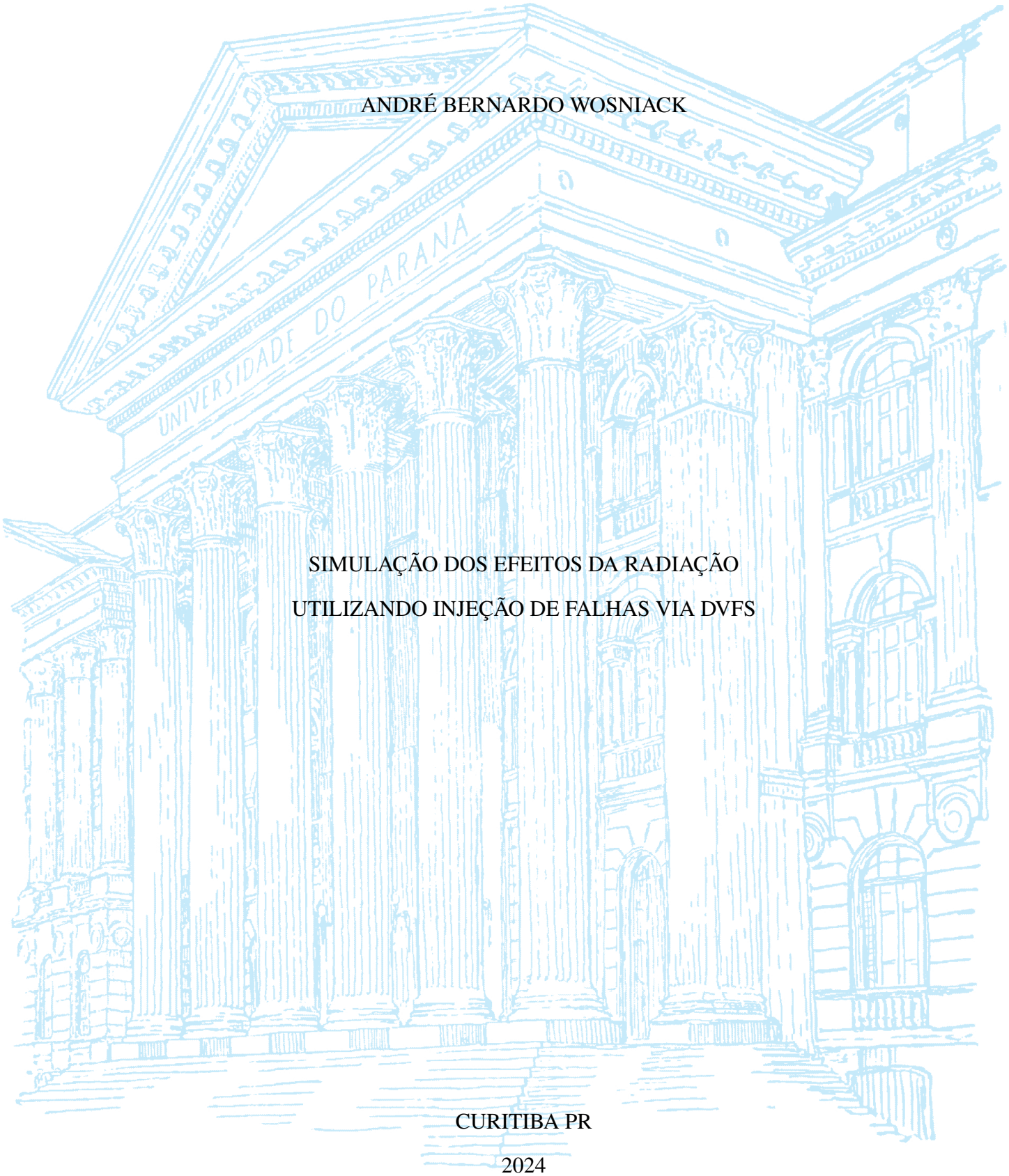
UNIVERSIDADE FEDERAL DO PARANÁ

ANDRÉ BERNARDO WOSNIACK



SIMULAÇÃO DOS EFEITOS DA RADIAÇÃO UTILIZANDO INJEÇÃO DE FALHAS VIA DVFS

CURITIBA PR

2024

ANDRÉ BERNARDO WOSNIACK

SIMULAÇÃO DOS EFEITOS DA RADIAÇÃO UTILIZANDO INJEÇÃO DE FALHAS VIA DVFS

Trabalho apresentado como requisito parcial à conclusão do Curso de Bacharelado em Ciência da Computação, Setor de Ciências Exatas, da Universidade Federal do Paraná.

Área de concentração: *Computação*.

Orientador: Marcos Antonio Zanata Alves.

CURITIBA PR

2024

Universidade Federal do Paraná
Setor de Ciências Exatas
Curso de Ciência da Computação

Ata de Apresentação de Trabalho de Conclusão de Curso 2

Título do Trabalho: SIMULAÇÃO DOS EFEITOS DA RADIAÇÃO UTILIZANDO INJEÇÃO DE FALHAS VIA DVFS

Autor(es):

GRR20171614 Nome: ANDRÉ BERNARDO WOSNIACK

Apresentação: Data: 31/07/2024 Hora: 15h30 Local: Laboratório HIPES

Orientador: Marco Antonio Zanata Alves

Membro 1: Daniel Alfonso Gonçalves de Oliveira

Membro 2: Rodrigo Machniewicz Sokulski

(nome) (assinatura)

AVALIAÇÃO – Produto escrito		ORIENTADOR	MEMBRO 1	MEMBRO 2	MÉDIA
Conteúdo	(00-40)				
Referência Bibliográfica	(00-10)				
Formato	(00-05)				
AVALIAÇÃO – Apresentação Oral					
Domínio do Assunto	(00-15)				
Desenvolvimento do Assunto	(00-05)				
Técnica de Apresentação	(00-03)				
Uso do Tempo	(00-02)				
AVALIAÇÃO – Desenvolvimento					
Nota do Orientador	(00-20)		**********	**********	
NOTA FINAL		**********	**********	**********	9.0,00

Os pesos indicados são sugestões.

Conforme decisão do colegiado do curso de Ciência da Computação, a entrega dos documentos comprobatório de trabalho de Conclusão de Curso 2 deve deve respeitar os seguintes procedimentos: o orientador deve abrir um processo no Sistema Eletrônico de Informações (SEI – UFPR); Selecionar o tipo: *Graduação: Trabalho Conclusão de Curso*; informar os interessados: nome do aluno e o nome do orientador; anexar esta ata escaneada e a versão final do PDF da monografia do aluno; Tramitar o processo para CCOMP (Coordenação de Ciência da Computação).

AGRADECIMENTOS

Primeiramente, gostaria de agradecer ao meu professor e orientador, Marco Antonio Zanata Alves, pela paciência, orientação e principalmente apoio durante todo o desenvolvimento deste trabalho. Seus conselhos e conhecimento foram essenciais para a conclusão deste projeto.

Agradeço também ao professor Daniel Oliveira pela disponibilidade e suporte oferecido para a realização da pesquisa.

Aos meus amigos e familiares, especialmente minha irmã Isabella Wosniack e meus pais Edilton e Susana Wosniack. Minha eterna gratidão pelo suporte e incentivo durante esta jornada.

Muito obrigado a todos!

RESUMO

A incessante busca por mais desempenho operacional nos equipamentos eletrônicos resulta, entre outros aspectos, na miniaturização dos componentes, a qual permite dispositivos menores e mais eficientes. No entanto, à medida que os componentes eletrônicos ficam mais compactos, estes também se tornam mais susceptíveis à influência de fenômenos externos, como a radiação ionizante, que pode ocasionar falhas e mau funcionamento do dispositivo. Um dos métodos utilizados para simular os impactos da radiação ionizante em dispositivos eletrônicos é através de experimentos conduzidos em aceleradores de partículas, os quais são capazes de incidir sobre o dispositivo centenas de anos de exposição natural a radiação em algumas horas. Devido ao número limitado de aceleradores de partículas disponíveis no mundo, pode ser necessário um grande deslocamento para realização de testes. Além disso, a construção e manutenção dos aceleradores possui um elevado custo financeiro, qual é transmitido para a realização dos testes. Este trabalho tem como objetivo simular falhas ocasionadas por radiação ionizante através da tecnologia *Dynamic Voltage and Frequency Scaling* (*DVFS*). Para isso, foi mapeado o comportamento dos processadores de dois *raspberry PI* e, em seguida, foram definidas frequências instáveis para a execução de um *benchmark*. Após comparar os dados obtidos com as saídas do mesmo *benchmark* executado em aceleradores submetidos a um acelerador de partículas, foi observado que as falhas geradas não seguem o mesmo padrão, sendo maior a probabilidade de erros aleatórios ocorrerem em um processador instável do que nos aceleradores submetidos a radiação ionizante. Além disso, processadores instáveis geram a maior parte de seus erros em poucos elementos, comportamento não observado no caso de aceleradores atingidos pela radiação. A tecnologia *DVFS* pode ser utilizada para injeção de falhas via *software* sem a sobrecarga adicional, se tornando uma ferramenta capaz de melhorar a robustez de um código.

Palavras-chave: DVFS, Radiação Ionizante, Raspberry PI, Injeção de Falhas.

ABSTRACT

The ceaseless search for more operational performance in electronic devices results, among other aspects, in the miniaturization of components, which allows for smaller and more efficient devices. However, as electronic components become more compact, they also become more susceptible to the influence of external phenomena, such as ionizing radiation, which can cause device failures and malfunctions. One of the methods used to simulate the impacts of ionizing radiation on electronic devices is through experiments conducted in particle accelerators, which are capable of focusing on the device hundreds of years of natural exposure to radiation in a few hours. Due to the limited number of particle accelerators available in the world, a large displacement may be necessary to carry out tests. Furthermore, the construction and maintenance of accelerators has a high financial cost, which is transferred to the tests. This paper aims to simulate failures caused by ionizing radiation using *Dynamic Voltage and Frequency Scaling* (*DVFS*) technology. To achieve this, the behavior of two *raspberry PI* processors was mapped and then unstable frequencies were defined for executing a *benchmark*. After comparing the data obtained with the outputs of the same *benchmark* executed on accelerators subjected to a particle accelerator, it was observed that the failures generated do not follow the same pattern, with a greater probability of random errors occurring in an unstable processor than in accelerators subjected to ionizing radiation. Furthermore, unstable processors generate most of their errors in a few elements, a behavior not observed in the case of accelerators affected by radiation. *DVFS* technology can be used for fault injection via *software* without additional overhead, becoming a tool capable of improving the robustness of a code.

Keywords: DVFS, Ionizing Radiation, Raspberry PI, Fault Injection.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ACRÔNIMOS

DINF	Departamento de Informática
PPGINF	Programa de Pós-Graduação em Informática
UFPR	Universidade Federal do Paraná
DGEMM	Double-Precision Generic Matrix-Matrix Multiplication
DVFS	Dynamic Voltage and Frequency Scaling
CMOS	Complementary Metal-Oxide-Semiconductor
SRAM	Static Random Access Memory
GDB	GNU Debugger
HPC	High Performance Computing
GPU	Graphics Processing Unit
CPU	Central Processing Unit

SUMÁRIO

1 INTRODUÇÃO

Com o objetivo de melhorar o desempenho e diminuir custos energéticos, designers de *hardware* tem buscado cada vez mais miniaturizar componentes eletrônicos, fazendo com que a confiabilidade dos mesmos seja comprometida (Lakshminarayanan e Sriraam, 2014).

Dependendo de sua localização, dispositivos eletrônicos podem estar expostos a diversos tipos de radiação (partículas energéticas), com os efeitos dessa exposição variando entre perda temporária de dados, até uma falha mais severa com danos permanentes ao dispositivo. Esses efeitos dependem em sua maior parte do nível de radiação do ambiente e da tecnologia utilizada. Em ambientes de elevada radiação, como o espaço que possui maior ocorrência de raios cósmicos, muitas das tecnologias utilizadas são antigas pois necessitam de diversos testes, contribuindo com o conhecimento dos efeitos da exposição à radiação. Outra opção, utilizando de tecnologias mais recentes, é uma adequação específica capaz de suportar a radiação. Como consequência, essa modificação torna o produto mais caro, podendo inviabilizar o seu uso. No entanto, a demanda por um melhor desempenho e por uma construção mais rápida aumenta a pressão para utilizar tecnologias mais avançadas e recentes, na qual os efeitos da radiação podem não ser completamente conhecidos (Schrimpf e Fleetwood, 2004).

Dispositivos eletrônicos mais recentes, em sua maior parte, seguem o padrão observado e conhecido como lei de Moore, a qual afirma que a quantidade de transistores em uma mesma área dobra a cada dois anos, com o mesmo custo de fabricação (Schaller, 1997). O aumento na quantidade de transistores permite operações mais complexas em um único chip. No entanto, aumenta também a probabilidade de falhas ocasionadas pela radiação, que pode vir a depositar energia suficiente para inverter o estado lógico de um componente. Como exemplo, tem-se o caso de transistores utilizados na microeletrônica, que requerem menos energia para mudança de estado devido a menor tensão operacional.

Para transistores CMOS, a suscetibilidade de uma falha ocasionada pela radiação cresce exponencialmente conforme a tensão diminui e reduz linearmente conforme a área do dispositivo, assim a taxa de falhas tende a aumentar com a miniaturização dos dispositivos eletrônicos. De acordo com (Dixit e Wood, 2011), isso já ocorre para memórias de acesso aleatório estáticas (SRAM) de 40nm.

Na microeletrônica, os erros causados pela radiação ionizante são denominados *Single Event Effects* (SEE), no qual a energia depositada pela interação entre uma única partícula energética e o dispositivo eletrônico leva ao seu mau funcionamento. A forma mais comum de SEE é chamada *Single Event Upsets* (SEUs) e ocorre quando a energia depositada causa apenas uma inversão do estado lógico de um bit, podendo ser classificado como *soft error*. Neste tipo de erro, ocorre uma falha transiente na qual o erro só aparece uma vez e ao repetir a mesma operação o erro desaparece. Os erros também podem ser classificados como *Hard error*, no qual o dispositivo é permanentemente danificado (Schrimpf e Fleetwood, 2004).

DVFS (*Dynamic Voltage and Frequency Scaling*) é uma técnica de gerenciamento de energia presente na maioria dos processadores atuais, a qual pode alterar a velocidade do processador de acordo com as limitações de energia e temperatura disponíveis. Como exemplo, tem-se a Intel com a tecnologia *Turbo Boost* (Rotem et al., 2012), AMD com *Precision Boost*, e processadores ARM. Esta técnica faz uso de uma ligação entre o *software* e o *hardware* para controlar o consumo energético em tempo de execução, sendo o *software* capaz de tomar as decisões e utilizar componentes de *hardware* para definir a frequência e tensão do processador. Embora o uso de *DVFS* seja para otimizar o desempenho e consumo de energia,

esta técnica possibilita induzir falhas no processador quando utilizadas frequências ou tensões fora das definidas como seguras para o componente. (Tang et al., 2017) conseguiram por meio da tecnologia *DVFS* em um processador ARM escalar privilégios além de extrair chaves criptográficas.

Atualmente, um dos principais métodos para avaliar os efeitos da radiação ionizante em um dispositivo são experimentos em acelerador de partículas. O dispositivo a ser testado é atingido por uma radiação de forma controlada, de modo a simular anos de exposição a radiação natural em apenas algumas horas. Assim é possível compreender e adequar o dispositivo, programa ou protótipo a suas possíveis vulnerabilidades quando exposto à radiação. Os experimentos com aceleradores de partículas carregam diversas dificuldades, como o elevado custo financeiro, rigorosos protocolos devido à periculosidade dos testes, entre alguns outros fatores.

Este trabalho tem por objetivo simular falhas ocasionadas pela radiação ionizante através de *DVFS*. Para comparar as falhas, foram utilizados da localidade espacial dos erros e também de sua magnitude. Dessa forma, esperamos diminuir os custos da simulação de falhas atual, além de simplificar o processo.

2 BACKGROUND

Este capítulo tem por objetivo apresentar os principais tópicos referentes ao presente trabalho.

2.1 ERROS CAUSADOS POR RADIAÇÃO

Em 1978 foi apresentado um artigo introduzindo o conceito de *soft errors* como erros aleatórios, não recorrentes e de apenas um bit causados pela radiação. O artigo relatou *soft errors* em memórias DRAM 16-kb da série Intel 2107 causados por partículas *alpha* originadas do decaimento radioativo de impuridades de urânio e tório nos encapsulamentos. Foi a primeira publicação de *soft errors* detectados no nível do mar. Por ter sido um problema importante para a Intel, foram efetuadas investigações até a causa do problema. Após muito esforço, foi constatado que o encapsulamento cerâmico estava contaminado com impuridades radioativas através da água utilizada no processo de fabricação. A fábrica das embalagens havia sido construída rio abaixo de uma velha mina de urânio, onde a água havia sido contaminada (Nicolaidis, 2010).

Outro exemplo a se mencionar foi o problema conhecido como *Hera* da IBM. Em 1986 a IBM observou um aumento de falhas em suas memórias fabricadas na América, e memórias idênticas fabricadas na Europa não apresentavam esses erros. Conhecendo o caso da Intel de alguns anos atrás, trocaram as embalagens cerâmicas de modo que os *chips* produzidos na Europa utilizavam embalagens da América e vice-versa. Após isso, os erros continuaram a se manifestar somente nos *chips* produzidos na América, mostrando que a origem dos erros não estava na embalagem, e sim no próprio *chip*.

Em uma análise mais profunda, foi constatado que os chips possuíam uma alta radioatividade, mas ainda restava achar qual a causa do problema. Alguns meses depois foi encontrada a origem da contaminação, uma garrafa de ácido nítrico utilizada durante a manufatura do *chip*. Na fábrica de origem, a máquina responsável por limpar as garrafas utilizava um jato de ar ionizado para remover a eletrostática após a lavagem. O elemento radioativo responsável pela ionização do ar estava vazando aleatoriamente, devido ao selo da máquina ter sido recentemente trocado.

Após ter encontrado a causa, foram removidas as garrafas contaminadas da produção, assim resolvendo o problema. Um erro aparentemente simples de uma máquina responsável por limpar garrafas causou muitas horas de trabalho para resolver o problema, e prejudicou a imagem da empresa (Nicolaidis, 2010).

Outro caso a ser mencionado é sobre o voo *Airbus* A330-303 realizado em 7 de outubro de 2008 que deixou pelo menos 119 pessoas feridas (ATSB, 2008), e a radiação é a única possível causa da falha ainda não descartada. Durante o voo, uma unidade de referência inercial de dados aéreos começou a produzir dados incorretos, causando o piloto automático a abruptamente abaixar o nariz do avião duas vezes. As figuras 2.1(a) e 2.1(b) mostram os danos causados à aeronave por passageiros e objetos devido à falha.

(a) Seção central acima dos passageiros.

(b) Seção traseira acima dos passageiros.

Figura 2.1: Exemplo dos danos ocasionados. Fonte: (ATSB, 2008)

2.2 PRODUÇÃO DE PROCESSADORES

Para a produção de processadores são utilizados grandes discos de silício, os quais passam por um processo litográfico onde o disco é banhado em reagentes químicos e é atingido por luz ultravioleta, através de uma máscara de acordo com o projeto para fazer as gravuras nesse disco. Esse processo é repetido diversas vezes, para no final o disco ser cortado em múltiplos *chips*. Durante todo esse processo, podem ocorrer alterações em algumas áreas do silício, prejudicando o desempenho de alguns *chips* (Naffziger et al., 2021). O próprio silício pode não ser completamente puro em alguma região ou podem surgir deformações durante o processo litográfico, fazendo com que nenhum *chip* fique completamente igual a outro em um nível atômico, e por consequência diferente no nível funcional.

Entre os *chips* ocorrem variações de quais frequências conseguem suportar, tensão, temperatura ou alguns sub-blocos podem não funcionar. Por isso os *chips* são testados, classificados e vendidos de acordo com seu desempenho, processo conhecido como *binning*.

2.3 RADIAÇÃO

A radiação é energia que possui origem e viaja através do espaço, podendo se comportar como onda ou partícula dependendo das circunstâncias. Ao se comportar como uma onda, a radiação é associada a um campo elétrico e um campo magnético, e também pode ser chamada de ondas eletromagnéticas. A Figura 2.2 apresenta o espectro eletromagnético, onde conseguimos observar alguns exemplos de origem para ondas, a quantidade de energia e se chegam a possuir a capacidade ionizante.

A quantidade de energia da radiação varia de acordo com a frequência e comprimento de sua onda. A luz vermelha, por exemplo, possui menos energia que a luz verde. Radiações com mais energia ganham a propriedade ionizante, onde são capazes de interferir com os elétrons de átomos criando íons (CDC, 2024a).

A principal preocupação de falhas geradas por radiação é quando esta se comporta na forma de partículas. Ao entrar em contato com a atmosfera terrestre, a radiação cósmica gera reações em cadeia, resultando em nêutrons carregados que, ao entrar em contato com semicondutores, podem vir a deixar um rastro ionizante, o qual pode influenciar o comportamento do dispositivo.

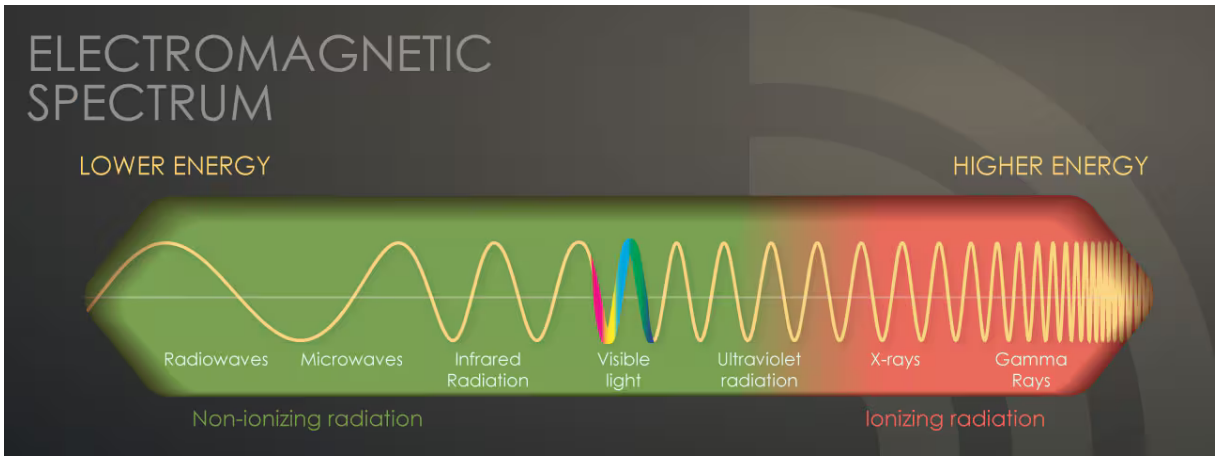


Figura 2.2: Espectro eletromagnético. Extraído de (CDC, 2024b)

2.4 SINGLE-EVENT EFFECT (SEE)

Segundo (Santini, 2015), a variação no desempenho ou estado de um dispositivo microeletrônico causada por apenas um evento de radiação que pode ser medida ou observada é denominada *Single-Event Effect* (SEE). Essa variação (ou erro) pode ser categorizada em dois grupos:

- **Soft error:** ocorre temporariamente, por exemplo, um *bit-flip* causado por raios cósmicos, onde para corrigir basta executar o programa novamente.
- **Hard error:** erro em que o dispositivo fica permanentemente danificado, onde a única correção é com uma peça nova.

O SEE ocorre quando uma partícula de alta energia atravessa um semicondutor e deixa para trás um rastro ionizante. Essa ionização pode causar um efeito localizado que vai desde um erro benigno na saída, um *soft error*, um *bit-flip* mais grave em memória ou registrador, ou em transistores mais potentes um curto-circuito. A Figura 2.3 apresenta um exemplo de SEE.

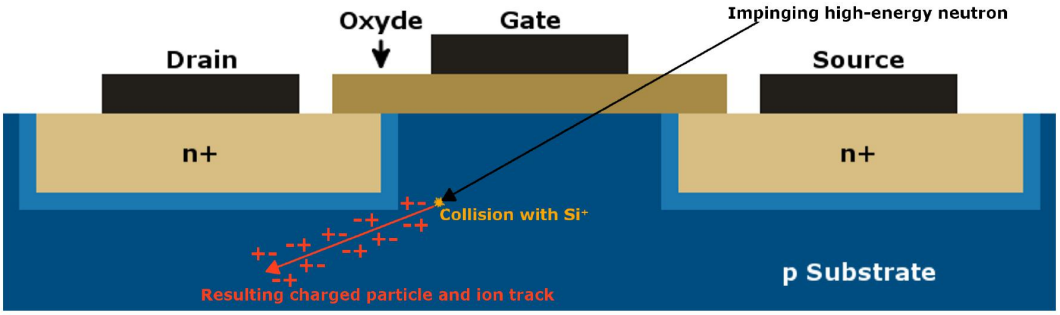


Figura 2.3: Radiação ionizante atingindo semicondutor. Extraído de (Santini, 2015)

O SEE pode ser classificado em diferentes tipos de eventos, de acordo com a Figura 2.4. Por exemplo, cita-se o *Single-Event Upset* (SEU) e *Single-Event Transient* (SET). O primeiro é uma troca em estado de bits da memória ou registradores causado pela interação de um íon com o chip. Já o segundo ocorre quando a carga coletada pela ionização é descarregada através do circuito, gerando um sinal espontâneo. Essa descarga pode desaparecer sozinha, ou se capturada por algum elemento durante seu período de atualização na saída evolui para um SEU.

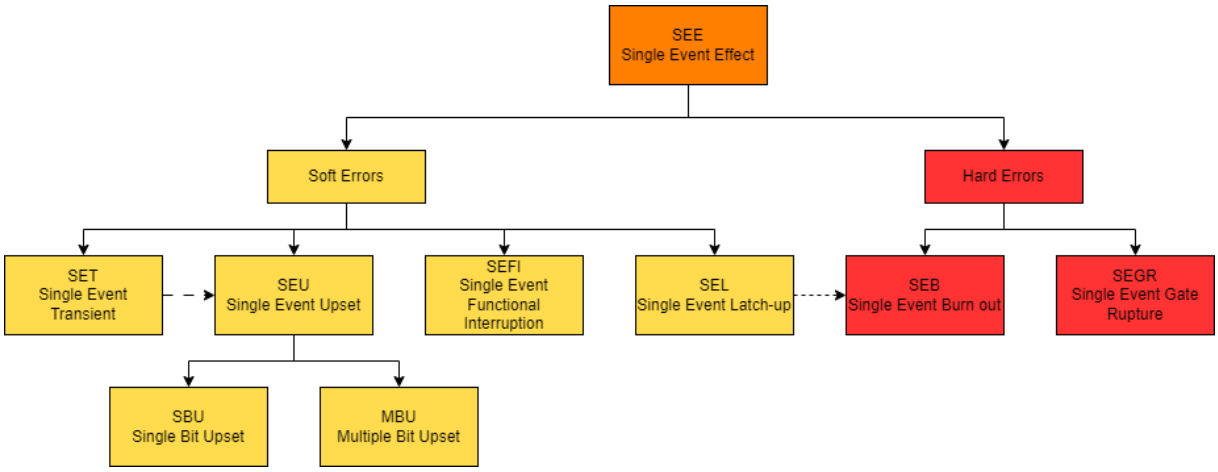


Figura 2.4: Diferentes tipos de Single-Event Effect. Adaptado de (Santini, 2015)

De acordo com (Oliveira, 2017), são três as possíveis reações de radiação causadoras de SEE próximas ao nível do mar:

- Reações de nêutrons de alta energia originados pela interação entre raios cósmicos e a atmosfera com silício e outros materiais.
- Partículas *alpha* emitidas por resquícios de impuridades radioativas do dispositivo.
- Reações entre nêutrons de baixa energia com altas concentrações de boro no dispositivo.

Com uma boa escolha no processo de fabricação e uma proteção adequada, os erros induzidos por partículas *alpha* e reações com boro são uma pequena fração do total de erros, os quais são majoritariamente causados por nêutrons de alta energia que não podem ser protegidos facilmente.

Os raios de alta energia tem como origem raios cósmicos primários que colidem com partículas presentes na atmosfera terrestre, sendo que menos de 1% dos raios primários atingem o nível do mar (Ziegler e Lanford, 1981). As colisões resultam em raios cósmicos secundários que podem decair ou sofrer interações parecidas, resultando em uma reação em cadeia gerando mais partículas exponencialmente até um máximo conhecido como *Pfotzer Maximum* (Pfotzer, 1936), a aproximadamente a 20 km da atmosfera terrestre. Após isso, as partículas começam a decair, perder energia ou serem absorvidas, causando uma redução em seu fluxo conforme a altitude diminui. Ainda assim, algumas partículas de alta energia conseguem atingir o solo terrestre.

É possível simular partículas energéticas originadas pela colisão entre a atmosfera terrestre e raios cósmicos em um acelerador de partículas com uma intensidade muito maior que a normal, em que uma hora de testes em um acelerador equivale à centenas de anos de exposição natural (Oliveira, 2017). Com essas simulações, é possível observar o comportamento de um protótipo eletrônico, código ou produto final quando submetidos à radiação e assim aplicar melhorias se necessário. Porém, as simulações têm um elevado custo financeiro, não sendo possível saber o exato momento em que a falha ocorrerá. Além disso, são poucos aceleradores de partículas capazes de efetuar as simulações e é preciso esperar o decaimento radioativo do material utilizado para simulações antes do transporte dos equipamentos.

A chance de ocorrer um erro ocasionado pela radiação em um único dispositivo é extremamente baixa, porém, em supercomputadores como o *Titan*, que possui mais de 18.000 GPU's trabalhando em conjunto, estes erros ocorrem em intervalos de dezenas de horas, como apresentado por (Tiwari et al., 2015).

2.5 FALHAS POR TEMPO

Circuitos digitais são construídos por diversos componentes eletrônicos, os quais precisam de diferentes tempos mínimos para gerar a saída esperada. Se um sinal elétrico não for propagado corretamente devido a uma violação de temporização no circuito, por exemplo, um processador operando em uma frequência instável, um valor incorreto pode ser propagado. Durante o design de um circuito sequencial, é preciso se atentar ao fato desses tempos serem respeitados. Como exemplo, a Figura 2.5 apresenta as limitações de temporização que precisam ser respeitadas no caso de um *flip-flop*.

Os parâmetros ilustrados na Figura 2.5 são descritos a seguir.

- *Tclk:* Período de um pulso do *clock* síncrono.
- *Idst:* Sinal de entrada do último *flip-flop Edst*. Necessário ser constante pelo período de *Tsetup* para ser estável antes da próxima borda de subida do *clock*.
- *Tsrc:* Latência da primeira unidade sequencial *Esrc* para ter uma saída constante depois de receber a borda de subida do *clock*.
- *Ttransfer:* Representa o tempo de transmissão da saída *Osrc* de *Esrc* para *Idst*, o que também é o tempo de execução dos componentes de lógica.

No caso do tempo mínimo para obter a saída não ser respeitado (diminuindo a tensão, por exemplo), os tempos *Tsrc* e *Ttransfer* aumentam, fazendo com que o *Tsetup* do *flip-flop Edst* não seja respeitado. Com isso, uma saída errada é gerada. Como exemplo, a Figura 2.6 ilustra um caso no qual o tempo mínimo não é respeitado.

Erros também podem ocorrer ao aumentar a tensão, fazendo com que a entrada *Esrc* fique instável. Isso faz com que a sua saída *Osrc* seja alterada de forma incorreta.

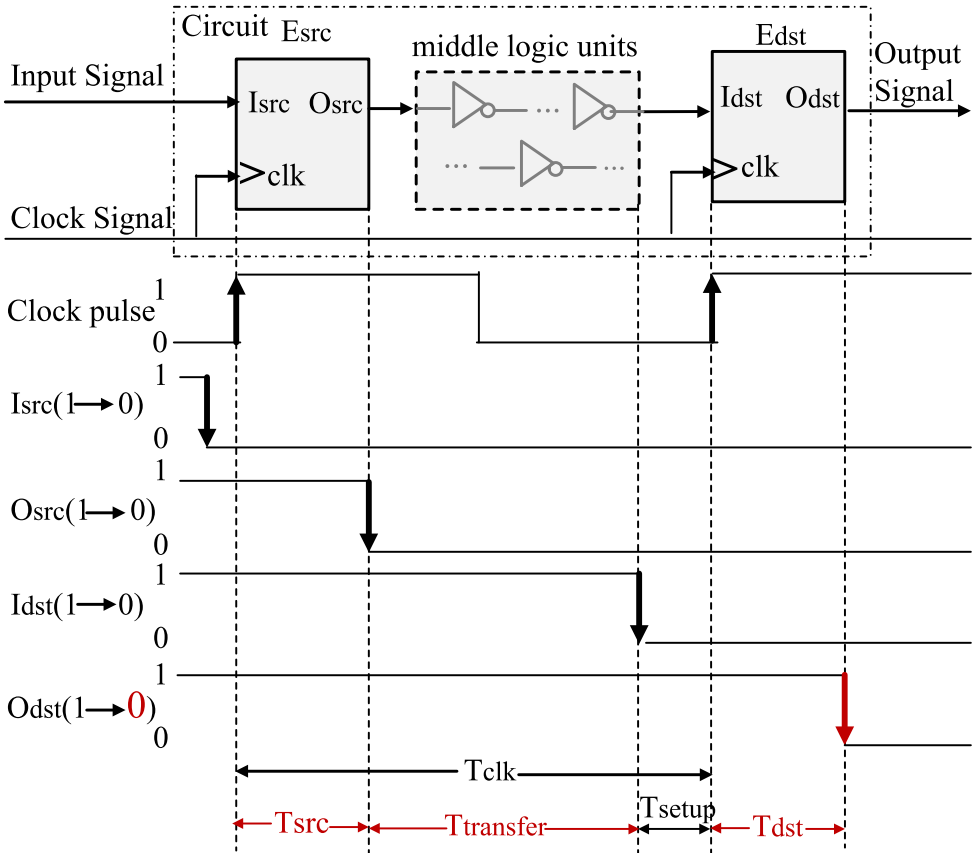


Figura 2.5: Limitações de temporização do circuito. Extraído de (Qiu et al., 2019)

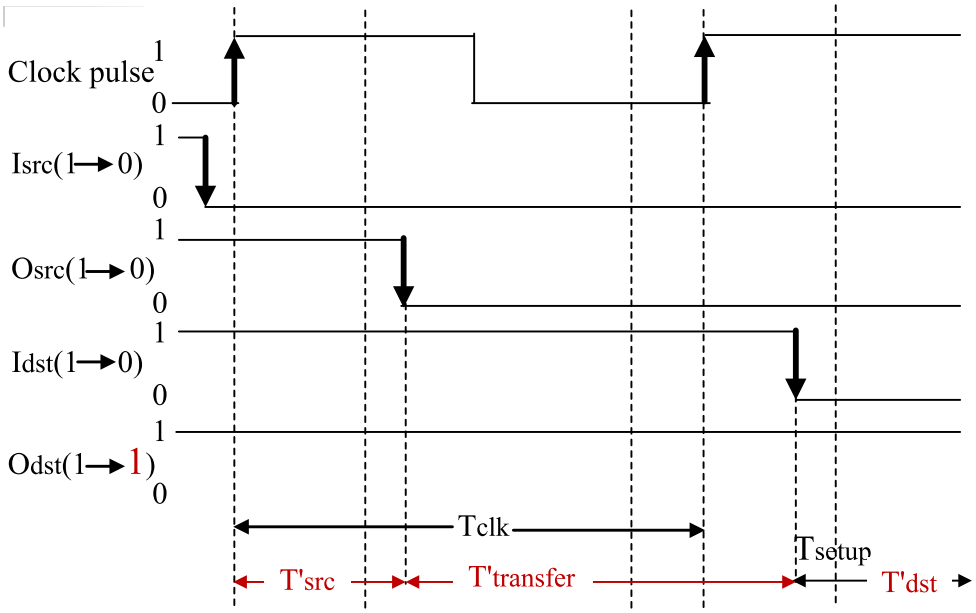


Figura 2.6: Temporização violada. Extraído de (Qiu et al., 2019)

2.6 DYNAMIC VOLTAGE AND FREQUENCY SCALING (DVFS)

Com vista a melhorar a eficiência energética e também o desempenho, a maioria dos processadores atuais utilizam a técnica *Dynamic Voltage and Frequency Scaling* (*DVFS*), que permite ajustar a frequência e tensão operacional dos núcleos do processador de acordo com a

carga de trabalho atual. Por exemplo, um celular sem uso imediato pode entrar em modo de baixa energia reduzindo a velocidade de processamento, assim prolongando a bateria e diminuindo sua temperatura, enquanto para reproduzir vídeos ou jogos é necessário aumentar o consumo de energia, causando uma maior temperatura e redução de sua bateria (Qiu et al., 2019). Por outro lado, sistemas modernos são capazes de efetuar um *overclocking* dinâmico, onde conforme a temperatura e disponibilidade de alimentação do chip, é possível acelerar o processamento em determinados momentos.

Outra vantagem do *DVFS* é que possibilita alguns dispositivos funcionarem apenas com uma refrigeração passiva, ou seja, com o processador diminuindo sua frequência e tensão ao atingir temperaturas muito altas. Desse modo, pode-se reduzir o tamanho e peso de dispositivos que em outro caso precisariam de uma refrigeração ativa para seu funcionamento (Zhou e Guo, 2015).

Embora essa técnica permita que circuitos integrados trabalharem com diferentes tensões e frequências correspondentes em tempo real de acordo com a carga, o *driver* regulador disponível para efetuar esses controles expõe interfaces para *softwares* privilegiados modificarem a tensão e frequência de operação (Qiu et al., 2019).

2.7 BENCHMARKS

De acordo com (Saavedra e Smith, 1996), *benchmarking* é o processo de executar uma carga de trabalho ou programa em um dispositivo para extrair resultados de forma a obter uma medição do desempenho do dispositivo para determinada carga de trabalho. Os resultados de diferentes dispositivos podem ser utilizados para fazer uma comparação e análise de diferenças, por exemplo, comparação de diferentes *CPU's* ou *GPU's*. A Figura 2.7 apresenta um exemplo de *benchmark* executado para efetuar uma comparação entre dois processadores AMD, na qual se observa que o desempenho em uma única *thread* é semelhante para ambos os processadores. No entanto, ao levar em conta todas as *threads*, a diferença aumenta devido, entre outros fatores, ao primeiro processador possuir 12 *threads*, enquanto o segundo possui 32. Os *benchmarks* são uma ferramenta essencial para a comparação e análise do comportamento de aplicações durante a injeção de falhas.

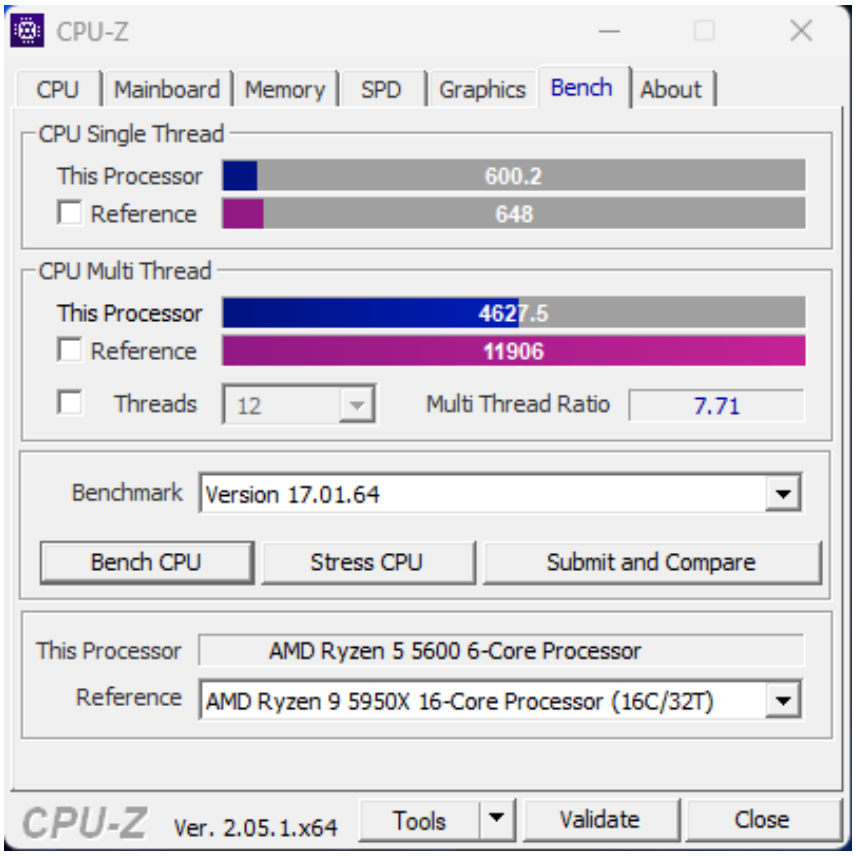


Figura 2.7: Exemplo de *benchmark* comparando processadores *AMD*. Fonte: O autor (2024)

O *benchmark* utilizado *DGEMM* (*Double-Precision Generic Matrix-Matrix Multiplication*) mede o desempenho de operações envolvendo a multiplicação de matrizes. É um *benchmark* padrão para medir a eficiência de bibliotecas numéricas e do *hardware* executando multiplicações de matrizes com números em ponto flutuante com precisão dupla.

A implementação utilizada do *benchmark DGEMM* faz uso do conceito *tile size*, onde a matriz é dividida em blocos menores para o seu processamento. Essa técnica é utilizada para otimizar o desempenho das operações, pois melhora a localidade espacial dos dados e a utilização da memória *cache*.

Utilizando dessa técnica, o processo de multiplicação das matrizes não é feito de forma direta. A matriz é dividida em blocos menores de tamanho previamente especificado, e esses blocos menores são processados. Utilizando de blocos menores para a multiplicação, a memória *cache* é melhor utilizada, acelerando o acesso aos dados. O tamanho dos blocos pode afetar o desempenho, e a escolha do tamanho depende de alguns fatores, como o tamanho da memória *cache*, o tamanho da matriz original ou o a largura de banda da memória.

2.8 WATCHDOG

Watchdog é um mecanismo em *hardware* utilizado para reiniciar o dispositivo caso este venha a travar devido a um erro de *software* ou *hardware*. Funciona através de um contador que é decrementado a cada ciclo do processador e, quando este contador atinge 0, o dispositivo é reiniciado automaticamente. Normalmente, esse contador fica sendo reinicializado durante a operação normal do dispositivo. No entanto, caso o dispositivo trave, o contador irá continuar sendo decrementado até causar a reinicialização do dispositivo (Murphy e Barr, 2001).

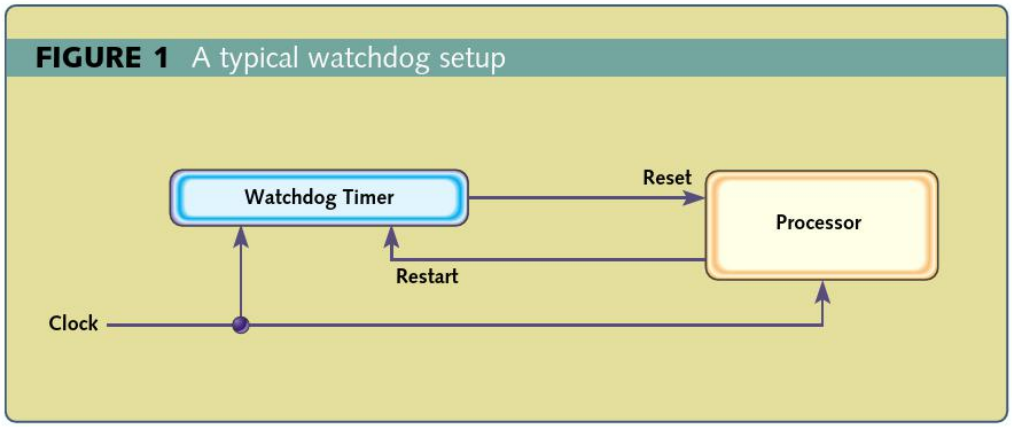


Figura 2.8: Exemplo de *watchdog*. Extraído de (Murphy e Barr, 2001)

2.9 CPU GOVERNOR

A maior parte dos processadores mais recentes dispõem de mecanismos capazes de alterar a frequência e tensão em tempo de execução para reduzir seu consumo energético, ou até mesmo possibilitar um *overclocking*. Inicialmente, essa técnica foi amplamente difundida em dispositivos móveis devido ao uso de baterias, porém, atualmente é algo comum em computadores e servidores. Para fazer o controle da frequência e tensão é utilizado um CPU *Governor* no Linux, sendo utilizado o subsistema do *kernel* chamado *cpufreq* para fazer esse controle (Pallipadi e Starikovskiy, 2006).

O *Cpufreq* permite definir um critério a ser utilizado para gerenciar a frequência da CPU, denominado CPU *Governor*. Por padrão existem 5 diferentes *governors*, descritos a seguir:

- *Performance*: mantém a CPU na frequência mais alta definida pelo usuário.
- *Powersave*: mantém a CPU na menor frequência definida pelo usuário.
- *Userspace*: exporta as informações de frequência e permite controle da frequência pelo usuário.
- *Ondemand*: uma melhoria do *governor userspace* que faz o controle da frequência dentro do próprio *kernel*, assim eliminando o *overhead* de escrita/leitura para a troca de informações entre o ambiente do usuário e o *kernel*. O algoritmo original[1] é apresentado a seguir:

```
1  para cada CPU no sistema
2     a cada X milissegundos
3        obter utilização desde última verificação
4        se (utilização > LIMIAR_SUBIDA)
5           aumentar frequência para MÁXIMO
6     a cada Y milissegundos
7        obter utilização desde última verificação
8        se (utilização < LIMIAR_DESCIDA)
9           diminuir frequência por 20%
```

Listing 2.1: Algoritmo original *ondemand*. Traduzido de (Pallipadi e Starikovskiy, 2006)

- *Conservative*: uma variação do *governor ondemand* com um algoritmo diferente.

[1]Foram feitas diversas melhorias no algoritmo original no decorrer do tempo.

3 CORRELATOS

Nessa seção serão descritos ataques executados através do *DVFS*, e trabalhos correlatos de experimentos em aceleradores de partículas.

3.1 CLKSCREW

O ataque *CLKScrew* permite extrair chaves criptográficas da *TrustZone*, e também escalar privilégios carregando *self-signed codes* na *Trustzone*. Para isso, o ataque faz uso da técnica de gerenciamento de energia *DVFS*, manipulando a tensão e frequência do processador para limites fora dos estabelecidos pelos vendedores (Tang et al., 2017).

Nesse trabalho, será utilizado o *DVFS* apenas para ocasionar falhas e coletar dados obtidos dessas falhas, sem tentar nenhum escalamento de privilégios ou acesso à *TrustZone*.

3.2 VOLTJOCKEY

Ao contrário do ataque *CLKscrew*, o *VoltJockey* manipula a tensão e não a frequência, fazendo com que ele seja mais difícil de detectar e prevenir. Com uma tensão muito baixa, o processador pode não ser capaz de atingir a frequência esperada, causando um atraso entre a comunicação de dois componentes/funções. Similarmente, se a tensão for muito alta, o processador pode apresentar um comportamento errático. Qiu et al. (2019) executaram o ataque, onde o núcleo vítima foi definido com uma alta frequência, e o núcleo atacante com uma frequência baixa. Após, foi definida uma tensão transiente no processador que seja suficientemente alta para o núcleo atacante ser capaz de executar suas tarefas corretamente, mas o núcleo vítima não seja capaz de funcionar normalmente, assim gerando erros no núcleo vítima, enquanto que o atacante funciona normalmente coletando informações dos erros. Ao implementar isso, conseguiram obter chaves AES protegidas pela *TrustZone*, e também carregar qualquer aplicativo na *TrustZone*.

Nesse trabalho também foi utilizado o *DVFS*, porém, no *Raspberry Pi* utilizado não é possível definir uma tensão ou frequência individual a cada núcleo, sendo apenas possível variar a frequência de todos os núcleos ao mesmo tempo, através de uma tensão fixa definida durante a inicialização.

3.3 CAROL-FI

CAROL-FI (Oliveira et al., 2017) é um injetor de falhas em *software* que faz uso do *GDB* e *Python* para estimar a propagação de erros, gerando informações importantes sobre como mitigar os efeitos de falhas. O código é executado pelo *GDB*, onde aleatoriamente é enviada uma interrupção. Com o programa parado, é alterado o valor de uma variável aleatória e retomada a execução do programa.

Nesse trabalho as falhas também foram geradas em *software*, porém sem nenhuma sobrecarga adicional afetando a execução do *benchmark*.

3.4 HARDENING STRATEGIES FOR HPC APPLICATIONS

(Oliveira, 2017) realizou experimentos de radiação em um acelerador de partículas para extrair uma taxa de erros realista. Além disso, foi avaliado o impacto dos experimentos

em diferentes algoritmos paralelos. Uma nova metodologia foi proposta, onde qualificou o erro em cada execução corrompida, correlacionando o número de elementos corrompidos com sua localidade espacial. O trabalho também fornece o erro relativo médio avaliando a magnitude do erro induzido pela radiação.

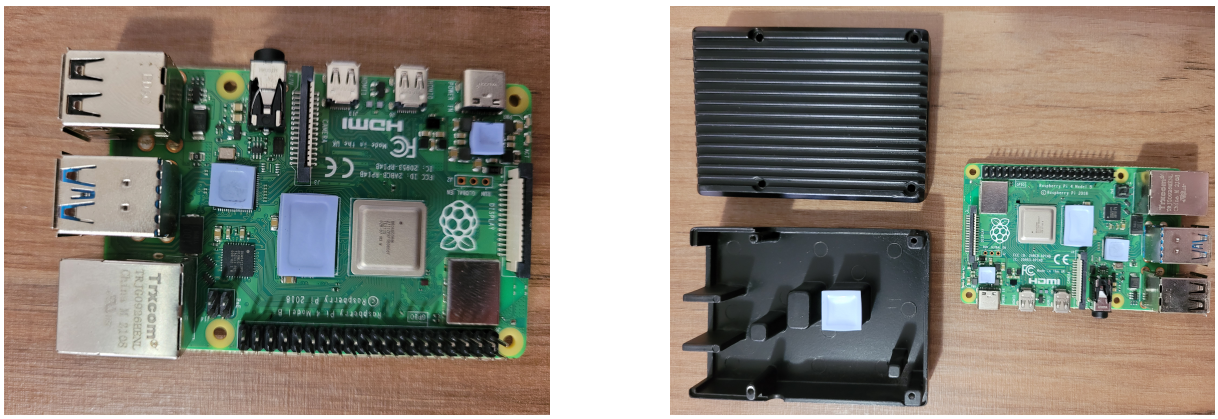
Nesse trabalho foram utilizados os dados dos experimentos com a radiação, e também executado o mesmo *benchmark* utilizado no acelerador de partículas para comparar os resultados, não submetendo os dispositivos ao acelerador de partículas.

4 METODOLOGIA

Para realização deste trabalho, inicialmente foi realizado um levantamento bibliográfico sobre ataques realizados utilizando *DVFS*, descritos na seção 3. Em seguida, foram adquiridos dois *Raspberry PI 4* para realizar o levantamento de dados sobre a estabilidade dos dispositivos em diferentes combinações de tensão e frequência.

4.1 DISPOSITIVOS UTILIZADOS

Com exceção da capacidade de memória *RAM*, onde um dispositivo tem 4 GB's e outro 8 GB's, ambos os dispositivos possuem o mesmo *hardware*. O processador *ARM Cortex-A72* nas especificações do *raspberry* atinge velocidades de até 1.5GHz, e em modelos mais recentes 1.8GHz. Foram instalados com o sistema operacional *Debian GNU/Linux 11 (bullseye)*, versão do *kernel* 6.1.21-v8+. A Figura 4.1(a) apresenta uma foto do *raspberry* utilizado, e a Figura 4.1(b) apresenta o dispositivo com seu *case*, utilizado em ambos para melhor dissipação de calor.



(a) Dispositivo utilizado

(b) Dispositivo com o *case*

Figura 4.1: *Raspberry* 4 GB. Fonte: O Autor (2024)

4.2 UTILIZANDO DVFS

Com o objetivo de gerar falhas durante a execução, foi utilizado o *DVFS* para definir uma combinação de tensão e frequência instável nos processadores de ambos os dispositivos. O *DVFS* utilizado foi através do utilitário *CPU Governor* do Linux, o qual permite alterar a frequência da *CPU* através do *software* em tempo de execução. A frequência máxima e tensão dos *raspberry's* foram configuradas através do arquivo texto de configuração do próprio *raspberry PI*, em que dois parâmetros foram alterados: *arm_freq* e *over_voltage*, descritos a seguir.

- O parâmetro *arm_freq* define a frequência máxima do dispositivo em MHz.
- O parâmetro *over_voltage* controla a tensão máxima, sendo um valor entre -16 e 8 que equivale a 0.95V até 1.55V, com passos de 0.25V. Em outras palavras, um valor de -16 equivale a 0.95V como tensão máxima, e especificando 8 tem-se 1.55V de tensão máxima.

Por padrão, durante a inicialização do *raspberry*, é atribuído ao processador o *CPU Governor ondemand*, o qual altera a frequência do processador de acordo com a carga de trabalho. Porém, isso faz com que durante a inicialização do próprio sistema operacional, por ser uma carga elevada, seja atribuída a frequência máxima ao processador e, se a frequência máxima definida for instável, o sistema não é inicializado. Para resolver este problema, foi alterado o gerenciador padrão durante a inicialização para o *powersave*, o qual limita a frequência em um valor estável (600 MHz), permitindo ao sistema inicializar corretamente.

Para isso, foi instalado o pacote *cpufrequtils* e alterado o arquivo de configuração no caminho */etc/init.d/cpufrequtils* com as seguintes definições:

Listing 4.1: /etc/init.d/cpufrequtils

```
...
ENABLE="true"
GOVERNOR="powersave"
...
```

Também foi feita a seguinte alteração no arquivo *raspi-config*, localizado em /etc/init.d/.

Listing 4.2: /etc/init.d/raspi-config

```
...
if [ -e $SYS_CPUFREQ_GOVERNOR ]; then
echo "powersave" > $SYS_CPUFREQ_GOVERNOR
...
```

Para alterar a frequência do processador durante a execução do *benchmark*, foi alterado o *CPU Governor* utilizando-se dos comandos apresentados a seguir:

Listing 4.3: Comandos utilizados para alterar a frequência do processador

```
sudo cpufreq-set -g conservative
sudo cpufreq-set -g performance
```

O comando *sudo* é utilizado para executar o comando com permissão de super usuário, pois deseja-se alterar a velocidade do processador a nível de *hardware*. O comando *cpufreq-set* permite alterar o *CPU Governor* em tempo de execução. Já a opção *-g* define que a alteração é global, ou seja, afetando todos os núcleos do processador. Por fim, os parâmetros *conservative* ou *performance* definem qual *CPU Governor* a ser utilizado.

4.3 AGENDAMENTO DE TAREFAS

Para a execução de programas logo após a reinicialização do dispositivo, foi utilizado o utilitário *Unix Crontab*, permitindo o agendamento de tarefas através do comando:

Listing 4.4: Comando para a edição do crontab

```
crontab -e
```

Adicionando a seguinte linha dentro do *crontab* agendamos a tarefa para ser executada logo após a inicialização.

Listing 4.5: Comando agendando a execução do script após a inicialização

```
@reboot sh /home/pi/radiation-benchmarks/src/openmp/dgemm/script.sh
```

4.4 MAPEAMENTO DA ESTABILIDADE DO PROCESSADOR

De forma a alterar o arquivo de configuração do *raspberry* variando os parâmetros de tensão e frequência, foi utilizado um *script bash*, disponível em *Github* (Wosniack, 2024). O *script* usa contadores manipulados através de arquivos texto para fazer o controle de execuções inicializadas e finalizadas, de acordo com a frequência e tensão do dispositivo.

Através do agendamento *crontab*, o *script* é executado após a inicialização do sistema operacional, onde primeiramente ele carrega as informações de tensão e frequência atuais do dispositivo e faz a verificação se ambas estão dentro de seus valores máximos configurados.

Com a frequência e tensão dentro dos limites, são carregadas as demais informações de arquivos texto, utilizando contadores separados para execuções inicializadas e finalizadas para o par de frequência e tensão atual. Após todas as informações dos arquivos serem carregadas para o *script*, é inicializada a execução do *benchmark* e incrementado o contador de execuções inicializadas dentro do arquivo texto.

Nesse momento, o dispositivo se encontra com o *CPU Governor powersave*, ou seja, está trabalhando em sua frequência miníma estável. Após o contador de execuções inicializadas ser incrementado, é alterado para o *CPU Governor performance* onde o processador será forçado a trabalhar em sua frequência máxima, ou seja, a frequência definida no arquivo de configuração do *raspberry*. O processador ficará nessa frequência por um tempo configurado, no caso foi utilizado 7 segundos. Se o dispositivo não falhar após esse tempo, o *CPU Governor* retorna para o modo *powersave*, e o arquivo com o contador de execuções finalizadas é incrementado.

O *script* repete o fluxo de inicializar a execução, trocar a frequência do processador, e se não falhar, finalizar a execução até se totalizarem 100 execuções inicializadas. Atingindo o limite de execuções, é incrementada a frequência atual do processador por um valor configurado, a alteração é salva dentro do arquivo de configuração do *raspberry* e o dispositivo é reinicializado. Caso a frequência chegue ao seu valor máximo definido, então ela retorna para o seu menor valor e a tensão é incrementada para o próximo nível. Dessa forma, todas as possíveis combinações de tensão e frequência são analisadas.

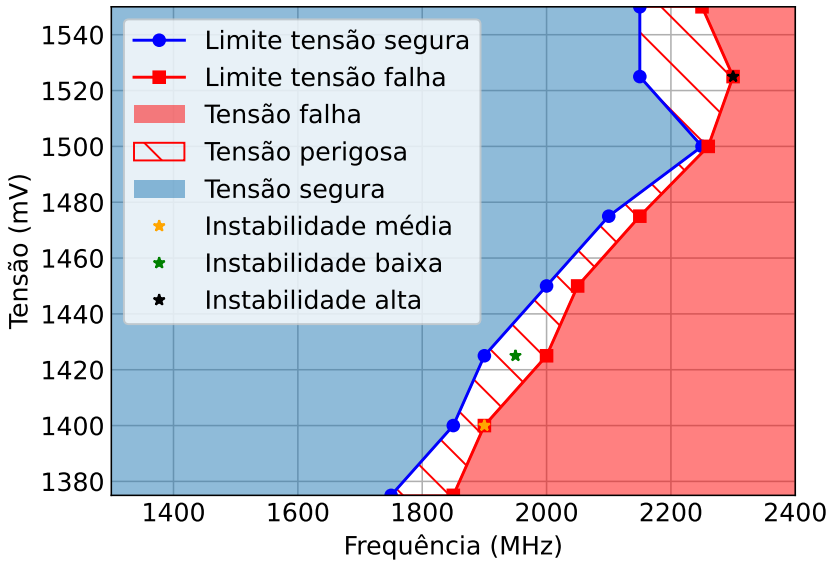
As tabelas 4.1 e 4.2 apresentam os resultados do mapeamento, contendo os valores de tensão e frequência utilizados e quantas execuções finalizaram. Ambos os dispositivos testados possuem o mesmo processador, porém, devido às imperfeições resultantes durante o processo de produção de processadores, é possível observar na prática a diferença nos limites de operação de cada dispositivo. As Figuras 4.2(a) e 4.2(b) apresentam gráficos gerados a partir das tabelas. Utilizando desses resultados, foram escolhidas frequências instáveis para as demais execuções no *raspberry*, ou seja, níveis que não levaram a totalidade de finalização dos experimentos.

Tensão (v) /	Frequência (MHz)													
Overvoltage	**1750**	**1800**	**1850**	**1900**	**1950**	**2000**	**2050**	**2100**	**2150**	**2200**	**2250**	**2300**	**2350**	**2400**
1.35/0	100	100	100	100	100	100	100	100	97	100	98	89	0	0
1.375/1	100	99	72	0	0	0	0	0	0	0	0	0	0	0
1.4/2	100	100	100	90	0	0	0	0	0	0	0	0	0	0
1.425/3	100	100	100	100	99	98	0	0	0	0	0	0	0	0
1.45/4	100	100	100	100	100	100	97	0	0	0	0	0	0	0
1.475/5	100	100	100	100	100	100	100	100	93	0	0	0	0	0
1.5/6	100	100	100	100	100	100	100	100	100	100	100	0	0	0
1.525/7	100	100	100	100	100	100	100	100	100	99	95	60	0	0
1.55/8	100	100	100	100	100	100	100	100	100	99	98	0	0	0

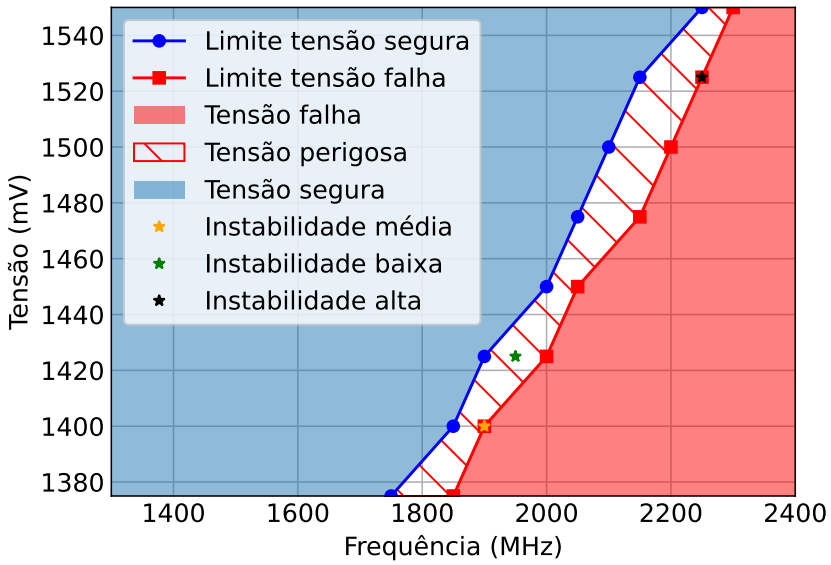
Tabela 4.1: Percentagem de execuções finalizadas - Raspberry 4 GB. Fonte: O Autor (2024)

Tensão (v) /	Frequência (MHz)													
Overvoltage	**1750**	**1800**	**1850**	**1900**	**1950**	**2000**	**2050**	**2100**	**2150**	**2200**	**2250**	**2300**	**2350**	**2400**
1.35/0	100	99	100	100	100	100	100	100	90	83	26	0	0	0
1.375/1	100	94	28	0	0	0	0	0	0	0	0	0	0	0
1.4/2	100	100	100	87	0	0	0	0	0	0	0	0	0	0
1.425/3	100	100	100	100	99	96	0	0	0	0	0	0	0	0
1.45/4	100	100	100	100	100	100	96	0	0	0	0	0	0	0
1.475/5	100	100	100	100	100	100	100	96	95	0	0	0	0	0
1.5/6	100	100	100	100	100	100	100	100	91	33	0	0	0	0
1.525/7	100	100	100	100	100	100	100	100	100	95	33	0	0	0
1.55/8	100	100	100	100	100	100	100	100	100	100	100	27	0	0

Tabela 4.2: Percentagem de execuções finalizadas - Raspberry 8 GB. Fonte: O Autor (2024)



(a) Mapeamento *Raspberry* 4 GB.



(b) Mapeamento *Raspberry* 8 GB

Figura 4.2: Limites de tensão e frequência dos processadores. Fonte: O Autor (2024)

4.5 REINICIALIZAÇÃO DOS DISPOSITIVOS

Foi utilizado o *watchdog* do próprio *raspberry PI* para fazer a reinicialização caso o aparelho trave após a inicialização do *kernel*. Para ativar o *watchdog*, é necessário adicionar no arquivo de configuração do *raspberry* o seguinte parâmetro:

Listing 4.6: */boot/config.txt*

```
1 dtparam=watchdog=on
```

No entanto, caso ocorram erros durante a inicialização do *kernel* e antes do monitoramento *watchdog*, é necessária a reinicialização manual do dispositivo, removendo a energia do mesmo.

Para automatizar a reinicialização manual do dispositivo, foi utilizado um *Arduino* controlando relés, estes conectados as fontes dos *raspberry*. Essa solução resultou em uma

tomada inteligente, na qual é possível desligar ou ligar o *raspberry* através do *Arduino*. Em seguida, foi utilizada a interface serial entre o *Arduino* e o computador, sendo possível enviar e receber comandos entre ambas as partes. Esta interface permite ligar ou desligar o *raspberry* através do computador. A Figura 4.3 ilustra as ligações entre os dispositivos.

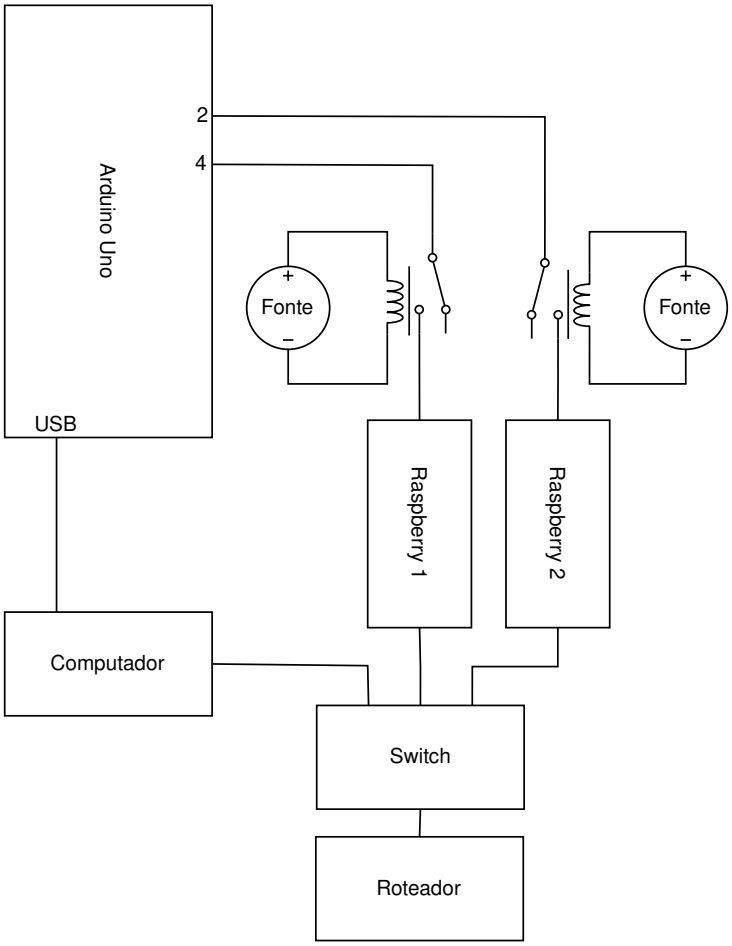


Figura 4.3: Diagrama das conexões. Fonte: O Autor (2024)

Foi utilizado um programa *Python* disponível em (Wosniack, 2024), responsável por criar uma interface serial com o *Arduino* enviando e recebendo *strings* para controlar a saída do relé, por exemplo, enviar a mensagem de *LIGAR* para ligar o raspberry, ou *DESLIGAR* para desligar. O programa também faz um monitoramento através de *ping's* a cada 15 segundos para ambos os *raspberry*. Caso o computador não receba resposta dos dispositivos em 6 tentativas, é enviado o comando para desligar seguido do comando ligar para o *Arduino* o qual age de acordo com o relé.

4.6 EXECUÇÕES DO BENCHMARK

As execuções do *benchmark DGEMM* (*Double-Precision Generic Matrix-Matrix Multiplication*) foram realizadas com as dimensões da matriz 2048x2048. Não foram executadas variações no tamanho da matriz devido aos dispositivos possuírem capacidade de memória distintas.

O *tile size*, ou seja, as subdivisões da matriz para efetuar os cálculos foi definida como 16, e o número de *threads* durante a execução foram 4, o número de *threads* do processador.

As execuções foram realizadas com duas temporizações distintas, 3 ou 6 segundos. Esse tempo é a duração do *CPU Governor* em *performance* ou *powersave* durante a execução do *benchmark*. Em outras palavras, por quanto tempo o dispositivo permanece em cada uma das frequências antes de retornar para a outra. Os experimentos foram realizados com cada um desses valores isoladamente, afim de determinar o comportamento geral dessa variável nos experimentos.

Além das temporizações, também foram feitas execuções do *benchmark* em pares de frequência e tensão exibidos nas tabelas 4.1 e 4.2 com estabilidades distintas. Inicialmente, foi escolhido o par 1.4V ou *overvoltage=2* e frequência 1900MHz, pois ambos os dispositivos apresentaram estabilidade similar nessa mesma frequência e tensão. Esse par então foi categorizado como uma instabilidade média e foram escolhidos outros pares para representar instabilidades baixa e alta.

Para a instabilidade baixa foi escolhido o par 1.425V ou *overvoltage=3* a 1950MHz onde ambos os dispositivos raramente apresentaram erros durante a execução.

Podemos notar que embora ambos os dispositivos possuírem o mesmo processador *ARM Cortex-A72*, eles apresentam diferentes limites de tensão e frequência devido à variação durante a fabricação dos *chips*, diferença exibida nas Figuras 4.1 e 4.2.

Para representar a instabilidade alta foram escolhidos diferentes pares de tensão e frequência para cada dispositivo, mas ainda tentando obter um resultado similar em termos de percentagem de falhas, onde no *Raspberry 4 GB* foi o par 1.525v ou *overvoltage=7* a 2300MHz, e no *Raspberry 8 GB* foi o par 1.525V ou *overvoltage=7* a 2250MHz.

4.7 SAÍDAS DO BENCHMARK

O próprio *benchmark* utilizado gera *logs* de cada execução, contendo diversas informações como o tempo para computar a iteração *KerTime*, e o tempo acumulado do programa até o momento *AccTime*. Além disso, em cada iteração podem ocorrer um ou mais erros. No caso do *benchmark DGEMM*, o erro é uma célula incorreta da matriz, com a posição exibida em *p: [linha, coluna]*, o valor esperado em *e:* e o valor lido em *r:*.

Listing 4.7: Exemplo de saída do *benchmark*

```
...
#IT 121 KerTime:2.6936211 AccTime:327.7292374
#ERR p: [337, 1750], r: -2.2136512014570642e+34, e: -2.4433801911313936e+34
...
```

As saídas do *benchmark* são interpretadas através de um *parser* classificando os erros de cada iteração, calculando a magnitude e gerando gráficos para facilitar sua interpretação.

A magnitude do erro é medida através do erro relativo, o qual é calculado com a seguinte equação:

$$relative\ error = \frac{|read - expected|}{|expected|} \times 100$$

, onde *read* é o valor do elemento incorreto e *expected* o valor correto.

4.8 CLASSIFICAÇÃO DOS ERROS

Os erros foram agrupados em iterações, onde todos os erros de uma iteração são classificados juntos em uma de quatro categorias: *single*, *random*, *line* ou *square*. A Figura 4.4 ilustra essa classificação.

- *Single*: Em uma nova leitura de erros após a iteração, se há pelo menos um erro, assume-se que é apenas um erro, logo é classificado como *single*.
- *Random*: Há mais de um erro, porém pelo menos um dos erros não compartilha nenhum índice em comum na matriz.
- *Line*: Todos os erros compartilham apenas uma linha ou coluna na matriz.
- *Square*: Todos os erros compartilham as mesmas colunas e linhas na matriz.

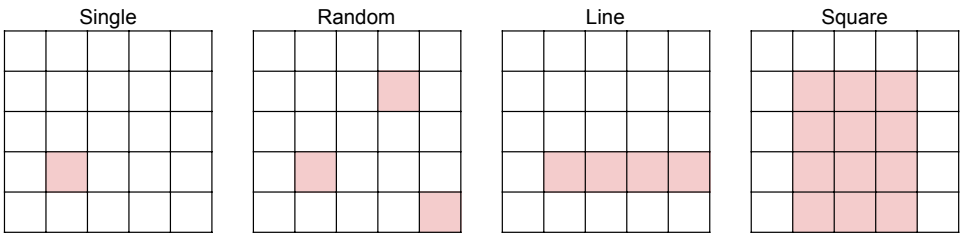


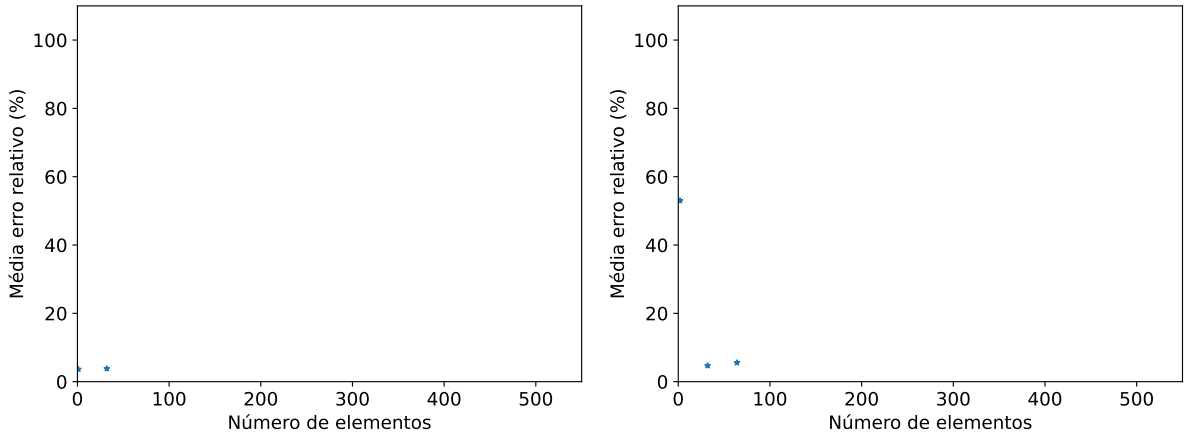
Figura 4.4: Exemplo da classificação dos erros. Fonte: O Autor (2024)

5 RESULTADOS

Esta seção tem por objetivo apresentar os resultados obtidos, e fazer uma comparação com os resultados disponibilizados por (Oliveira, 2017), onde foram submetidos aceleradores a um acelerador de partículas.

5.1 INSTABILIDADE BAIXA

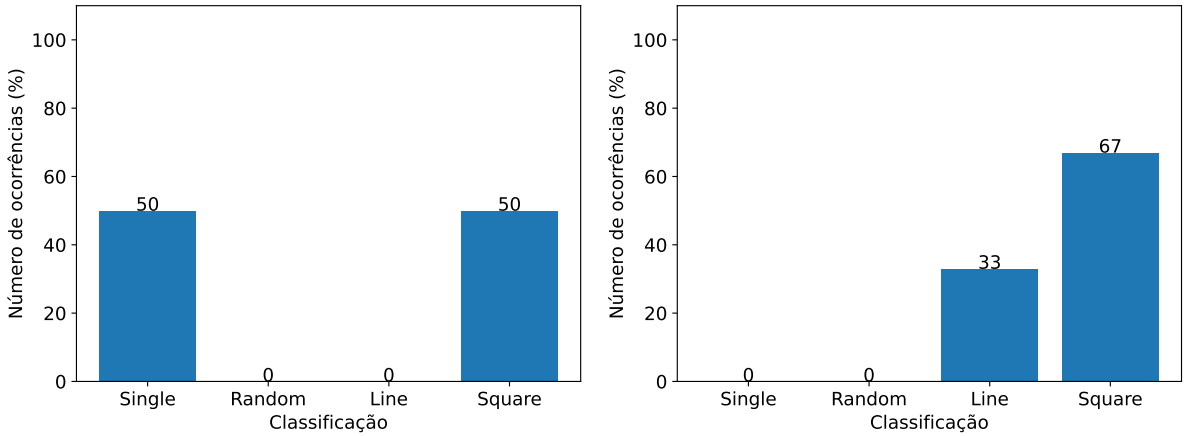
A execução do *benchmark* com uma baixa instabilidade resulta em raramente um erro ocorrer, como podemos observar nas Figuras 5.1 e 5.2. Isso ocorre devido a chance do dispositivo falhar ser extremamente baixa, pois o processador se encontra quase completamente dentro de seus limites normais de operação.



(a) 6 segundos - *Raspberry* 4 GB

(b) 3 segundos - *Raspberry* 8 GB

Figura 5.1: Média do erro relativo e número de elementos incorretos *DGEMM* instabilidade baixa. Fonte: O Autor (2024)



(a) 6 segundos - *Raspberry* 4 GB

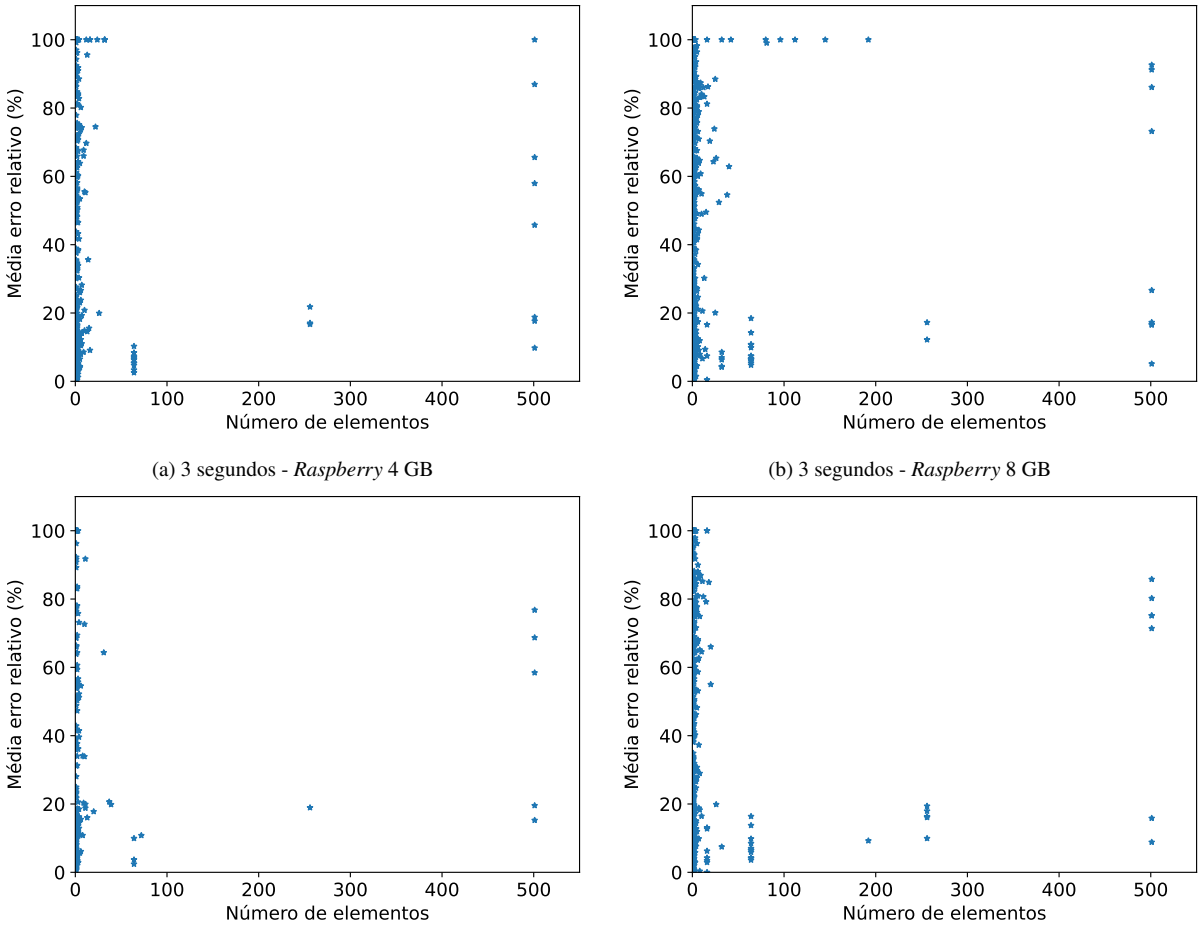
(b) 3 segundos - *Raspberry* 8 GB

Figura 5.2: Localidade espacial *DGEMM* instabilidade baixa. Fonte: O Autor (2024)

5.2 INSTABILIDADE MÉDIA

As execuções em média instabilidade geram diversos erros, pois o dispositivo acaba produzindo mais falhas que as execuções com a baixa instabilidade, e também, não fica instável conseguindo manter a execução por mais tempo se comparado a alta instabilidade.

Observando os gráficos da Figura 5.3, notamos que a maior parte dos erros ocorre em poucos elementos, onde pelo menos 70% das classificações ocorrem com apenas 1 ou 2 elementos incorretos. Outro ponto a ressaltar é que o número de elementos incorretos possuem uma tendência a algumas potências de 2, como podemos observar na Tabela 5.1. Essa tendência provavelmente ocorre devido aos parâmetros utilizados durante a execução do próprio *benchmark*.



(a) 3 segundos - *Raspberry* 4 GB

(b) 3 segundos - *Raspberry* 8 GB

(c) 6 segundos - *Raspberry* 4 GB

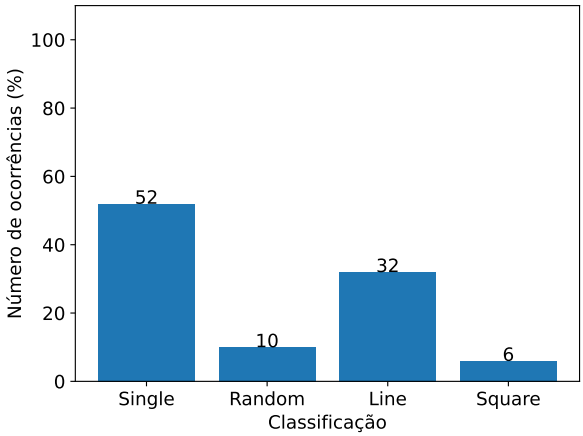
(d) 6 segundos - *Raspberry* 8 GB

Figura 5.3: Média do erro relativo e número de elementos incorretos *DGEMM* instabilidade média. Fonte: O Autor (2024)

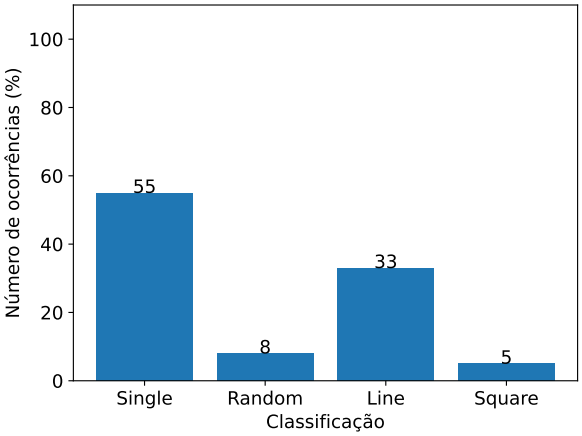
Ao comparar execuções de um mesmo dispositivo entre as duas temporizações, conforme apresentado na Figura 5.4, nota-se um padrão onde ao manter a frequência instável por mais tempo resulta no aumento em erros de apenas um elemento da matriz, e uma leve redução em erros classificados como aleatórios e linha.

Tamanho	Dispositivo - Temporização(segundos)				
	4 GB - 3	4 GB - 6	8 GB 3	8 GB - 6	Total
1	192	114	368	281	955
2	69	31	119	75	294
3	31	16	43	34	124
4	13	10	25	23	71
5	10	1	20	10	41
6	7	3	12	8	30
7	5	0	8	6	19
8	0	2	4	5	11
9	3	1	2	1	7
10	3	2	4	2	11
11	1	3	2	1	7
12	2	0	1	1	4
13	2	1	2	0	5
14	1	0	1	0	2
15	1	0	1	1	3
16	3	0	5	9	17
17	0	0	1	0	1
18	0	0	0	1	1
19	0	0	1	0	1
20	0	1	0	2	3
22	1	0	0	0	1
23	0	0	1	0	1
24	1	0	1	0	2
25	0	0	2	0	2
26	1	0	1	1	3
29	0	0	1	0	1
30	0	0	0	0	0
31	0	1	0	0	1
32	3	0	6	1	10
37	0	1	0	0	1
38	0	0	1	0	1
39	0	1	0	0	1
40	0	0	1	0	1
42	0	0	1	0	1
64	11	3	13	11	38
72	0	1	0	0	1
80	0	0	1	0	1
81	0	0	1	0	1
96	0	0	1	0	1
112	0	0	1	0	1
145	0	0	1	0	1
192	0	0	1	1	2
256	3	1	2	5	11
501	8	5	9	7	29

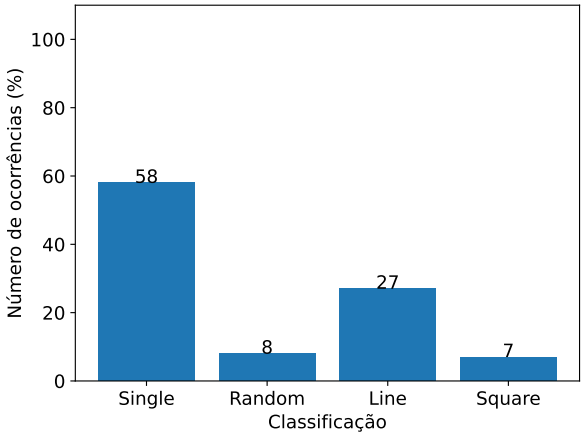
Tabela 5.1: Ocorrência do número de elementos incorretos por iteração. Fonte: O Autor (2024)



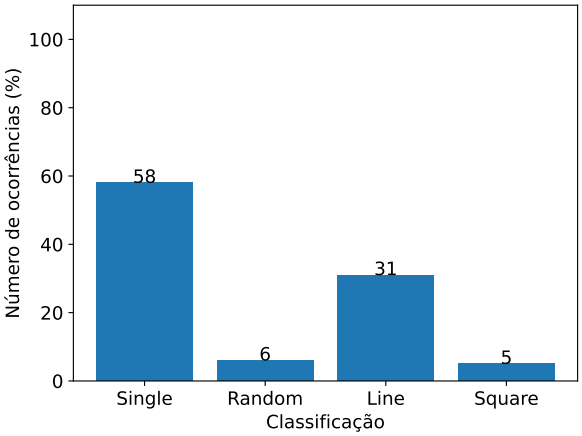
(a) 3 segundos - *Raspberry* 4 GB



(b) 3 segundos - *Raspberry* 8 GB[1]



(c) 6 segundos - *Raspberry* 4 GB



(d) 6 segundos - *Raspberry* 8 GB

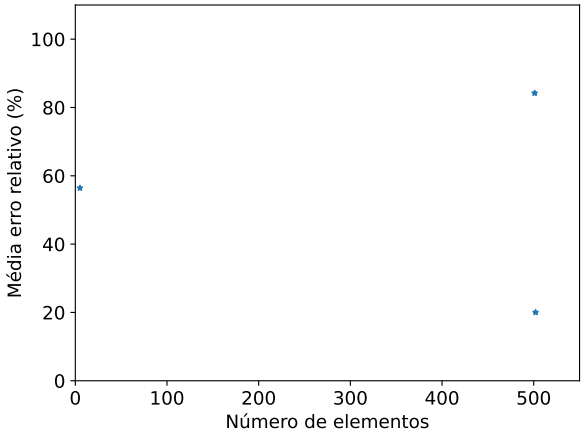
Figura 5.4: Localidade espacial *DGEMM* instabilidade média. Fonte: O Autor (2024)

5.3 INSTABILIDADE ALTA

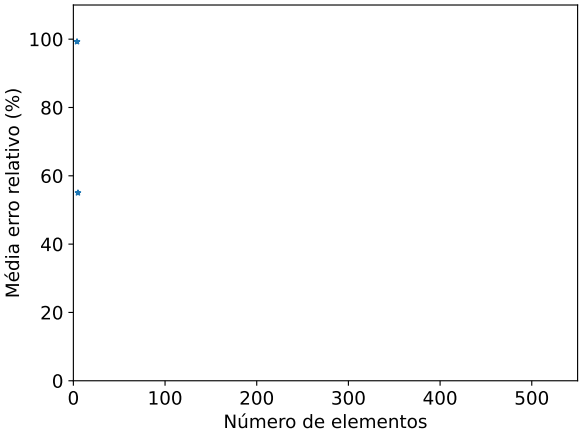
Com o processador sendo forçado a trabalhar em uma frequência instável, os dispositivos tendem a travar assim que a frequência máxima é aplicada. A execução do *benchmark* se torna muito breve, pois assim que o dispositivo trava ele é forçado a reiniciar.

Como podemos observar nas Figuras 5.5 e 5.6, poucos erros foram capturados durante o decorrer das execuções. Essa falta de erros se deve ao fato do processador estar muito instável, não sendo capaz de manter a execução do *benchmark* por tempo suficiente para gerar erros.

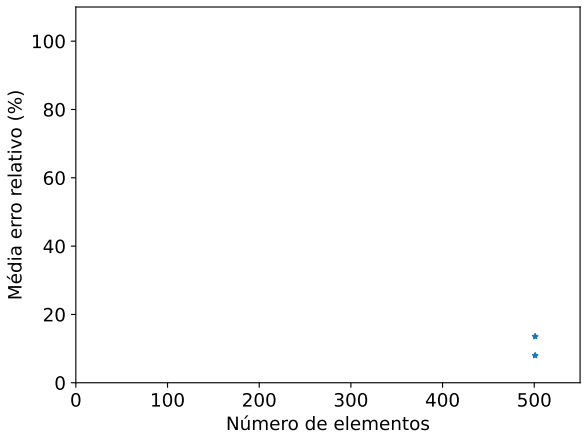
[1]Gráfico totalizando 101% devido a erros de arredondamento.



(a) 3 segundos - *Raspberry* 4 GB

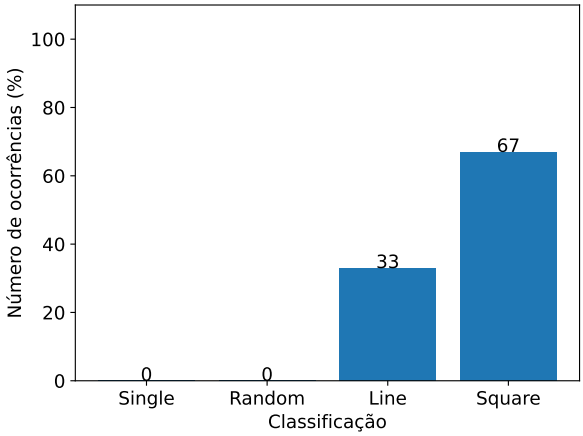


(b) 6 segundos - *Raspberry* 4 GB

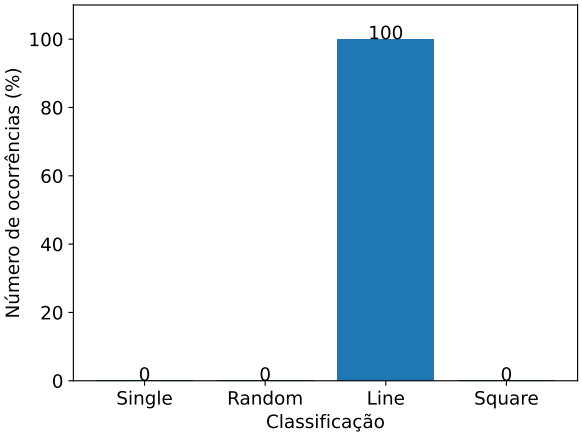


(c) 6 segundos - *Raspberry* 8 GB

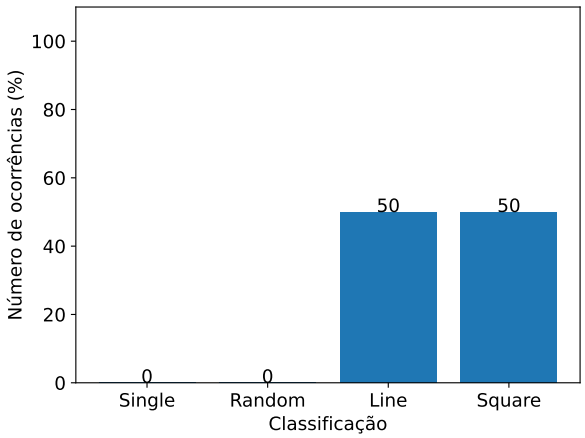
Figura 5.5: Média do erro relativo e número de elementos incorretos *DGEMM* instabilidade alta. Fonte: O Autor (2024)



(a) 3 segundos - *Raspberry* 4 GB



(b) 6 segundos - *Raspberry* 4 GB



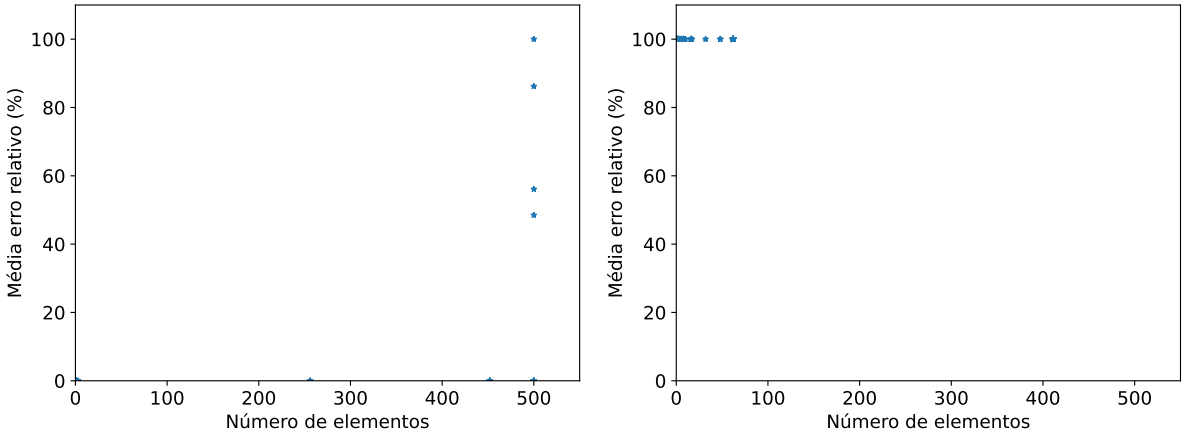
(c) 6 segundos - *Raspberry* 8 GB

Figura 5.6: Localidade espacial *DGEMM* instabilidade alta. Fonte: O Autor (2024)

5.4 COMPARAÇÃO DOS RESULTADOS

Esta seção apresenta os resultados disponibilizados por (Oliveira, 2017) submetidos através do *parser* utilizado neste trabalho para fins de comparação.

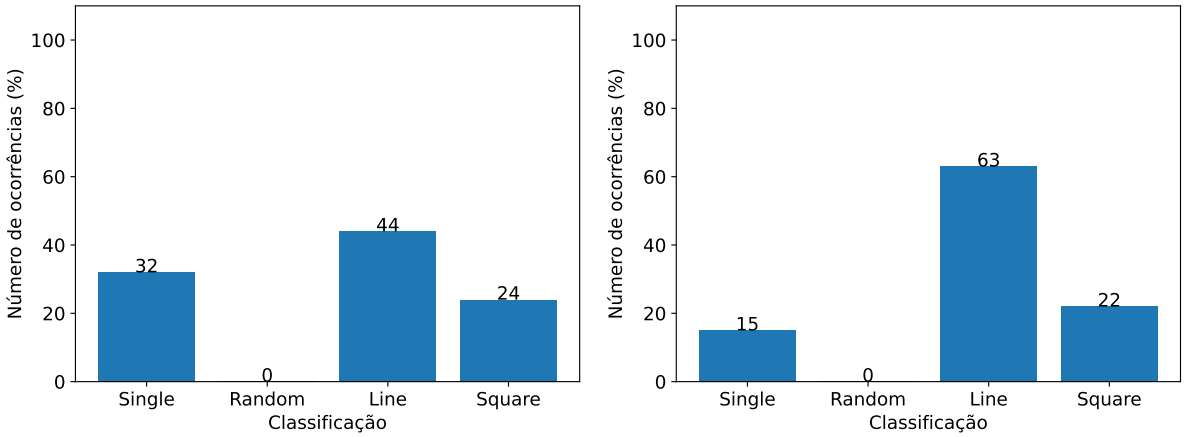
A Figura 5.7 apresenta a magnitude e o número de elementos incorretos de dois aceleradores. Ambos possuem um comportamento distinto quando submetidos a radiação, onde a *Xeon Phi* apresenta ser mais vulnerável a erros se comparada com a *K40*, onde qualquer perturbação faz os erros atingirem sua magnitude máxima.



(a) K40 (b) Xeon Phi

Figura 5.7: Média do erro relativo e número de elementos incorretos *DGEMM*. Fonte: (Oliveira, 2017)

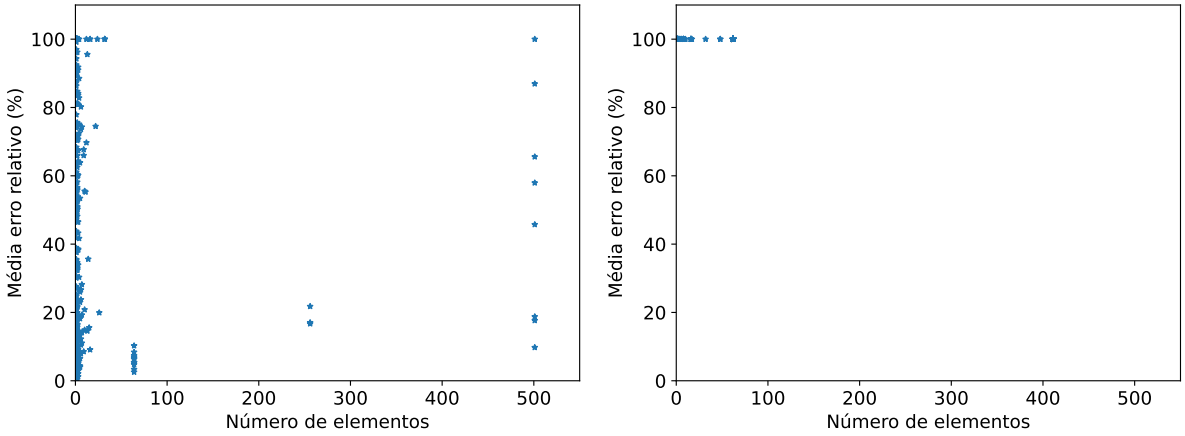
São apresentadas a localidade espacial dos erros na Figura 5.8. O acelerador *Xeon Phi* apresenta uma maior vulnerabilidade para erros classificados em linhas se comparada a *K40*.



(a) K40 (b) Xeon Phi

Figura 5.8: Localidade espacial *DGEMM*. Fonte: (Oliveira, 2017)

Para a comparação com o *raspberry*, foi escolhido o *Xeon Phi* devido a ambos serem processadores. A Figura 5.9 apresenta a comparação entre os resultados do trabalho realizado, e o trabalho de (Oliveira, 2017). Nota-se que o comportamento das falhas em relação as suas magnitudes é diferente.

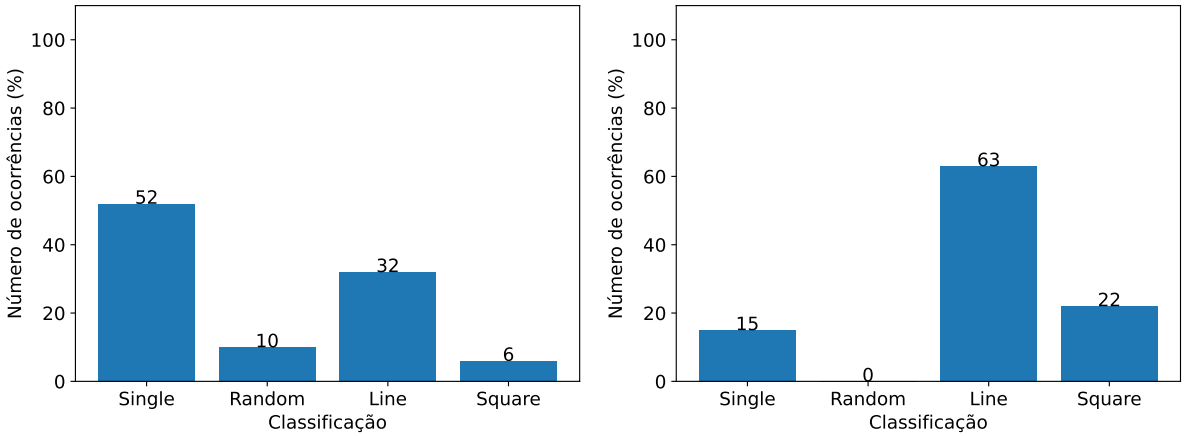


(a) 3 segundos - *Raspberry* 4 GB

(b) Xeon Phi

Figura 5.9: Comparação média do erro relativo e número de elementos incorretos *DGEMM*.

Na comparação entre a localidade espacial, apresentada na Figura 5.10, é possível observar que a radiação causa um comportamento diferente, no qual mais linhas e quadrados de uma matriz são corrompidas e não chega a ocorrer nenhum caso de erros aleatórios. A localidade espacial obtida durante os testes com *DVFS* possui um viés devido a existência de algumas células mais fracas no *chip*, as quais são mais facilmente perturbadas com a instabilidade do processador. Por outro lado, a localidade espacial dos aceleradores submetidos ao acelerador de partículas depende principalmente da arquitetura do dispositivo testado e do código utilizado durante os testes.



(a) 3 segundos - *Raspberry* 4 GB

(b) Xeon Phi

Figura 5.10: Comparação localidade espacial *DGEMM*.

Com as informações apresentadas, não foi possível simular as falhas ocasionadas por radiação através de *DVFS*. A maior parte dos erros gerados através da instabilidade do processador são com poucos elementos afetados, o que parece não ocorrer na prática com a radiação.

Outro ponto a ressaltar é a grande diferença entre as arquiteturas comparadas, em que o processador do *Raspberry Pi 4* é ARM, enquanto que o *Xeon Phi* é um processador Intel e a K40 é uma *GPU*.

6 CONCLUSÕES

Este trabalho tem como objetivo apresentar um estudo da simulação de falhas causadas pela radiação ionizante através da tecnologia *DVFS*. Os dispositivos utilizados para os experimentos foram dois *raspberry's* devido a sua acessibilidade, seu potencial de automação e por seus processadores suportarem a tecnologia *DVFS*.

Durante os testes realizados, foi observada a diferença entre processadores de um mesmo modelo, onde devido à variação no processo de fabricação, ambos possuem limites de frequência distintos.

Ao longo deste trabalho, foram criadas ferramentas e códigos para efetuar os testes, assim como a conexão entre os diferentes dispositivos necessários. Por exemplo, foi desenvolvida uma tomada inteligente composta por relés, responsável por automatizar a reinicialização dos *raspberry*. Esta tomada é controlada por um *arduino*, o qual se comunica com o computador através da interface serial por meio de *strings*. Por sua vez, o computador é responsável por monitorar ambos os *raspberry*, através de comandos *ping*.

Para o tratamento dos dados obtidos, foi implementado em *Python* um código responsável por categorizar e calcular a magnitude dos erros introduzidos pela instabilidade do processador durante a execução do *benchmark*.

Os resultados obtidos neste trabalho foram comparados com as saídas do mesmo *benchmark* executado em aceleradores submetidos a um acelerador de partículas, o qual incide sobre os dispositivos o equivalente a centenas de anos de exposição a radiação ionizante. Para isso, foram utilizadas duas diferentes métricas, a magnitude e localidade espacial dos erros.

Analisando a localidade espacial, nota-se que a probabilidade de erros aleatórios ocorrerem em um processador instável é maior que em aceleradores submetidos a radiação ionizante, os quais não apresentam tal erro quando executada a mesma configuração do *benchmark*.

No caso da magnitude dos erros, observa-se que o processador instável gera a maior parte de seus erros em poucos elementos, em que 1 ou 2 elementos incorretos totalizam ao menos 70% de todos os erros. Já no caso dos aceleradores atingidos pela radiação, não é observado um viés em relação ao número de elementos incorretos.

Com base nos resultados obtidos, conclui-se que o comportamento das falhas ocasionadas pela radiação ionizante é diferente do observado durante os testes realizados com o processador instável. Porém, considera-se que o método proposto pode ser utilizado para avaliação de técnicas de mitigação de falhas a fim de criar sistemas mais robustos.

É importante ressaltar que este trabalho utilizou valores arbitrários para realização de alguns testes, por exemplo, o tempo para permanecer em frequência máxima, o tempo utilizado para marcar uma execução como completa, as definições do próprio *benchmark*, entre outros. Tais valores de configuração podem ser facilmente modificados através de definições globais nos arquivos disponibilizados e são considerados possíveis trabalhos futuros. Além disso, não foram realizadas variações no tamanho de entrada do *benchmark*, apenas realizando execuções com as dimensões de 2048x2048.

REFERÊNCIAS

ATSB (2008). In-flight upset 154km west of learmonth, wa 7 october 2008 vh-qpa airbus a330-303. Relatório Técnico AO-2008-70, Australian Transport Safety Bureau, Australia.

CDC (2024a). Cdc - the eletromagnetic spectrum. `https://www.cdc.gov/nceh/radiation/spectrum.html`. Acessado em 22/04/2024.

CDC (2024b). Cdc - what is radiation. `https://www.cdc.gov/nceh/radiation/what_is.html`. Acessado em 22/04/2024.

Dixit, A. e Wood, A. (2011). The impact of new technology on soft error rates. *2011 International Reliability Physics Symposium*, páginas 5B.4.1–5B.4.7.

Lakshminarayanan, V. e Sriraam, N. (2014). The effect of temperature on the reliability of electronic components. Em *2014 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT)*, páginas 1–6.

Murphy, N. e Barr, M. (2001). Watchdog timers. *Embedded Systems Programming*, 14(11):79–80.

Naffziger, S., Beck, N., Burd, T., Lepak, K., Loh, G. H., Subramony, M. e White, S. (2021). Pioneering chiplet technology and design for the amd epyc™ and ryzen™ processor families : Industrial product. Em *2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA)*, páginas 57–70.

Nicolaidis, M. (2010). *Soft errors in modern electronic systems*, volume 41. Springer Science & Business Media.

Oliveira, D., Frattin, V., Navaux, P., Koren, I. e Rech, P. (2017). Carol-fi: An efficient fault-injection tool for vulnerability evaluation of modern hpc parallel accelerators. Em *Proceedings of the Computing Frontiers Conference*, CF'17, página 295–298, New York, NY, USA. Association for Computing Machinery.

Oliveira, D. A. G. d. (2017). Hardening strategies for hpc applications.

Pallipadi, V. e Starikovskiy, A. (2006). The ondemand governor. Em *Proceedings of the linux symposium*, volume 2, páginas 215–230.

Pfotzer, G. (1936). Dreifachkoinzidenzen der ultrastrahlung aus vertikaler richtung in der stratosphäre. *Zeitschrift für Physik*, 102(1):41–58.

Qiu, P., Wang, D., Lyu, Y. e Qu, G. (2019). Voltjockey: Breaching trustzone by software-controlled voltage manipulation over multi-core frequencies. Em *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, CCS '19, página 195–209, New York, NY, USA. Association for Computing Machinery.

Rotem, E., Naveh, A., Ananthakrishnan, A., Weissmann, E. e Rajwan, D. (2012). Power-management architecture of the intel microarchitecture code-named sandy bridge. *IEEE Micro - MICRO*, 32:20–27.

Saavedra, R. H. e Smith, A. J. (1996). Analysis of benchmark characteristics and benchmark performance prediction. *ACM Trans. Comput. Syst.*, 14(4):344–384.

Santini, T. C. (2015). Increasing embedded software radiation reliability through cache memories.

Schaller, R. (1997). Moore's law: past, present and future. *IEEE Spectrum*, 34(6):52–59.

Schrimpf, R. e Fleetwood, D. (2004). *Radiation Effects And Soft Errors In Integrated Circuits And Electronic Devices*. Selected Topics In Electronics And Systems. World Scientific Publishing Company.

Tang, A., Sethumadhavan, S. e Stolfo, S. (2017). {CLKSCREW}: exposing the perils of security-oblivious energy management. Em *26th {USENIX} Security Symposium ({USENIX} Security 17)*, páginas 1057–1074.

Tiwari, D., Gupta, S., Gallarno, G., Rogers, J. e Maxwell, D. (2015). Reliability lessons learned from gpu experience with the titan supercomputer at oak ridge leadership computing facility. Em *SC '15: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, páginas 1–12.

Wosniack, A. (2024). TCC - DVFS. `https://github.com/awosniack/tcc/tree/master`.

Zhou, L. e Guo, S. (2015). Thermal management of arm socs using linux cpufreq as cooling device. *Computer Modelling and New Technologies*, páginas 162–167.

Ziegler, J. F. e Lanford, W. A. (1981). The effect of sea level cosmic rays on electronic devices. *Journal of Applied Physics*, 52(6):4305–4312.