**Universidade Federal de Itajubá**

**Felipe dos Santos**

# Sistemas Operacionais Reconfiguráveis

**Relatório final**
**Programa: PIBITI**
**Orientador: Prof. Dr. Otávio De Souza Martins Gomes**

**Itajubá**
**2021**

## AGRADECIMENTOS

Aos meus pais Simone Aparecida Almeida dos Santos e Luis Ricardo dos Santos, pelo apoio e incentivo em meus estudos.
Ao Professor Dr. Otávio De Souza Martins Gomes, por todas as orientações durante o desenvolvimento da pesquisa.
À UNIFEI (Universidade Federal de Itajubá), por proporcionar um ambiente de estudo.
Ao IESTI (Instituto de Engenharia de Sistemas e Tecnologia da Informação), por fornecer ferramentas e recursos para o desenvolvimento da pesquisa.

## RESUMO

A necessidade por desempenho e eficiência em computadores digitais para lidar com as tarefas dos usuários pode ser suprida com o paradigma de computação reconfigurável. A computação reconfigurável apresenta alguns desafios que podem ser contornados por estruturas de controle que as façam úteis e simples de utilizar. Os Sistemas Operacionais Reconfiguráveis possibilitam a abstração de conhecimentos específicos dos desenvolvedores, o que possibilita a aderência do paradigma ao público em geral em um futuro próximo. Para isso, serão levantados estudos sobre o estado da arte de Sistemas Operacionais Reconfiguráveis e também a futura implementação de um caso de estudo. O caso de estudo será o Sistema Operacional Reconfigurável *ReconOS*. O *ReconOS* apresenta as características necessárias para a implementação em *hardware* acessível, assim como a documentação e manutenção por parte dos criadores.

## LISTA DE ABREVIATURAS E SIGLAS

| | |
|---|---|
| POSIX | Portable Operating System Interface |
| ASIC | Application-Specific Integrated Circuit |
| RAM | Random Access Memory |
| FIFO | First-In First-Out |
| IOMMU | Input-Output Memory Management Unit |
| GPU | Graphics Processing Unit |
| SISD | Single Instruction Single Data |
| SIMD | Single Instruction Multiple Data |
| MIMD | Multiple Instruction Multiple Data |
| SoC | System-on-a-Chip |
| API | Application Programming Interface |
| CMP | Chi Multi-Processor |
| CWE | Common Weakness Enumeration |
| OS4RC | Operating System For Reconfigurable Computing |
| LEAP | Latency-insensitive Environment for Application Programming |
| MIT | Massachussets Institute of Technology |
| RAMPSoC | Runtime Adaptive Multi-Processor System-on-a-Chip |
| BORPH | Berkeley Operating system for ReProgrammable Hardware |
| eCos | Embedded Configurable Operating System |
| RHINO | Reconfigurable Hardware Interface for computatioN and radiO |
| RISC | Reduced Instruction Set Computer |
| SaaS | Software as a Service |
| IaaS | Infrastructure as a Service |
| PaaS | Platform as a Service |

**SUMÁRIO**

# 1 INTRODUÇÃO

A maioria dos computadores digitais de hoje em dia são organizados em camadas, onde cada camada possui a sua linguagem e um interpretador que possa entender essa linguagem. A camada relacionada a arquitetura do conjunto de instruções fornece um conjunto de instruções, como o próprio nome sugere, fixo e característico de cada arquitetura. Instruções (*Software*) determinam uma operação a ser realizada no dispositivo físico (*Hardware*) e possuem um ciclo padrão dentro do caminho de dados de toda CPU, o ciclo buscar-decodificar-executar.

Os processadores de uso geral, atualmente, apresentam conjunto de instruções fixo, o que garante grande flexibilidade, reduz custos de produção e facilita o processo de desenvolvimento por parte dos desenvolvedores de alto nível. Porém a eficiência não apresenta um bom resultado em aplicações específicas. A computação reconfigurável possibilita a modificação da camada lógica digital do processador, isto é, permite a criação de um conjunto de instruções flexível e adaptável de acordo com as instruções necessárias pela aplicação em execução.

A computação reconfigurável baseia-se em dispositivos lógicos reprogramáveis que podem atingir um desempenho elevado juntamente com a flexibilidade da programação a nível de portas lógicas. Um dispositivo de hardware típico utilizado em computação reconfigurável são as *FPGAs* (SKLIAROVA; FERRARI, 2003).

Atualmente, entende-se o sistema operacional como uma camada de software que possibilitou a interação com o hardware de maneira segura e efetiva (DEITEL; DEITEL; CHOFFNES, 2005).

O objetivo deste trabalho é realizar o estudo das tecnologias de Sistemas Operacionais reconfiguráveis que vieram suprir a demanda de uma infraestrutura mais flexível e adaptável para o controle dos dispositivos reconfiguráveis. A fomentação do tema na comunidade acadêmica nacional também é um objetivo a ser cumprido por esta pesquisa.

## 2 OBJETIVOS PROPOSTOS E JUSTIFICATIVA

Com base na grande presença de computadores digitais na sociedade em geral, a demanda por desempenho nesses dispositivos aumenta cada vez mais. Para isso, é necessário o desenvolvimento de novas tecnologias que possibilitem o aumento do desempenho e diminua o consumo de energia dos computadores digitais. A computação reconfigurável é uma arquitetura de computadores digitais que está diretamente atrelada ao aumento de desempenho e redução do consumo de energia nos dispositivos digitais. Para utilizar essa arquitetura, são necessárias camadas de abstrações que possibilitem ao desenvolvedor final desenvolver aplicações sem a necessidade de conhecimento acerca dos detalhes da arquitetura. Os Sistemas Operacionais reconfiguráveis permitem essa tarefa.

O seguinte trabalho tem como objetivo levantar informações relacionadas ao estado da arte de Sistemas Operacionais reconfiguráveis, assim como definir um caso de estudo que possibilite a implementação e o estudo mais aprofundado futuramente.

# 3 REFERENCIAL TEÓRICO

## 3.1 Sistemas Operacionais

Atualmente, entende-se o sistema operacional como uma camada de software que possibilitou a interação com o hardware de maneira segura e efetiva. A grande maioria dos dispositivos atuais apresentam um sistema operacional, gerenciando diretamente o software e hardware para que a ação desejada pelo usuário seja realizada (DEITEL; DEITEL;CHOFFNES, 2005). A história dos sistemas operacionais está diretamente ligada à evolução da arquitetura dos computadores. Em meados da década de 1940, os primeiros computadores foram construídos, como é de se imaginar, a tecnologia da época não permitia grandes avanços neste ramo, a ausência de um sistema operacional era clara. Na década seguinte, os laboratórios de pesquisa da General Motors obtiveram sucesso na implementação do primeiro sistema operacional para seu computador IBM 701. O sistema era definido por um *job*, um conjunto de instruções de programas correspondentes a uma tarefa computacional, que eram processados em lotes (DEITEL; DEITEL; CHOFFNES, 2005).

A década de 1960 foi extremamente importante devido ao surgimento dos sistemas de multiprogramação, que apresentavam capacidade de gerenciar diversos *jobs* ao mesmo tempo. Dentre os sistemas que surgiram neste período destaca-se o UNIX, programado na linguagem C. Padrões sobre o UNIX foram criados para diferentes finalidades, o POSIX tinha como objetivo flexibilizar a execução do UNIX, o MINIX, muito similar ao UNIX, porém voltado para fins educacionais (TANENBAUM; BOS, 2016). Posteriormente surgiram sistemas que vieram a ser relevantes atualmente, como o Microsoft Windows, macOS, GNU Linux e Android.

## 3.2 Arquiteturas

### 3.2.1 Computação Reconfigurável

A computação reconfigurável é um paradigma da computação que está em rápido desenvolvimento. A proposta da computação reconfigurável é trazer flexibilidade e performance para as aplicações atuais. Nesta área ainda existem muitos desafios, como a síntese de hardware ser um processo lento em relação à síntese de software, dificultando o crescimento da computação reconfigurável (ECKERT et al., 2016). Um balanço entre flexibilidade e eficiência definem a computação reconfigurável, possibilitando até superar o limite de velocidade do microprocessador para configurar uma forte conexão entre circuitos de hardware e resolver problemas específicos. A eficiência e flexibilidade do hardware pode ser melhorado através da programação incorporada nele, o balanço é alcançado mapeando o algoritmo na arquitetura. Os dispositivos reconfiguráveis são a base para esta área de estudo da computação, o processo de alterar a estrutura dos dispositivos reconfiguráveis em tempo de inicialização (BARIK; SINHA, 2016). A computação reconfigurável baseia-se em dispositivos lógicos reprogramáveis que podem atingir um desempenho elevado juntamente com a flexibilidade da programação a nível de portas lógicas. Um dispositivo de hardware típico utilizado em computação reconfigurável são as FPGA (SKLIAROVA; FERRARI, 2003). As *FPGA* são um dos dispositivos lógicos programáveis mais disponíveis no mercado atual. Inicialmente, as *FPGAs* foram substitutas dos circuitos integrados ASICs, passando a evoluir lentamente para sistemas embarcados complexos, trazendo consigo a performance dos *ASICs* e uma grande flexibilidade (BARIK; SINHA, 2016). Um conjunto de blocos lógicos configuráveis, rede de interconexão programável e células de entrada e saída compõem uma *FPGA*. A função na *FPGA* é implementada em módulos implementados nos blocos lógicos, ligados através de

interconexões. Todos os três componentes básicos da *FPGA* podem ser programados pelo usuário, programados quantas vezes forem necessários (BARIK; SINHA, 2016). Dois usos gerais da *FPGA* podem ser destacados, o uso como aceleradores de suporte para os processadores e outra aplicação no processamento cooperativo com elementos de processamento ao redor. Assim, vemos a flexibilidade de uso do dispositivo reconfigurável, dependendo apenas da sua programação lógica definida (ECKERT et al., 2016).

Atualmente, uma ou mais *FPGAs* podem trabalhar em conjunto, controlados por um *host software* para acessar e controlar as *FPGAs*. O sistema operacional host apenas traz suporte para drivers dos dispositivos para acessar a lógica reconfigurável. O que dificulta o processo é a falta de abstrações, detalhes de baixo nível que não são familiares, influenciando todo os níveis do processo de projeto. As novas arquiteturas de *FPGA*, *FPGA* multiplexado por tempo, *FPGA* parcialmente reconfigurável e *FPGA* de placas de coprocessamento, são capazes de suportar diversos circuitos dependentes ou independentes. Com todo o desenvolvimento das arquiteturas que aceitam múltiplos circuitos, vem a necessidade de um ambiente mais flexível no tempo de execução, juntamente com o gerenciamento de recursos consistente, eficiente e correto. Dessa forma, é interessante considerar a introdução dos sistemas operacionais na computação reconfigurável (WIGLEY; KEARNEY, 2001b; BARIK; SINHA, 2016).

Em geral, as maiores tarefas de um sistema operacional está em esconder *hardware* e programas de forma eficiente, limpa e consistente, em outras palavras, garantir a abstração e o gerenciamento de recursos. A abstração pode ser vista diretamente nos processos, onde tarefas são realizadas como se estivessem em um *hardware* virtual. Entrando mais profundamente no assunto, existem os threads, que são tarefas susceptíveis a serem executados paralelamente neste mesmo hardware virtual, destacando a importância da comunicação e sincronização fornecida pelo sistema operacional (ECKERT et al., 2016). Os artigos Wigley e Kearney (2001a) e Wigley e Kearney (2002) identificam pontos importantes para o sistema operacional voltado para computação reconfigurável. Carregamento de aplicações, particionamento, gerenciamento de memória, escalonamento, proteção e entrada/saída, e comunicação e sincronização são partes importantes que compõem um sistema operacional, principalmente para a computação reconfigurável. A ideia da computação reconfigurável é trazer estas funções do sistema operacional para o nível de hardware, de forma que se obtenha uma solução mais eficiente Eckert et al. (2016) trazem as seguintes discussões sobre os tópicos citados no parágrafo anterior.

O problema do particionamento nas lógicas reconfiguráveis pode ser resolvido internamente ao sistema operacional, a solução sugerida pelo autor traz a ideia da utilização de áreas reconfiguráveis dos dispositivos, chamados de ilhas, um módulo reconfigurável pode ser instanciado internamente a essa área, evitando a alocação de módulos reconfiguráveis de outras áreas. A principal desvantagem desta solução está no surgimento de fragmentações internas. Outra solução é proposta, as áreas reconfiguráveis ficam sobre uma a arquitetura de vetores unidimensionais e bidimensionais, isso reduz significativamente o efeito da fragmentação interna, apesar do surgimento da fragmentação externa, esta pode ser tratada por técnicas de desfragmentação. Para o carregamento de aplicações no sistema, os módulos reconfiguráveis devem atuar como processadores equivalentes que interagem com a *CPU*, isso é chamado de *hw-application*. O objetivo desta utilização é, como citado anteriormente, trazer a função de software para o nível de hardware, trazendo ganhos em termos de tarefas. Esta área está em constante pesquisa na área científica atual, a dificuldade de solução está na necessidade de uma comunicação e sincronização sofisticada.

O escalonamento na computação reconfigurável depende de onde está configurada a hw-application. Teoricamente, a área reconfigurável pode ser utilizada de 3 formas, possibilitar o *multitasking*, onde o contexto da hw-application precisa ser extraído, salva e restaurada. Para hw-application o contexto é dado pelas memórias e elementos de armazenamento como *flip-flops*, blocos de *RAMs* e *FIFOs* das áreas reconfiguráveis ocupadas, sendo esta uma tarefa

complexa. Dessa forma, ainda se utiliza da abordagem *single-tasking* para as *FPGAs*, dada a dificuldade de tratamento de contextos das aplicações. A comunicação e sincronização na *hw-application* tem uma grande complexidade, dada a necessidade da comunicação com o software e hardware. *Mutexes* e semáforos são necessários para a sincronização em hardware. Fornecer uma comunicação e sincronização padronizada é um importante requisito para a performance e produtividade do sistema. Por fim, a proteção e entrada/saída, o mecanismo para *hw-application* seria o modo convencional, gerenciando a proteção de memória para entrada e saída com *IOMMUs*. Sobre a necessidade de proteção, *malwares* podem ser encontrados em *hw-application*, da mesma forma que infectam aplicações em software. Eckert et al. (2016) ainda descreveram exemplos de aplicações reais de sistemas operacionais para computação reconfigurável. Exemplos como BORPH, ReconOS, R3TOS, SPREAD, CAP-OS e RTSM. Trazem diferentes abordagens de arquiteturas, interfaces e algoritmos que tratam de escalonamento, por exemplo, Adetomi (2019) propõe o uso de frameworks para fornecer técnicas e mecanismos para reconfiguração em tempo de execução, e comunicação e sincronização entre circuitos do *hardware* reconfigurável. Um dos objetivos do autor foi trazer confiabilidade ao uso de dispositivos reconfiguráveis, tratando processos de leitura e escrita de registradores, configurações de operação, entre outros, otimizando o acesso às portas do dispositivo reconfigurável. Além disso, foram utilizados frameworks voltados para a comunicação entre processos, explorando as infraestruturas de clocks da *FPGA*, uma possibilidade de uma melhor realocação de circuitos para trazer maior confiabilidade. Por fim, o autor ainda destacou que na computação reconfigurável a segurança recebe atenção insuficiente comparada à sua importância. Segurança no carregamento, na execução, conceitos de segurança de endereço, configurações de tarefas seguras e eficientes para realocação de tarefas ainda estão em estudo e foram discutidos no artigo.

### 3.2.2 Computação Heterogênea

Computação heterogênea remete ao uso de diferentes núcleos de processamento para maximizar a performance, combinando as CPUs com unidades de processamento gráficos ou *FPGAs*, conhecidos como aceleradores. Essa nova arquitetura rompe com o projeto tradicional de caminhos de processamento, trazendo novos desafios e oportunidades na área de computação de alta performance (BRODTKORB et al., 2010). Um sistema de computação heterogênea inclui máquinas heterogêneas, redes de alta velocidade, interfaces, sistemas operacionais, protocolos de comunicação e ambientes de programação, tudo isso combinado para produzir um impacto positivo na facilidade de uso e performance (KHOKHAR et al., 1993). Na computação heterogênea encontram-se os processamentos seriais e paralelos. A presença de múltiplos dispositivos no sistema traz para os programas a utilização da concorrência e paralelismo, além da melhora de performance e energia. O ambiente de computação heterogênea está se tornando mais diversificado, explotando as capacidades dos microprocessadores multinúcleos, combinados com processadores digitais de sinais, *FPGA* e *GPUs* (KAELI et al., 2015).

A busca por um desempenho cada vez melhor expandiu o domínio das arquiteturas convencionais, surgindo multiprocessadores de memória compartilhada e Chi *MultiProcessor*, a primeira consiste na conexão de diversos processadores em um único sistema de memória, enquanto o CMP contém múltiplos núcleos integrados no mesmo chip, ambos são exemplos de paralelismo de máquinas (STRINGHINI; GONÇALVES; GOLDMAN, 2012). Hoje em dia, ao analisar as máquinas mais rápidas do mundo, claramente estas máquinas se utilizam do paralelismo de forma extremamente avançada, sendo relacionados a clusters ou MPP (M*assively Parallel Processors*). Dentro da lista das máquinas mais rápidas, encontram-se alguns supercomputadores que se utilizam de aceleradores de *GPUs* da NVIDIA, portanto, a

heterogeneidade *CPU-GPU* é explorada até em sistemas que demandam um nível de processamento muito alto. Modelos de programação como o *OpenMP* podem ser utilizados para dividir o código a ser executado na CPU e na GPU, com a finalidade de explorar o paralelismo em aplicações científicas, não somente gráficas como se utiliza convencionalmente (STRINGHINI; GONÇALVES; GOLDMAN, 2012).
Com as recentes observações dos ganhos de desempenho que foram observados em aplicações envolvendo aceleradores, com baixo consumo de energia, fabricantes como *Intel*, *AMD* e *ARM* tem anunciado projetos de chips heterogêneos. Em geral, os modelos de programação dos aceleradores correspondem a uma evolução do modelo *SIMD*, onde a aplicação da instrução sobre um conjunto de dados ocorre paralelamente de forma simultânea. A heterogeneidade deste tipo de computação está justamente em se explorar todos os dispositivos presentes na máquina. Entretanto, é importante que o desenvolvedor saiba identificar as tarefas mais adequadas para a execução num acelerador e aquelas mais adequadas à uma *CPU* (STRINGHINI; GONÇALVES; GOLDMAN, 2012). *SISD, SIMD* e *MIMD* são exemplos de tipos de paralelismo que se encontram nas arquiteturas de sistemas. O *SISD* não expressa nenhum paralelismo, tratam um único fluxo de instruções e dados de forma sequencial. Já o *SIMD* são máquinas que executam o mesmo fluxo de instruções de forma paralela em cada unidade de execução, então o fluxo de dados é distinto. Por fim, o *MIMD* se diferencia por apresentar fluxos independentes em cada unidade de processamento, explorando o paralelismo, com múltiplos fluxos de dados (STRINGHINI; GONÇALVES; GOLDMAN, 2012).

### 3.2.3 Arquiteturas Híbridas de Computador

O conceito de arquitetura híbrida está diretamente relacionado à computação heterogênea, a interpretação de cada autor traz visões diferentes sobre esta relação. Alguns autores conceituam que os dois conceitos são iguais em geral, enquanto outros defendem as diferenças particulares que se podem encontrar entre eles. Técnicas podem ser combinadas para gerar as chamadas arquiteturas híbridas. A vantagem desse tipo de sistema está no sinergismo alcançado pela combinação de duas ou mais técnicas. Esse sinergismo traz um sistema mais poderoso e com menor propensão a falhas (GOLDSCHMIDT; PASSOS, 2005 apud SASSI, 2006). Com a evolução do poder de processamento das *CPUs* e *GPUs*, o paralelismo ganhou grande foco, com novas arquiteturas e *APIs*. Houve também crescimento no desenvolvimento de algoritmos que trabalham de forma hibrida, combinando as arquiteturas de *CPU* e *GPU*, embora sejam distintas. Este tipo de algoritmo aproveita de tudo que a arquitetura híbrida tem a oferecer, como a execução de instruções com maior poder de processamento (MACIEL; DUARTE, 2016). Chips híbridos contendo componentes de *CPU* e *FPGA* são novos desenvolvimentos que trouxeram um novo conceito ao mercado. Sistemas híbridos *CPU/FPGA* entregam uma estrutura que suporta tarefas em nível de *hardware* e *software*. Esta combinação traz uma customização de *hardware* bem significante. Desenvolvedores podem configurar livremente as conexões da *FPGA*, o grande poder da computação reconfigurável (AGRON et al., 2006; ANDREWS; NIEHAUS; ASHENDEN, 2004). Ao longo dos anos, o *ReconOS* vem sendo utilizado para implementar diversas aplicações em sistemas híbridos *CPU/FPGA*. Essas experiências confirmaram que a abordagem híbrida *multi-threading* oferecido pelo *ReconOS* simplifica o processo de desenvolvimento. O *ReconOS* possibilita ao desenvolvedor avaliar diferentes mapeamentos de *threads* para o *hardware* e *software* para obter sua performance no sistema alvo (AGNE et al., 2013).
A rápida velocidade de processamento de recursos reconfiguráveis da FPGA fazem com que seja uma ótima escolha como chip de aceleração. Atualmente, a arquitetura híbrida *CPU/FPGA* é uma área frequente de pesquisa visando melhorar a velocidade de processamento. A grande chave para o melhor aproveitamento da arquitetura está na eficiência de comunicação, que vem

sendo o objetivo principal de diversas pesquisas.

### 3.3 Vulnerabilidades em Sistemas Operacionais

A vulnerabilidade é uma fragilidade no sistema que torna susceptível a invasão externa no sistema. É um princípio da segurança da informação que deve ser evitada em qualquer área, podendo comprometer dados armazenados, roubar de informações, entre outras ações que caracterizam uma invasão. As vulnerabilidades podem estar relacionadas entre elas, podem ter causas diferentes para um tipo de vulnerabilidade que se enquadra em um mesmo problema. A vulnerabilidade pode ser definida como a existência de uma fragilidade ou erros de implementação e projeto que são inesperados e indesejados, que pode servir de porta de entrada para um atacante, comprometendo a segurança do sistema. Cada sistema operacional presenta seu pacote de vulnerabilidades locais e remotos susceptíveis a ataques. Segundo o Boletim de segurança da Microsoft MS08-067 em 2009, mais de 90% dos ataques recebidas pela empresa foram realizadas pela exploração da vulnerabilidade de *Buffer Overflow* (SHARMA; KUMAR; SHARMA, 2011; DAHBUR; MOHAMMAD; TARAKJI, 2011).

### 3.4 Segurança em Sistemas Operacionais

De acordo com (NETMARKETSHARE, 2020), o *Windows* estava presente em mais de 87% dos desktops na internet até o final de 2020. O *macOS* possui pouco mais de 10% de presença no mercado e o restante é dividido entre GNU/Linux, que possui por volta de 1,5% dos desktops e o *Chrome OS*, com aproximadamente 0,5% de presença nos desktops. Dado o número de usuários com o sistema da *Microsoft*, é possível compreender a razão pela qual o sistema *Windows* é o foco de ataques maliciosos. Com base no site (CVE DETAILS, 2019), que propõe uma visualização simplificada para a base de dados de vulnerabilidades nacional dos Estados Unidos, o número de vulnerabilidades encontradas entre 2015 e 2019 para as últimas versões dos sistemas operacionais são 1.104 para o Núcleo Linux, 1.111 para o *Windows* 10 e 1.188 para o *macOS*. Os dados não determinam qual sistema proporciona maior segurança ao usuário, visto que existem vulnerabilidades que oferecem diferentes tipos e níveis de perigo ao usuário, porém contribuem para demonstrar que falhas de segurança estão presentes em ambos os sistemas operacionais atuais. Com relação a sistemas operacionais móveis no mesmo período, foram registradas 2.520 vulnerabilidades no *Android* e 1.219 no *iOS*. Todas as vulnerabilidades são organizadas e numeradas de acordo com o *CWE*, conforme proposto por (THE MITRE CORPORATION, 2021). A maior parte dos ataques explora pequenos erros de código que levam ao transbordamento de *buffer* ou o uso da memória depois que ela é liberada, permitindo que o atacante insira códigos sobrescrevendo endereços de retorno, ponteiros de exceção, ponteiros de função virtual e outros dados que controlam a execução dos programas. Muitos desses problemas poderiam ser evitados se linguagens seguras fossem utilizadas no lugar de C e C + +. E mesmo com essas linguagens pouco seguras, muitas vulnerabilidades poderiam ser evitadas se os estudantes fossem mais bem treinados na compreensão das armadilhas da validação de parâmetros e dados (TANENBAUM; BOS, 2016).
Em 2020, foi disponibilizado pela (THE MITRE CORPORATION, 2021) uma tabela com os principais *CWE encontrados* entre os anos de 2018 e 2019. O destaque da tabela fica por conta de quatro vulnerabilidades associadas a um mesmo conceito, o que indica que existe uma insuficiência nas questões de autenticação e autorização. Quando um usuário solicita acesso a um recurso ele precisa ser identificado, isto é, reconhecido como um novo usuário ou como um usuário que já solicitou acesso a um recurso antes. Depois de ser reconhecido, o usuário passa pelo tríplice autenticação, autorização e responsabilidade. A autenticação consiste em verificar se o usuário é, de fato, um usuário que está registrado no sistema. A autorização define as

permissões, baseadas no que o usuário é, possui ou ambos, que o usuário possui para lidar com o recurso solicitado. Por fim, as ações do usuário são registradas, realizando a etapa de responsabilidade do usuário pelas ações tomadas. As quatro vulnerabilidades são *Insufficiently Protected Credentials* (Credenciais insuficientementes protegidas), *Missing Authentication for Critical Function* (Falta de autenticação para funções críticas), *Missing Authorization* (Falta de autorização) e *Incorrect Authorization* (Autorização incorreta).

## 3.5 Sistemas Operacionais Reconfiguráveis

### 3.5.1 Panorama Geral

A computação reconfigurável surge a partir da necessidade de obter maior desempenho em determinadas aplicações, seja do ramo científico, econômico ou de qualquer área. No artigo de (BONDALAPATI; PRASANNA, 2002) é citado um exemplo de aplicação para a computação reconfigurável. Nas operações de filtragem, um conjunto de pixels vizinhos são utilizados para adaptar o valor de cada pixel. A janela de pixels da vizinhança é aplicada a todos os pixels da imagem. O grande número de pixels na imagem e a dependência de dados locais é uma excelente oportunidade para explorar o paralelismo em *hardware* reconfigurável. A imagem pode ser dividida e cada valor pode ser processado independentemente em paralelo por unidades de processamento dedicadas à aplicação. As placas de vídeo dedicadas são utilizadas para fins semelhantes atualmente, como mineração de criptomoedas, treinamento de inteligências artificiais avançadas etc., porém não possuem a possibilidade de serem reconfiguradas, o algoritmo deve ser projetado para a arquitetura em questão.
Os computadores de uso geral são projetados a partir de uma única arquitetura e conjunto de instruções limitado (SKLIAROVA; FERRARI, 2003). Os desenvolvedores de baixo nível projetam um compilador e/ou interpretador, a linguagem de programação para a arquitetura e o conjunto de instruções em questão e então o programador de alto nível pode criar uma gama de aplicações. Os computadores de arquitetura e conjunto de instruções fixo apresentam grande flexibilidade, visto que os programadores têm a capacidade de resolver uma grande quantidade de problemas em diversas áreas. Porém se, em vez de modificar a aplicação, fosse possível modificar o hardware do dispositivo, as aplicações resolveriam problemas com maior eficiência. Por exemplo, um problema X precisa de um tempo T e um problema Y precisa de um tempo 2T para ser resolvido em um computador convencional baseado na arquitetura de *Von Neumann* e *Harvard*. Já um *ASIC* consegue resolver o mesmo problema X em um tempo igual a T/8, porém não conseguiria resolver o problema Y. Se um dispositivo reconfigurável fosse empregado nessa situação, ele poderia ser configurado para obter o desempenho de um *ASIC* específico para o problema X e também para o problema Y, obtendo a maior eficiência na resolução dos dois problemas.
No artigo de (ECKERT DOMINIK MEYER; KLAUER, 2016), são citados sistemas operacionais com o foco em computação reconfigurável. Brebner (1996) foi um dos primeiros sistemas operacionais projetados para uma *FPGA* com a abordagem do termo *"hardware virtual"*, demonstrando a semelhança do *hardware* com a memória virtual e suas possibilidades de organização. O trabalho de (COMPTON et al., 2000) propôs soluções para problemas particionamento, como fragmentação e sobrecarga devido ao tempo para realocação. *OS4RC (2001)*, foi um projeto de um sistema operacional reconfigurável discutido em (WIGLEY; KEARNEY, 2001a). O projeto ficou apenas na idealização e discussão teórica. *HybridOS* é introduzido por (KELM; LUMETTA, 2008) como uma extensão para os sistemas operacionais e computadores tradicionais. O HybridOS é um conjunto de ferramentas criadas para utilizar *FPGAs* como unidades aceleradores para aplicações no sistema operacional *GNU/Linux*. O objetivo do projeto é fornecer uma estrutura para desenvolvedores criarem aplicações de

propósito geral e reduzir a dificuldade de mapear aplicativos em um modelo *CPU*/acelerador, como uma *API*. *LEAP (2011)* (Ambiente de latência intensiva para programação de aplicativos) foi desenvolvido em cooperação entre a *Intel* e o *MIT*. O *LEAP* é um sistema operacional para *FPGAs* que possui camadas de abstração para dispositivos, entrada e saída padrão e serviços para gerenciamento de memória. A idéia por trás do ambiente de latência intensiva é separar a comunicação e a computação, de modo que a camada de comunicação seja insensitiva às latências das camadas subsequentes. O sistema fornece uma interface extensível de gerenciamento de recursos para compilação em tempo real para *FPGAs* compatíveis (FLEMING; ADLER, 2016). *RIFFA (2012)* é um framework de integração reutilizável para *FPGAs*. O *framework* fornece comunicação e sincronização para *FPGAs* por meio de interfaces simples para software e hardware. O conceito chave do *RIFFA* é utilizar *FPGAs* como aceleradores, a conexão *PCI Express* é utilizada para realizar a ligação entre o barramento de sistema da *CPU* e as unidades aceleradoras (*FPGAs*). O *framework RIFFA* foi atualizado e novas versões foram desenvolvidas posteriormente, incorporando novas funcionalidades e dispositivos compatíveis (JACOBSEN et al., 2015). *OS4RS* (MIGNOLET et al., 2003) é um projeto de sistema operacional que permite o reescalonamento dinâmico de tarefas entre a *CPU* e as unidades lógicas dedicadas, de modo que não seja tarefa do programador definir onde a aplicação irá ser processada. O *OS4RS* também possui uma camada de abstração de *hardware* que proporciona uma interface simples para as unidades lógicas reconfiguráveis. *Hybrid Threads* foi introduzido em (ANDREWS et al., 2004) como um modelo de arquitetura que une a programação padrão com os dispositivos reconfiguráveis. O projeto apresenta um sistema operacional que possui camadas de abstração para lidar com o modelo de *threads*. A camada de *hardware* lida com threads estáticas, o que não permite a reconfiguração dinâmica em tempo de execução das unidades lógicas reconfiguráveis. Novas implementações foram publicadas posteriormente, funcionalidades como preemptividade de prioridade, *mutexes* e semáforos são implementadas no sistema (ANDREWS et al., 2005). *BORPH* é um sistema operacional baseado em *UNIX* desenvolvido para computadores reconfiguráveis baseados em *FPGAs*. *BORPH* fornece suporte de *kernel* para aplicações FPGA semelhante à forma que o sistema operacional convencional fornece para aplicações em *software*. Por um lado, o *BORPH* gerencia recursos reconfiguráveis de um dispositivo reconfigurável, assim como outros recursos do processador, como tempo de *CPU* e memória, alocando aos aplicativos do usuário, conforme necessário. Por outro lado, o *BORPH* isola a complexidade do usuário, como detalhes de baixo nível sobre o sistema, para que o usuário possa se concentrar exclusivamente no desenvolvimento de aplicativos (SO; BRODERSEN, 2007). *ReconOS* foi introduzido em (LUBBERS; PLATZNER, 2007) e recebeu contribuições posteriormente (HAPPE; TRABER; KELLER, 2015). O *ReconOS* é um sistema operacional para *FPGAs* com suporte a threads de *software* e *hardware*, baseado no sistema operacional de tempo real para dispositivos embarcados *eCos*. *CAP-OS* (GÖHRINGER et al., 2011) é um sistema operacional para *hardware* reconfigurável, mais especificamente *RAMPSoC*, com o foco em lidar com alterações de *hardware* em tempo real e comunicação, escalonamento dinâmico, mapeamento de tarefas de *hardware* e alocação de recursos.
Os projetos citados acima apresentam um resumo das obras que introduziram questões relacionadas a sistemas operacionais para dispositivos reconfiguráveis, alguns dos projetos serão abrangidos detalhadamente na seção seguinte.

### 3.5.2 BORPH

O *BORPH* é um sistema operacional voltado para sistemas reconfiguráveis. Consiste em uma versão do *kernel Linux* que faz o tratamento de um dispositivo reconfigurável como uma *CPU*. O sistema traz um conceito de processo de *hardware*, que consiste em um projeto de *hardware*

para ser executado em uma *FPGA*, por exemplo, se comportando igual a um programa normal de usuário (SO; BRODERSEN, 2007). O *BORPH* ainda oferece suporte ao sistema de arquivos gerais em tempo de execução para processos de *hardware* como se fossem *softwares*. Isso permite a comunicação entre *hardware* e *software*, e cria um sistema de arquivos virtual para permitir o acesso a memórias e registradores definidos no dispositivo reconfigurável (HAMILTON; SO, 2010). A implementação do *BORPH* no artigo (SO; BRODERSEN, 2007) foi realizada no *BEE2*, uma plataforma reconfigurável dedicada a processamento de sinais desenvolvida por volta de 2003 no *Berkeley Wireless Research Center*. A plataforma é composta por 5 *Xilinx Virtex-2pro*, sendo a arquitetura da CPU *PowerPC*. Existe também a implementação na plataforma *ROACH*, uma evolução para o *BEE2*, portanto com a mesma linha de *FPGAs* da *Xilinx*.
O projeto do *BORPH* não apresenta modificações há mais de dez anos, sendo o *RHINO* (WINBERG; LANGMAN; SCOTT, 2011) a última atualização a utilizar o *BORPH*. Alguns endereços da *web* que demonstravam o processo de instalação e a página oficial na plataforma da Universidade de *Berkeley* estão inativados. De forma oficial, é difícil encontrar os arquivos do sistema e o processo de instalação. O autor do trabalho (HAMILTON; SO, 2010) efetuou a portabilidade do sistema *BORPH* para a plataforma *netFPGA* e disponibilizou em seu *GitHub* os arquivos do *BORPH* e o processo de instalação, sendo uma possibilidade para encontrar os arquivos do *BORPH*. Diante do que foi exposto, existe a possibilidade de efetuar a instalação do *BORPH* em outras arquiteturas, visto que utiliza o núcleo *Linux*. Porém esse trabalho ainda não foi realizado.

### 3.5.3 FENIKS

O *FENIKS* é um sistema operacional para computação reconfigurável desenvolvido por pesquisadores da *Microsoft* (ZHANG et al., 2017). O intuito do sistema operacional é facilitar a utilização de *FPGAs* nos centros de dados e colaborar com a computação em nuvem. O FENIKS apresenta uma separação entre o nível de sistema operacional, em ordem de favorecer os desenvolvedores por meio de abstrações da comunicação e configuração interna do *hardware*, e o nível de configuração dos aceleradores. Segundo os autores, a separação clara entre os dois níveis da estrutura facilita a manutenção da lógica reconfigurável e das interfaces para os provedores de nuvem. Outro ponto interessante do sistema é a possibilidade de permitir que diversos aplicativos utilizem uma única *FPGA* através da alocação de recursos e também que vários *FPGAs* trabalhem junto em uma única carga de trabalho. A característica citada anteriormente é de grande importância para a exploração do paralelismo de processamento na computação em nuvem. Os autores do *FENIKS* realizaram uma observação importante sobre a implementação de sistemas operacionais reconfiguráveis como os citados e explorados anteriormente. A maioria dos sistemas apresentam uma abordagem onde a *FPGA* e a *CPU* são embarcadas em uma mesma placa. Portanto, a frequência da comunicação entre a *CPU* e a *FPGA* pode ser tão excessiva como a comunicação entre os núcleos de um mesmo processador, o que ocasionaria problemas em centros de dados com *FPGAs* nem sempre em conjunto com a *CPU*. Esse ponto é levado em conta no *FENIKS*. O artigo (HUANG; HSIUNG, 2009) apresentou uma abordagem de virtualização dos dispositivos reconfiguráveis. No *FENIKS*, a abordagem do *hardware* reconfigurável permite a reconfiguração do *hardware* em vários dispostivos aceleradores virtualizados. Os testes apresentados no artigo apresentam um exemplo para compressão e outro com uma aplicação para *firewall*. Ambos foram implementados em apenas um *FPGA*, sendo a área dividida para as duas tarefas. Em ambos os testes, os resultados apresentaram desempenho superior no *FPGA* em relação ao mesmo tipo de aplicativo sendo executado em um processador de propósito geral. Não existem outros artigos, até o momento em que esse artigo está sendo escrito, explorando o sistema, visto que é

um projeto fechado da *Microsoft* e não é disponibilizado para o público.

### 3.5.4 ReconOS

O *ReconOS* é um sistema operacional para computação reconfigurável, oferecendo um modelo de programação *multithread* unificado, além do gerenciamento de *threads*, executando threads de software e threads mapeados para *hardware* reconfigurável. É ideal para desenvolvedores com conhecimentos em sincronização de processos para desenvolver aplicativos híbridos com encadeamentos sendo executados na *CPU* (*software*) e no dispositivo reconfigurável (*hardware*) (LÜBBERS; PLATZNER, 2010).
O *ReconOS* em sua primeira versão foi executado a partir do sistema operacional *eCos*, em um processador *PowerPC* embarcado em uma *FPGA* da *Xilinx*. As *FPGAs* eram o *Virtex-2 Pro* e o *Virtrex-4*. Para utilizar o *eCos*, segundo o autor, somente *FPGAs* da família *Virtrex* são suportados. A partir da segunda versão, o sistema recebeu suporte para ser executado em um sistema operacional de núcleo *Linux*. As últimas atualizações ofereceram suporte para a arquitetura *MicroBlaze* com sistema operacional de núcleo *Linux* ou *Xilkernel*, além da plataforma de arquitetura *ARM Xilinx Zynq*. Segundo os criadores do projeto, o *ReconOS* está sendo adaptado para a arquitetura x86 com suporte à aceleradores por meio barramento *PCIe* (LÜBBERS; PLATZNER, 2009). O *MicroBlaze* da *Xilinx* é um *soft processor* de 32/64 bits com arquitetura de conjunto de instruções *RISC*, isto é, é uma arquitetura de processador que pode ser totalmente sintetizada por meio de linguagem de descrição de *hardware* a partir de um *software* de síntese e possui conjunto de instruções reduzido. Como é um dispositivo da *Xilinx*, o *MicroBlaze* só pode ser sintetizado por meio do *software* Vivado da *Xilinx* (XILINX, 2021). Uma alternativa de *soft processor* da *Intel* é o *Nios II*, com registradores de 32 bits, arquitetura de conjunto de instruções *RISC* e sintetizado a partir do *software Quartus Prime* (INTEL, 2021).
Para instalar o sistema operacional e suas dependências, o site oficial do *ReconOS* disponibiliza dois métodos de instalação. O primeiro é o modo fácil que, a partir de um *shell script* em um sistema operacional *GNU/Linux*, efetua a instalação e configuração automática do ambiente. Existe o procedimento passo a passo tendo como referência a plataforma *Zynq* e com todos os arquivos disponíveis para *download* (RECONOS, 2021). O *ReconOS* possui suporte para a arquitetura *ARM*, *PowerPC* e *MicroBlaze* por meio do sistema operacional *GNU*/*Linux*. A partir disso, existe a possibilidade de expandir os dispositivos suportados como, por exemplo, instalá-lo em um *Raspberry Pi* ou *Arduino* (*ARM*), e realizar as interfaces de sistema operacional com uma *FPGA*. Quanto ao *MicroBlaze*, se identificadas as propriedades da arquitetura, pode ser possível a portabilidade do *ReconOS* para o *Nios II* da *Intel*, visto que ambos partilham bastantes características de microarquitetura (*endianness*, tamanho dos registradores, arquitetura de conjunto de instruções baseada em *RISC* etc). O artigo de (SANAL; PINALKUMAR, 2018) demonstra o *ReconOS* sendo aplicado em um contexto de processamento de imagem. Uma barreira para o desempenho de aplicativos desse tipo é a execução sequencial que, segundo o autor, pode ser contornada com o uso da FPGA. Para o projeto, foi utilizado o sistema ReconOS/Linux e a placa de desenvolvimento *Zynq-7000 Zybo*, que possui um *CPU ARM* e uma *FPGA*, sendo uma plataforma híbrida. A plataforma utilizada em questão possui semelhanças com diversos dispositivos, o que reforça ainda mais a possibilidade de portabilidade do *ReconOS* para um dispositivo como o *Arduino MKR Vidor 4000* por exemplo. Quanto aos resultados, não apresentaram apenas grande aumento de desempenho, como também apresentaram diminuição da complexidade de implementação a partir dos *threads* de *hardware* com relação à implementação por *software*.

# 4 DESCRIÇÃO DAS ATIVIDADES DESENVOLVIDAS

O trabalho foi realizado por meio de pesquisas bibliográficas com base em estudos já concluídos sobre o tema. Deste modo, a mesma foi baseada na análise da literatura já publicada em forma de livros, artigos e literatura acadêmica (teses, dissertações, relatórios, trabalhos apresentados em congressos etc.).

A possibilidade de implementação de um Sistema Operacional reconfigurável em placas de desenvolvimento disponíveis no mercado e capazes de serem adquiridas pela universidade foi verificada de acordo com informações presentes em artigos e sites dos criadores.

Como citado anteriormente, as *FPGAs* são dispositivos típicos utilizados em computação reconfigurável, para isso serão utilizadas placas de desenvolvimento com *FPGAs* para a futura adaptação e implementação de Sistemas Operacionais reconfiguráveis expostos no trabalho.

# 5 RESULTADOS OBTIDOS E ANÁLISE

Os artigos Wigley e Kearney (2001) e Wigley e Kearney (2002) identificam pontos importantes para o sistema operacional voltado para computação reconfigurável. Carregamento de aplicações, particionamento, gerenciamento de memória, escalonamento, proteção e entrada/saída, e comunicação e sincronização são partes importantes que compõem um sistema operacional, principalmente para a computação reconfigurável. A ideia da computação reconfigurável é trazer estas funções do sistema operacional para o nível de hardware, de forma que se obtenha uma solução mais eficiente.

Eckert et al. (2016) trazem as seguintes discussões sobre os tópicos citados no parágrafo anterior. O problema do particionamento nas lógicas reconfiguráveis pode ser resolvido internamente ao sistema operacional, a solução sugerida pelo autor traz a ideia da utilização de áreas reconfiguráveis dos dispositivos. A principal desvantagem desta solução está no surgimento de fragmentações internas. Para o carregamento de aplicações no sistema, os módulos reconfiguráveis devem atuar como processadores equivalentes que interagem com a CPU, isso é chamado de hw-application. O objetivo desta utilização é, como citado anteriormente, trazer a função de software para o nível de hardware, trazendo ganhos em termos de tarefas. Esta área está em constante pesquisa na área científica atual, a dificuldade de solução está na necessidade de uma comunicação e sincronização sofisticada.

No decorrer do trabalho, foram levantadas informações sobre diversos Sistemas Operacionais reconfiguráveis, porém os que demonstraram mais relevância e possibilidade de evolução foram o FENIKS, BORPH e ReconOS.

O FENIKS é um sistema operacional para computação reconfigurável desenvolvido por pesquisadores da Microsoft (ZHANG et al., 2017). Pouco se sabe sobre o sistema ainda por ser uma iniciativa privada da Microsoft, porém existem grandes possibilidades de ganhar mais atenção em um futuro próximo devido a necessidade de performance exigida nos servidores com o aumento de SaaS, IaaS e PaaS.

BORPH (Berkeley Operating System for ReProgrammable Hardware) é um sistema operacional para dispositivos reconfiguráveis desenvolvido na Universidade da Califórnia, Berkeley em 2007. A semântica UNIX é mantida no sistema BORPH (SO; BRODERSEN, 2007). Acreditava-se que o sistema BORPH, por se tratar de um trabalho de Berkeley, apresentaria evolução durante o tempo. Porém o sistema não apresentou evoluções até o presente momento, tampouco possui seus portais com acesso aos arquivos e procedimentos de instalação. Espera-se que o projeto apresente novas colaborações futuramente.

O ReconOS é um sistema operacional para dispositivos reconfiguráveis baseado, primeiramente, no eCos (Embedded Configurable Operating System) com o foco em oferecer um ambiente de programação unificado e portabilidade de aplicativos para dispositivos reconfiguráveis (LUBBERS; PLATZNER, 2007). Para dar suporte a sistemas em tempo real, a abordagem RTOS oferece escalonamento preemptivo dinâmico baseado em prioridade para threads, reduz as latências de interrupção, tempo de execução limitado para chamadas do sistema e são altamente reconfiguráveis para lidar com escassez de memória. Em alguns artigos, o ReconOS é citado como um framework, visto que necessita de um núcleo de outro sistema operacional para ser executado. Desde sua primeira versão em 2006, o ReconOS evoluiu e apresentou 4 novas versões até o momento em que este artigo está sendo escrito, incluindo suporte ao núcleo Linux e outros novos recursos (RECONOS, 2021).

Pretende-se realizar a futura implementação do sistema em uma arquitetura ARM compatível. É ideal para desenvolvedores com conhecimentos em sincronização de processos para desenvolver aplicativos híbridos com encadeamentos sendo executados na CPU (software) e no dispositivo reconfigurável (hardware) (LÜBBERS; PLATZNER, 2010).

# 6 CONCLUSÕES

Na área de Computação reconfigurável ainda existem muitos desafios, como a síntese de *hardware* ser um processo lento em relação à síntese de *software*, dificultando o crescimento da computação reconfigurável (ECKERT et al., 2016).

Realizar a reconfiguração de uma *FPGA* para se comportar de acordo com a tarefa que a requisite é um processo manual que visa ser automatizado por meio dos Sistemas Operacionais reconfiguráveis. Com isso, caso um programador deseje realizar o desenvolvimento de uma aplicação para este paradigma, não será necessário conhecimentos profundos acerca de síntese de *hardware* e outras áreas do conhecimento. A ideia é que o paradigma se torne consolidado e com um ecossistema de ferramentas que o torne útil para aplicações dos usuários. Com isso, o desenvolvedor pode se concentrar no algoritmo da aplicação, assim como acontece com os computadores comuns atualmente.

Por se tratar de uma pesquisa com o intuito de estudar o estado da arte sobre Sistemas Operacionais reconfiguráveis, foram levantados diversos trabalhos anteriores que discutem o tema. A partir disso, obteve-se a conclusão de que existem poucos trabalhos nacionalmente desenvolvidos, sendo o único citado o *EPOS* (FRÖHLICH, 2001). Além disso, foi possível obter informações necessárias para a futura implementação do *ReconOS*.

A pesquisa levantou informações sobre o estado da arte em ordem de construir uma base para o desenvolvimento de projetos na área de Sistemas Operacionais reconfiguráveis. Assim foi possível definir um caso de estudo que possui características que possibilitam a implementação e desenvolvimento na área. O Sistema Operacional reconfigurável que apresentou evoluções e possui propriedades de arquitetura e APIs que tornam a implementação possível para a pesquisa é o *ReconOS*. Espera-se que em um futuro próximo a computação reconfigurável se torne relevante não só para um público específico, mas também para o público em geral.

## REFERÊNCIAS

SKLIAROVA, I.; FERRARI, A. B. Introdução à computação reconfigurável. Portugal: Revista do DETUA, 2003.

DEITEL, H.; DEITEL, P.; CHOFFNES, D. Sistemas Operacionais. São Paulo: Pearson Prentice Hall, 2005. (3. ed.).

TANENBAUM, A. S.; BOS, H. Sistemas Operacionais Modernos. São Paulo: Pearson Education do Brasil, 2016. (4. ed.).

ECKERT, M. et al. Operating system concepts for reconfigurable computing: review and survey. International Journal of Reconfigurable Computing, Hindawi, v. 2016, 2016.

BARIK, R.; SINHA, A. Taxonomy of reconfigurable computing and operating system. 2016.

WIGLEY, G.; KEARNEY, D. The first real operating system for reconfigurable computers. In: IEEE. Proceedings 6th Australasian Computer Systems Architecture Conference. ACSAC 2001. [S.l.], 2001. p. 130–137.

ADETOMI, A. A. Dynamic reconfiguration frameworks for high-performance reliable real-time reconfigurable computing. The University of Edinburgh, 2019.

KHOKHAR, A. A. et al. Heterogeneous computing: Challenges and opportunities. Computer, IEEE, v. 26, n. 6, p. 18–27, 1993.

KAELI, D. et al. Heterogeneous Computing with OpenCL 2.0. Elsevier Science, 2015. ISBN 9780128016497. Disponível em: <https://books.google.com.br/books?id=oER9AwAAQBAJ>.

STRINGHINI, D.; GONÇALVES, R. A.; GOLDMAN, A. Introdução à computação heterogênea. In: Anais da XXXI Jornada de Atualização em Informática do XXXII Congresso da Sociedade Brasileira de Computação [Internet]. [S.l.: s.n.], 2012. p. 16–19.

BRODTKORB, A. R. et al. State-of-the-art in heterogeneous computing. Scientific Programming, IOS Press, v. 18, n. 1, p. 1–33, 2010.

GOLDSCHMIDT, R.; PASSOS, E. Data mining: um guia Prático. Elsevier Editora, 2005. ISBN 9788535218770. Disponível em: <https://books.google.com.br/books?id=JJYHNrREwyEC>.

MACIEL, D. S.; DUARTE, M. Análise e testes de algoritmos utilizando sistemas de arquiteturas híbridas cpu/gpu. Revista Eletrônica eF@ tec, v. 6, n. 1, p. 17–17, 2016.

ANDREWS, D.; NIEHAUS, D.; ASHENDEN, P. Programming models for hybrid cpu/fpga chips. Computer, IEEE, v. 37, n. 1, p. 118–120, 2004.

AGNE, A. et al. Reconos: An operating system approach for reconfigurable computing. IEEE Micro, IEEE, v. 34, n. 1, p. 60–71, 2013.

SHARMA, G.; KUMAR, A.; SHARMA, V. Windows operating system vulnerabilities. International Journal of Computing and Corporate Research, v. 1, n. 3, 2011.

NETMARKETSHARE. Operating System Market Share. 2020. Disponível em: <https://netmarketshare.com/operating-system-market-share.aspx?options=%7B%22filter%22%3A%7B%22%24and%22%3A%5B%7B%22deviceType%22%3A%7B%22%24in%22%3A%5B%22Desktop%2Flaptop%22%5D%7D%7D%5D%7D%2C%22dateLabel%22%3A%22Trend%22%2C%22attributes%22%3A%22share%22%2C%22group%22%3A%22platform%22%2C%22sort%22%3A%7B%22share%22%3A-1%7D%2C%22id%22%3A%22platformsDesktop%22%2C%22dateInterval%22%3A%22Monthly%22%2C%22dateStart%22%3A%222019-11%22%2C%22dateEnd%22%3A%222020-10%22%2C%22segments%22%3A%22-1000%22%7D>.

CVE DETAILS. Top 50 Products By Total Number Of "Distinct"Vulnerabilities. 2019. Disponível em: <https://www.cvedetails.com/top-50-products.php?year=0>. Acesso em: 28 jan. 2021.

THE MITRE CORPORATION. Top 50 Products By Total Number Of "Distinct"Vulnerabilities. 2021. Disponível em: <https://cwe.mitre.org/>. Acesso em: 04 fev. 2021.

BONDALAPATI, K.; PRASANNA, V. K. Reconfigurable computing systems. Proceedings of the IEEE, IEEE, v. 90, n. 7, p. 1201–1217, 2002.

ECKERT DOMINIK MEYER, J. H. M.; KLAUER, B. Operating system concepts for reconfigurable computing: Review and survey. Hindawi Publishing Corporation International Journal of Reconfigurable Computing Volume 2016, Article ID 2478907, 11 pages, p. 1–11, 2016.

COMPTON, K. et al. Configuration relocation and defragmentation for FPGAs. [S.l.], 2000.

WIGLEY, G.; KEARNEY, D. The development of an operating system for reconfigurable computing. In: IEEE. The 9th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM'01). [S.l.], 2001. p. 249–250.

KELM, J. H.; LUMETTA, S. S. Hybridos: runtime support for reconfigurable accelerators. In: Proceedings of the 16th international ACM/SIGDA symposium on Field programmable gate arrays. [S.l.: s.n.], 2008. p. 212–221.

FLEMING, K.; ADLER, M. The leap fpga operating system. In: FPGAs for Software Programmers. [S.l.]: Springer, 2016. p. 245–258.

JACOBSEN, M. et al. Riffa 2.1: A reusable integration framework for fpga accelerators. ACM Transactions on Reconfigurable Technology and Systems (TRETS), ACM New York, NY, USA, v. 8, n. 4, p. 1–23, 2015.

MIGNOLET, J.-Y. et al. Infrastructure for design and management of relocatable tasks in a heterogeneous reconfigurable system-on-chip. In: IEEE. 2003 Design, Automation and Test in Europe Conference and Exhibition. [S.l.], 2003. p. 986–991.

ANDREWS, D. et al. Programming models for hybrid fpga-cpu computational components: a missing link. IEEE Annals of the History of Computing, IEEE Computer Society, v. 24, n. 04, p. 42–53, 2004.

ANDREWS, D. et al. hthreads: A hardware/software co-designed multithreaded rtos kernel. In: IEEE. 2005 IEEE Conference on Emerging Technologies and Factory Automation. [S.l.], 2005. v. 2, p. 8–pp.

SO, H. K.-H.; BRODERSEN, R. W. Borph: An operating system for fpga-based reconfigurable computers. [S.l.]: University of California, Berkeley, 2007.

LUBBERS, E.; PLATZNER, M. Reconos: An rtos supporting hard-and software threads. In: IEEE. 2007 International Conference on Field Programmable Logic and Applications. [S.l.], 2007. p. 441–446.

HAPPE, M.; TRABER, A.; KELLER, A. Preemptive hardware multitasking in reconos. In: SPRINGER. International Symposium on Applied Reconfigurable Computing. [S.l.], 2015. p. 79–90.

GÖHRINGER, D. et al. Operating system for runtime reconfigurable multiprocessor systems. International Journal of Reconfigurable Computing, Hindawi, v. 2011, 2011.

HAMILTON, B. K.; SO, H. K.-H. Borph: Operating system support on the netfpga platform. In: North American NetFPGA Developers Workshop. [S.l.: s.n.], 2010.

WINBERG, S.; LANGMAN, A.; SCOTT, S. The rhino platform: charging towards innovation and skills development in software defined radio. In: Proceedings of the South African Institute of Computer Scientists and Information Technologists Conference on Knowledge, Innovation and Leadership in a Diverse, Multidisciplinary Environment. [S.l.: s.n.], 2011. p. 334–337.

ZHANG, J. et al. The feniks fpga operating system for cloud computing. In: Proceedings of the 8th Asia-Pacific Workshop on Systems. [S.l.: s.n.], 2017. p. 1–7.

HUANG, C.-H.; HSIUNG, P.-A. Hardware resource virtualization for dynamically partially reconfigurable systems. IEEE Embedded Systems Letters, IEEE, v. 1, n. 1, p. 19–23, 2009.

LÜBBERS, E.; PLATZNER, M. Reconos: An operating system for dynamically reconfigurable hardware. In: Dynamically Reconfigurable Systems. [S.l.]: Springer, 2010. p. 269–290.

LÜBBERS, E.; PLATZNER, M. Reconos: Multithreaded programming for reconfigurable computers. ACM Transactions on Embedded Computing Systems (TECS), ACM New York, NY, USA, v. 9, n. 1, p. 1–33, 2009.

XILINX. MicroBlaze. 2021. Disponível em: <https://www.xilinx.com/products/intellectual-property/microblazecore.html>. Acesso em: 14 julho de 2021.

INTEL. Nios II. 2021. Disponível em: <https://www.intel.com.br/content/www/br/pt/products/programmable/processor/nios-ii.html>. Acesso em: 14 julho de 2021.

RECONOS. ReconOS - History. 2021. Disponível em: <http://www.reconos.de/about/history/>. Acesso em: 28 abril de 2021.

SANAL, S.; PINALKUMAR, J. Multithreaded image processing using reconos on reconfigurable computing system. In: IEEE. 2018 International Conference on Emerging Trends and Innovations In Engineering And Technological Research (ICETIETR). [S.l.], 2018. p. 1–5.

## APÊNDICE A – ASSINATURA DO ALUNO E DO ORIENTADOR

Este relatório apresenta o registro da pesquisa intitulada “Sistemas Operacionais Reconfiguráveis”, desenvolvida no período de Setembro/20 a Agosto/21, sob a orientação do Prof. Otávio de Souza Martins Gomes, tendo em vista as orientações estipuladas pelo Edital 02/2020.

Itajubá, 30 de setembro de 2021.

Felipe dos Santos – Aluno de IC

Otávio de Souza Martins Gomes - Orientador