science/article/pii/S0045790616300222.

D. M. Cambre, E. Boemo, and E. Todorovich. Energy evaluation in the nios ii processor as a function of cache sizes. In *2008 4th Southern Conference on Programmable Logic*, pages 55–61, 04 2008. doi: 10.1109/SPL.2008.4547732.

Meihua Xu, Huacheng Yao, and Xuan Huan. Performance test of dual-core processor system based on nios ii. In *2012 IEEE Symposium on Electrical Electronics Engineering (EEESYM)*, pages 82–85, 06 2012. doi: 10.1109/EEESym.2012.6258593.

Yuwen Sun and Allen C. Cheng. Machine learning on-a-chip: A high-performance low-power reusable neuron architecture for artificial neural networks in ecg classifications. *Computers in Biology and Medicine*, 42(7):751 – 757, 2012. ISSN 0010-4825. doi: https://doi.org/10.1016/j.compbiomed.2012.04.007. URL http://www.sciencedirect.com/science/article/pii/S001048251200073X.

Eriko Nurvitadhi, Ganesh Venkatesh, Jaewoong Sim, Debbie Marr, Randy Huang, Jason Ong Gee Hock, Yeong Tat Liew, Krishnan Srivatsan, Duncan Moss, Suchit Subhaschandra, and Guy Boudoukh. Can fpgas beat gpus in accelerating next-generation deep neural networks? In *Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, FPGA '17, page 5–14, New York, NY, USA, 2017. Association for Computing Machinery. ISBN 9781450343541. doi: 10.1145/3020078.3021740. URL https://doi.org/10.1145/3020078.3021740.

Sparsh Mittal. A survey of fpga-based accelerators for convolutional neural networks. *Neural Computing and Applications*, 32(4):1109–1139, 2 2020. ISSN 1433-3058. doi: 10.1007/s00521-018-3761-1. URL https://doi.org/10.1007/s00521-018-3761-1.

Tarek S. Abdelrahman. Cooperative software-hardware acceleration of k-means on a tightly coupled cpu-fpga system. *ACM Trans. Archit. Code Optim.*, 17(3), August 2020. ISSN 1544-3566. doi: 10.1145/3406114. URL https://doi.org/10.1145/3406114.

Xin Wei, Wenchao Liu, Lei Chen, Long Ma, He Chen, and Yin Zhuang. Fpga-based hybrid-type implementation of quantized neural networks for remote sensing applications. *Sensors (Basel)*, 19(4):924, 02 2019. ISSN 1424-8220. doi: 10.3390/s19040924. URL https://www.ncbi.nlm.nih.gov/pubmed/30813259. 30813259[pmid].

Raj Jain. *The art of computer systems performance analysis - techniques for experimental design, measurement, simulation, and modeling*. Wiley professional computing. Wiley, 1991. ISBN 978-0-471-50336-1.

Tim Hesterberg. Bootstrap. *WIREs Computational Statistics*, 3(6):497–526, 2011. doi: 10.1002/wics.182. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/wics.182.