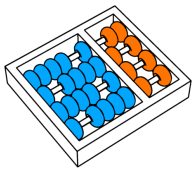
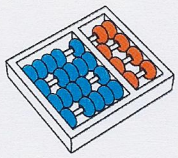


Caio Hoffman

"Análise de desgaste de técnicas de correção de erros em *Phase-Change Memories*"

CAMPINAS
2013



Universidade Estadual de Campinas
Instituto de Computação

Caio Hoffman

"Análise de desgaste de técnicas de correção de erros em *Phase-Change Memories*"

Orientador(a): **Prof. Dr. Guido Costa Souza de Araújo**
Co-Orientador(a): **Prof. Dr. Rodolfo Jardim de Azevedo**

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação do Instituto de Computação da Universidade Estadual de Campinas para obtenção do título de Mestre em Ciência da Computação.

Este exemplar corresponde à versão final da Dissertação defendida por Caio Hoffman, sob orientação de Prof. Dr. Guido Costa Souza de Araújo.

Assinatura do Orientador(a)

CAMPINAS
2013

Ficha catalográfica
Universidade Estadual de Campinas
Biblioteca do Instituto de Matemática, Estatística e Computação Científica
Maria Fabiana Bezerra Muller - CRB 8/6162

H675a

Hoffman, Caio, 1983-
Análise de desgaste de técnicas de correção de erros em phase-change memories / Caio Hoffman. – Campinas, SP : [s.n.], 2013.

Orientador: Guido Costa Souza de Araújo.
Coorientador: Rodolfo Jardim de Azevedo.
Dissertação (mestrado) – Universidade Estadual de Campinas, Instituto de Computação.

1. Memória de mudança de fase. 2. Códigos corretores de erros (Teoria da informação). 3. Memórias resistivas. I. Araújo, Guido Costa Souza de,1962-. II. Azevedo, Rodolfo Jardim de,1974-. III. Universidade Estadual de Campinas. Instituto de Computação. IV. Título.

Informações para Biblioteca Digital

Título em outro idioma: Analysis of wear-out of error correction techniques in phase-change memories
Palavras-chave em inglês:
Phase-change memories
Error-correcting codes (Information theory)
Resistive memories
Área de concentração: Ciência da Computação
Titulação: Mestre em Ciência da Computação
Banca examinadora:
Guido Costa Souza de Araújo [Orientador]
André Rauber Du Bois
Mario Lúcio Côrtes
Data de defesa: 01-07-2013
Programa de Pós-Graduação: Ciência da Computação

TERMO DE APROVAÇÃO

Dissertação Defendida e Aprovada em 01 de Julho de 2013, pela
Banca examinadora composta pelos Professores Doutores:

Prof. Dr. André Rauber Du Bois
CDTEC / UFPEL

Prof. Dr. Mario Lúcio Côrtes
IC / UNICAMP

Prof. Dr. Guido Costa Souza de Araújo
IC / UNICAMP

Instituto de Computação
Universidade Estadual de Campinas

Análise de desgaste de técnicas de correção de erros em *Phase-Change Memories*

Caio Hoffman[1]

01 de julho de 2013

Banca Examinadora:

- Prof. Dr. Guido Costa Souza de Araújo (Orientador)
- Prof. Dr. André Rauber Du Bois
  Centro de Desenvolvimento Tecnológico - UFPel
- Prof. Dr. Mario Lúcio Côrtes
  Instituto de Computação - UNICAMP
- Prof. Dr. Sandro Rigo (Suplente)
  Instituto de Computação - UNICAMP
- Prof. Dr. Wang Jiang Chau (Suplente)
  Escola Politécnica - USP

[1]Bolsista CNPq (processo 133161/2011-0) 2011 – Bolsista FAPESP (processo 2011/05266-3) 2011-2013

Abstract

Phase-change memory brings new opportunities for the electronics industry. Due to projections of high scalability of the fabrication process, PCM is seen as a new main memory in computing systems, replacing the traditional DRAM, whose scale problems require new future technologies that are still unknown.

However, PCM has low endurance when compared with DRAM and robust failure recovery techniques are required to increase its lifetime. To address that, some error correcting techniques have been proposed, based on the non-volatile features of the PCM memories.

In this work, we model and analyze the bit-flip probabilities of five such techniques (ECP, parity, SECDED, SAFER and BCH), in order to evaluate its impact to the wear out of the PCM. Using the bit-flip rate of 15%, obtained experimentally from the execution of the SPEC2006 benchmark, we mathematically modelled and simulated these techniques using both an empirical and theoretical probability rates.

Our results show a clear degradation in techniques that use error-correcting codes, contradicting the previous results in the literature. Only ECP has not shown any degradation. We have also done power analyses of the above listed techniques so as to relate the endurance and the energy required by each technique. Again, the ECP stood out in the results, like SAFER as well. Finally, analytical probabilistic models for ECP and SECDED were proposed and an analysis of PAYG technique (based on ECP's analytical model) was performed.

Resumo

Phase-change memory (PCM) traz novos ensejos para industria eletrônica. Devido às projeções de alta escalabilidade do processo de fabricação da PCM, cogita-se usá-la como memória principal em sistemas de computação, substituindo a tradicional DRAM cujos problemas de miniaturização do processo de fabricação demandam tecnologias ainda desconhecidas.

Contudo, PCM tem problemas de durabilidade e técnicas de recuperação de falhas robustas são extremamente necessárias para recuperação e prologamento do seu tempo de vida, medido em número de escritas. As técnicas mais comuns de recuperação de falhas são os códigos de correção de erros. Porém, outras técnicas de recuperação vêm sendo propostas na literatura, aproveitando as características de não-volatilidade da PCM.

Neste trabalho, usando uma modelagem matemática, analisou-se como a probabilidade de *bit-flip* dos principais códigos de correção de erros – paridade, SECDED e BCH – e das principais técnicas de recuperação de falhas – ECP e SAFER – está relacionada à durabilidade da PCM. A partir da taxa de *bit-flip* medida através da execução do SPEC2006 e por meio dos modelos matemáticos, comparou-se os resultados dos modelos de simulação utilizando-se a probabilidade teórica de 50% e a taxa obtida experimentalmente de 15%.

Os resultados revelaram uma visível degradação da durabilidade dos mecanismos de recuperação de falhas que usam códigos de correção de erros, contradizendo os resultados da literatura. A técnica ECP foi a única que não mostrou degradação. Além disso, uma análise de eficiência energética foi feita, relacionando durabilidade da PCM e o consumo de energia. Novamente, a técnica ECP se destacou nos resultados, como também a técnica SAFER. Finalmente, foram propostos modelos analíticos probabilísticos das técnicas ECP, SECDED e uma análise da técnica PAYG baseada no modelo analítico da ECP.

Aos meus pais, Claudemir e Janete e à minha irmã, Iara, pela família que são.

E à minha namorada, Katia, pelo amor, pela dedicação e pelo companheirismo.

Agradecimentos

Agradeço à FAPESP e ao CNPq pelo fomento de meus estudos e deste projeto de pesquisa.

Agradeço ao professor Rodolfo, por ter dado uma perspectiva nova a este trabalho, provendo conhecimento e boa parte da infraestrutura que permitiram a realização desta dissertação. Obrigado pelas críticas e conselhos sempre muito construtivos.

Agradeço ao professor Guido, por todos esses anos de convivência, dos quais aprendi muito. Sempre muito solícito quando precisei. Até me arrependo de não ter, aproveitado tanto quanto poderia, esse tempo de convivência que tivemos durante o mestrado.

Também não posso esquecer da professora Christiane Neme Campos, agradeço muito por todo apoio que recebi, que começou quando cursei sua disciplina, mas perdura até então. Realmente, tem sido uma convivência importante para mim.

Agradeço ao professor Mario Lúcio Côrtes, por tão construtiva experiência com o PED, com certeza aprendi muito. Também à professora Mariana Rodrigues Motta, pelos esclarecimentos sobre estatística.

Aos colegas do LSC, pelo convívio e, principalmente, pela paciência e vontade de ajudar, sempre que precisei.

Ao pessoal da secretaria do IC, sempre tão prestativos. Também aos pesquisadores Doe H. Yoon e Moinuddin K. Qureshi pela prestatividade, principalmente a este último, que não só respondeu às dúvidas, mas de modo altruísta forneceu informações e o programa para chegar nos resultados que eu não conseguia.

Por fim, aqueles que são mais importante para mim. Agradeço pelo apoio que sempre recebi da minha mãe, do meu pai e da minha irmã. Nunca questionando e sempre apoiando minhas decisões, mesmo que as não entendessem. Agradeço a minha namorada por inúmeras coisas de todos esses anos juntos, mas, em especial, por suas críticas sempre precisas e seu apoio constante nestes dois últimos anos. Agradeço À Deus e aos meus santos protetores, pois forças nunca me faltaram e na solidão nunca fiquei.

"If the fool would persist in his folly he would become wise."

William Blake

Sumário

Lista de Tabelas

Lista de Figuras

Capítulo 1

Introdução

1.1 Motivação

A indústria de componentes e sistemas eletrônicos vem ao longo dos anos superado desafios continuamente; problemas nos processos de fabricação, ao se trabalhar nos limites físicos dos dispositivos eletrônicos, têm sido resolvidos por soluções inovadoras, permitindo o progresso da indústria e a manutenção de custos aceitáveis de produção. Todavia, se para os processos de fabricação já se sabe os próximos passos para manter a escalabilidade da miniaturização – substituição do processo de litografia ótica pela litografia ultravioleta extrema [50] –, bem como um custo competitivo – utilização de *wafers* de 450 mm ao invés das tradicionais 300 mm [49] –, para os dispositivos eletrônicos o futuro não é límpido.

Em particular, a construção de memórias densas e de baixo custo, para suprir a crescente necessidade de armazenamento, é uma das situações mais críticas da indústria. Memórias *Flash* têm superado as péssimas expectativas [51] mantendo sua escalabilidade [5] e, por conseguinte, seu custo por *gigabyte* cada vez menor. Para as memórias dinâmicas (DRAM), as complexas dificuldades de utilização de processos de fabricação abaixo de 30 nm [51] parecem, novamente, superadas pela indústria (protótipos de 25 nm têm sido fabricados [12]), mantendo o custo-benefício dentro do sustentável. Mesmo assim, diversos desafios futuros [51] não trazem tranquilidade à indústria, que precisa de soluções e alternativas à curto prazo.

Esse cenário torna tecnologias emergentes de memória atraentes para pesquisa e desenvolvimento. Não é à toa que nos últimos anos não é difícil de encontrar conteúdo produzido, tanto na literatura quanto na imprensa especializada, sobre tecnologias emergentes de armazenamento como: FeRAM, FETRAM, ReRAM, MRAM, STT-MRAM, PCRAM. Notoriamente, cada tecnologia possui vantagens e desvantagens, que acabam não definindo uma delas, irrefutavelmente, como favorita.

Dessas tecnologias, a PCM têm se destacado pela alta capacidade, densidade e ve-

locidade, sendo cotada para substituição da memória *Flash*, bem como da DRAM [29]. Apesar disso, para que a PCM seja disposta como memória principal em sistemas de computação, é necessário torná-la tão vantajosa quanto a DRAM, o que ainda não ocorreu. Soluções arquiteturais têm sido propostas para os desafios essenciais: redução da latência de escrita e leitura [29,35,38]; redução do consumo de energia dessas operações [11,18,58]; uniformização de desgaste e segurança [36, 40, 45]; e redução do desgaste de células e recuperação de falhas [43, 46]. Todos esses tópicos são analisados de alguma forma neste trabalho, com especial atenção aos dois últimos e com profundidade no último.

A PCM é constituída por células construídas com a utilização de um material calcogênico, que pode assumir dois estados diferentes nos quais suas resistividades diferem de várias ordens de grandeza, possibilitando diferenciá-los eletricamente. As mudanças de estado acontecem por meio do aquecimento do material, ao se aplicar pulsos de corrente elétrica. Esse estresse termomecânico, ocorrendo milhares de vezes, causa rupturas nos constituintes da célula, levando-a permanentemente a um desses estados. Quando isso acontece, há impossibilidade de escrita na linha de memória que possui tal célula e, consequentemente, a página não pode ser escrita, levando à falha do sistema de memória.

Portanto, mecanismos de recuperação de falhas são cruciais em PCM. Uma das formas mais comuns de recuperação de falhas é a utilização de código de correção de erros, já comumente usados em memórias *Flash* e DRAM. Contudo, hipoteticamente, esses códigos trazem além dos benefícios esperados (isto é, recuperação de falhas), também malefícios para PCM: visto que a escrita de uma nova palavra de dados numa linha de memória, mesmo que difira de apenas um *bit* da anterior, gerará um novo código de verificação para correção de erros, que pode ter vários *bits* diferentes do código anterior. Logo, pode haver um desgaste maior nos *bits* que armazenam o código de verificação, uma vez que sempre tanto a palavra de dados quanto o código de verificação são gravados na linha.

1.2 Objetivos

Essa hipótese – baseada no argumento de que códigos de correção de erros aceleram o desgaste de memória resistivas, apresentado por Schechter et al. em [43] – é avaliada neste trabalho de maneira aprofundada, através da elaboração de modelos matemáticos e de simulação, das principais técnicas de correção de erros divulgadas na literatura, que utilizam ou não códigos de correção de erros.

Além disso, a construção de modelos analíticos probabilísticos dessas técnicas estudadas, é um dos focos de estudo deste trabalho, visando acelerar as avaliações dessas técnicas de maneira mais eficiente daquela proporcionada pelos modelos de simulação utilizados na literatura.

1.3 Organização do trabalho

Este trabalho está organizado da seguinte forma. No Capítulo 2, aprofunda-se os conceitos já enunciados nesta introdução sobre PCM, elencando os principais problemas e trabalhos divulgados na literatura para suavizá-los ou eliminá-los. Além disso, fornece-se uma visão geral das outras tecnologias de memória.

No Capítulo 3 é apresentada, conceitualmente, a probabilidade de *bit-flip*, o fator para avaliação das principais técnicas de recuperação de falhas em PCM – ECP, DRM, SECDED, SAFER e FREE-p. As assunções do modelo matemático no qual o simulador das técnicas está embasado, como também a avaliação das simulações dessas técnicas, são analisadas e discutidas ao longo do capítulo. Finalizando-o, estão descritos os modelos analíticos construídos com sucesso e é feita uma análise dos casos que não resultaram em modelos viáveis, avaliando as razões que levaram a isso.

A probabilidade de *bit-flip* experimental é descrita e analisada no Capítulo 4. Neste capítulo, usam-se os *benchmarks* do SPEC2006 para se obter a estatística da probabilidade de *bit-flip*. Além disso, é elaborado um coeficiente de relação entre o tempo de vida da memória e o consumo energético da escrita, para avaliar quais técnicas são mais eficientes em ambos os critérios. O capítulo é finalizado com uma avaliação da técnica PAYG, baseada no modelo analítico da técnica ECP_6.

Finalmente, no Capítulo 5 são apontados as conclusões deste trabalho, além do que pode ser feito em trabalhos futuros.

1.4 Contribuições

- Modelagem da probabilidade *bit-flip* para uma avaliação mais realista da influência dos códigos de verificação na duração de uma PCM, para cinco técnicas de correção de erros da literatura: ECP, DRM, SECDED, SAFER e FREE-p.
- Aprimoramento do modelos de simulação utilizados na literatura para avaliações de durabilidade de memória, com o uso da modelagem matemática da probabilidade de *bit-flip*.
- Avaliação do consumo energético para escrita em memória PCM para cada uma das técnicas estudadas, o que não foi feito originalmente na publicação dessas técnicas.
- Construção de modelos analíticos probabilísticos para as técnicas ECP e SECDED, com a apresentação de uma nova modelagem, por meio do problema (ou paradoxo) do aniversário generalizado.

Capítulo 2

Conceitos e Estado da Arte

Por conta da necessidade de manter a constante miniaturização de componentes eletrônicos, permitindo oferecer produtos cada vez mais complexos a custo satisfatório, a indústria lida com dois vieses problemáticos: como fabricar e como projetar. Atualmente, na fabricação, usa-se o processo de litografia ótica [53] cuja precisão vem se tornando um problema para geometrias muito pequenas (< 20 nm), requerendo a reaplicação do processo mais de uma vez [50]. O que, logicamente, encarece a fabricação. A solução futura a ser adotada é a litografia ultravioleta extrema[1]. Contudo, ainda há um bom caminho a se percorrer nesta área, uma vez que o número de *wafers*[2] processadas por hora nessa tecnologia ainda é pequeno para comercialização em massa [13].

Com os custos de fabricação crescendo à medida que a escala dos componentes reduz, aumentar a produção é essencial. Por isso, a substituição de *wafers* com diâmetro de 300 mm para as de 450 mm é outro aspecto relevante nos desafios a serem enfrentados pela indústria. Embora sua implementação seja cara, pois exige toda a troca do maquinário atual, é um passo tecnológico necessário para a continuidade desse processo de miniaturização dos componentes eletrônicos. Todavia, essas soluções são conhecidas há algum tempo e, provavelmente, com um pouco mais de pesquisa e teste, logo estarão no mercado. Situação que não é equivalente para o projeto de circuitos e componentes eletrônicos em geometrias extremas. Uma vez que nessas dimensões pode-se atingir o limite físico de um material, sobrando duas opções: construí-lo de uma maneira diferente ou substitui-lo, sendo que pode-se desconhecer ambas as opções.

Dos principais componentes eletrônicos da indústria, as memórias que funcionam baseadas em armazenamento de carga elétrica se encontram com desafios complicadíssimos de projeto, para continuarem a serem fabricadas em processos de fabricação abaixo dos 45 nm [51]. As memórias *Flash* e dinâmica (DRAM) armazenam informação por meio de

[1] Tradução literal para *extreme ultraviolet lithography*.

[2] Substrato cristalino circular de silício utilizado na fabricação de circuitos eletrônicos.

carga elétrica e estão entre os componentes eletrônicos mais comercializados. Nesta era atual, a quantidade e a velocidade requiridas para se armazenar e transferir dados crescem ininterruptamente. Por conseguinte, é uma condição essencial que a indústria continue a construir, nos próximos anos, memórias mais densas, rápidas e de baixo custo. Na próxima seção apresenta-se mais detalhe sobre as memórias *Flash* e DRAM, discutindo seus mecanismos de funcionamento e os problemas de projeto atuais.

2.1 Principais tecnologias de memória atuais

Memórias geralmente são construídas no formato de matriz na qual as linhas e colunas são compostas por fios metálicos chamados de *word-line* e *bit-line*, respectivamente. Os elementos da matriz são os elementos de memória, comumente compostos por um dispositivo de armazenamento (transistor ou doido) e um dispositivo de acesso, veja esta organização na Figura 2.1. Em alguns tipos de memórias consegue-se acoplar os dois dispositivos em um único.

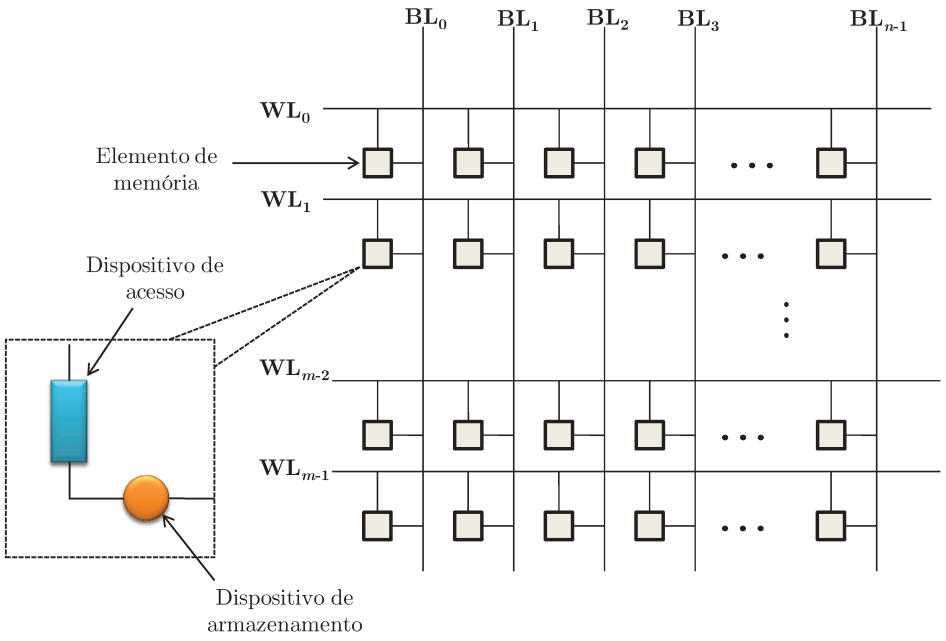


Figura 2.1: Organização genérica de uma memória com $m \cdot n$ elementos de memória, distribuídos por n *bit-lines* e m *word-lines*. Adaptada de [39].

As *word-lines* são acionadas a partir do resultado de decodificação de um endereço de memória; quando isso acontece, os elementos de memória conectados à *word-lines* podem ser manipulados através das *bit-lines*: por exemplo, em uma escrita, a *bit-line* recebe uma

alta injeção de corrente para alterar o estado do elemento de memória ou, em uma leitura, ela pode ser posta em um valor de tensão na qual o elemento de memória pode abaixá-lo ou mantê-lo, dependendo do valor de dados que armazena (ou estado que se encontra).

Os elementos de memória podem ser construídos de diversas formas e não há necessidade de haver sempre um dispositivo de armazenamento e um de acesso. Na memória *Flash*, descrita a seguir, não existe essa separação. Em geral, a função do dispositivo de acesso é isolar o dispositivo de armazenamento, evitando que este seja modificado não intencionalmente.

2.1.1 Memória *Flash*

Memórias *Flash* – ou EEPROM *Flash* – são formadas por células que se constituem de um único transistor MOS com dois *gates*, um trabalhando convencionalmente (chamado de *control gate*) e outro interno ao transistor e próximo do canal, servindo de armadilha de elétrons (chamado de *floating gate*). Na Figura 2.2, é visto o esquemático desse transistor, construído de forma similar a um nMOS.

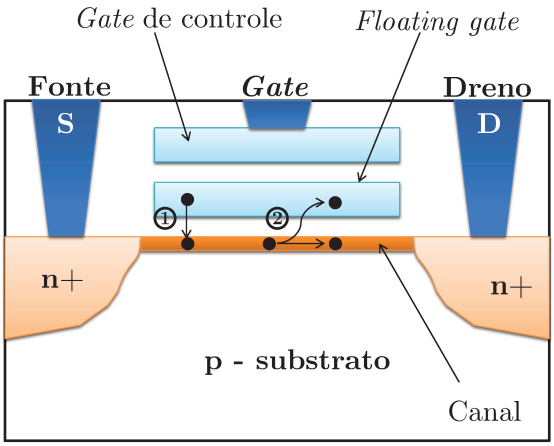


Figura 2.2: Transistor nMOS com duplo *gate* que constitui uma célula de memória *Flash*, adaptado de [34].

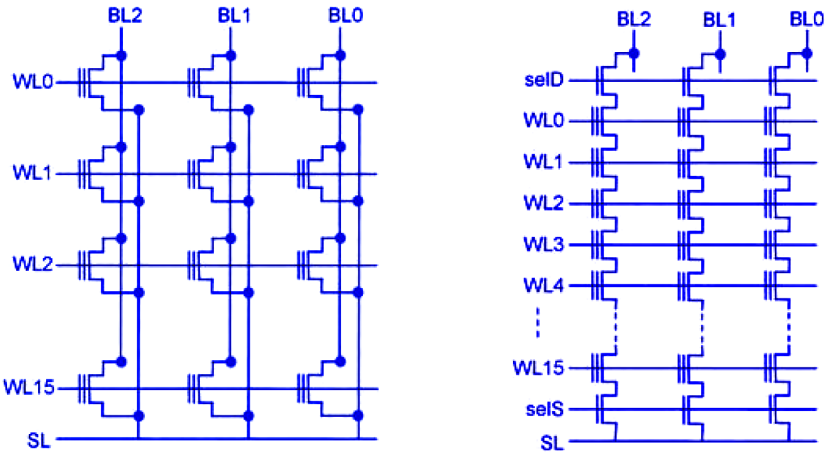
Existem duas formas de programar (escrever) uma célula de memória *Flash*: usando CHEI[3] ou o mecanismo de tunelamento de Fowler-Nordheim. Quando as células são conectadas individualmente às *bit-lines* e *word-lines* – tipo NOR, Figura 2.3(a) – a programação é feita usando CHEI; quando um conjuntos de transistores são ligados serialmente e compartilham uma mesma *bit-line* – tipo NAND, Figura 2.3(b) – a escrita é por tunelamento. Para ambos os tipos, a operação de apagar é feita através do tunelamento

[3]Sigla inglesa para *channel hot-electron injection*.

de Fowler-Nordheim. Como esse de processo de apagar é difícil de controlar individualmente, apaga-se um conjunto de células (como uma página, ou bloco ou setor) de uma vez. Essa instantaneidade da operação assemelha-se a um *flash*, por exemplo, de uma máquina fotográfica, por isso o nome de memórias *Flash* [34].

As operações de apagar e programar estão ilustradas na Figura 2.2 através da descrição do movimento dos elétrons denotados pelos pontos pretos. À esquerda do centro da figura (indicado em 1) pode-se ver um elétron movendo-se verticalmente do *floating gate* para o canal, característica do mecanismo de tunelamento de Fowler-Nordheim para desprogramação da célula. Nesse caso, V_{gs} e V_{gd} – diferença de tensão entre o *gate* e a fonte e entre o *gate* e o dreno, respectivamente – são negativas, por exemplo, -12 V. Isso repele os elétrons do *floating gate*. Outrossim, a programação via tunelamento ocorre com uma diferença de tensão da mesma magnitude, porém positiva.

Praticamente no centro da Figura 2.2 (indicado em 2) , observa-se a característica do mecanismo de injeção de elétrons (CHEI) para programação da célula. Embora parecido com o tunelamento, a diferença reside que os elétrons estão em fluxo contínuo da fonte para o dreno quando sofrem tunelamento. Em termos de tensões dos sinais elétricos, $V_{gd} < V_{gs}$, tal como $V_{gd} = 6V$ e $V_{gs} = 12V$. Note que, utilizando CHEI, nem todo elétron que perpassa o canal é capturado.



(a) Organização NOR (b) Organização NAND

Figura 2.3: Duas organizações de memória *Flash*, originais de [34]. SL (*source-line*) provê uma tensão de referência*, selD e selS são sinais que acionam os transistores de amplificação dos sinais aplicados, respectivamente, ao dreno e à fonte.

*Embora seja comum que os dispositivos de armazenamento estejam conectados entre a *bit-line* e o terra, alguns projetos de circuitos não conseguem fornecer a tensão 0 V, por isso, fornece-se uma tensão

Células de memória *Flash* programadas, possuem o *floating gate* carregado de elétrons e têm V_T (voltagem de *threshold*) aumentada, de forma que a diferença de tensão aplicada entre a *word-line*, conectada ao *gate*, e a *bit-line*, conectada ao dreno do transistor, precisem ser maior do que a tensão de *threshold*, isto é, $V_{gd} > V_T$. À medida que o *floating gate* perde cargas, V_T diminui, até ficar negativo. Nesse ponto a célula está sempre ativa, porquanto qualquer tensão (positiva) é maior do que V_T. Neste estado, é dito que a célula está apagada (*erased*).

Na Figura 2.3(a) é visto na organização NOR, que as células de memória são individuais, fato inerente ao uso de CHEI para escrita, pois este não provoca pertubação nas células vizinhas, como na utilização do tunelamento de Fowler-Nordheim [34]. Na NOR-*Flash*, a leitura é realizada aplicando-se um sinal elétrico de baixa tensão (caracterizando o nível lógico zero) à *word-line* desejada e um sinal de tensão de mais alta (caracterizando o nível lógico um) à *bit-line*. Se a célula estiver programada, a tensão na *word-line* não é suficiente para formar o canal entre a fonte e o dreno, dessa forma, a tensão na *bit-line* permanecerá inalterada. Contrariamente, se a célula estiver ativa, haverá fluxo de elétrons pelo canal, portanto, implicando uma queda de tensão na *bit-line*.

Para a organização NAND (Figura 2.3(b)), mais compacta, as células são tanto programadas quanto apagadas em blocos através de tunelamento [39]. Por conta de sua estrutura, a leitura difere daquela feita na NOR-*Flash*. Nesse caso, todas *word-lines* têm sua tensão elevada ao nível lógico um, à exceção daquela que deseja-se ler. Dessa forma, nessas células, independente de V_T, todas terão um canal conectando a fonte ao dreno, sendo consideradas como curto-circuitadas. A célula alvo é quem poderá influenciar na tensão aplicada à *bit-line*, *i.e.*, se programada, manterá a tensão, senão, abaixa-la-á.

Pela repetição das operações de apagar e escrever, alguns elétrons acabam ficando presos no óxido. Com o tempo o número de cargas acumuladas passam a manter V_T constante, impedindo que a célula forneça uma leitura confiável. Esse problema limita a durabilidade – também chamado de tempo de vida – de uma célula de memória *Flash* [33]. A durabilidade média desta célula é da ordem de 10^5 à 10^6 ciclos de programação e desprogramação [34].

Uma das características que faz a memória *Flash* tão utilizada atualmente, além de sua alta densidade (um transistor por célula), é sua capacidade de armazenar mais do que um *bit* por célula, as chamadas células MLC (*multi-level cell*) ou MLB (*multi-level bit*). Entretanto, com a miniaturização, independentemente do tipo de célula MLC ou SLC[4] (*single-level cell*), existem desafios de sensoriamento[5] pela diminuição das cargas

de referência através da *source-line*, da qual todas as outras tensões maiores do que ela serão consideras positivas e, complementarmente, todas as menores, negativas.

[4]Um *bit* por célula.

[5]O sensoriamento consiste na amplificação do sinal das *bit-lines* até valores nos quais é possível diferenciá-los em nível lógico. Para memórias que armazenam vários *bits* o sensoriamento é um pouco

armazenadas no *floating-gate*.

Como divulgado em [5], algumas células produzidas com 20 nm possuíam apenas 20 elétrons caracterizando os diferentes estados programados e desprogramados. Um valor pequeno que exige um projeto sofisticado para identificação dos estados, além de permitir que um número pequeno de cargas presas no óxido isolante possam minar a confiabilidade de leitura da célula. Claramente, números menores de carga nas células podem inviabilizar qualquer identificação de estado, atingindo um limitante físico para a engenharia atual.

2.1.2 DRAM

Sempre que este trabalho se referir à DRAM, estará, na verdade, referindo-se à *Synchronous Dynamic Random-Access Memory* (SDRAM) que são a forma de memória dinâmica mais utilizada atualmente. Uma célula DRAM é constituída basicamente por um transistor e um capacitor, comumente denotada por 1T1C, esquemático exibido na Figura 2.4. Em termos de construção, a célula é bem similar a um transistor nMOS à exceção do capacitor[6], conforme Figura 2.5.

A escrita em uma célula DRAM é bem simples, quando ela é ativada pela *word-line*, o sinal proveniente da *bit-line* irá, ou carregar o capacitor vazio até que este atinja uma tensão igual à *bit-line*, ou descarregá-lo, necessitando que a *bit-line* tenha uma tensão menor do que o capacitor. Já a leitura de uma célula se dá com a ativação do *gate* pela *word-line*, a *bit-line* é posta em uma tensão intermediária entre o nível de tensão considerado 0 lógico e aquele considerado 1 lógico. Se o capacitor estiver carregado, ele irá descarregar por meio da *bit-line* elevando a tensão da mesma. Caso o capacitor esteja descarregado, a tensão intermediária da *bit-line* irá carregar o capacitor de maneira parcial, rebaixando a tensão de intermediária para um nível menor.

Pode-se perceber que a leitura na DRAM é destrutiva: se o capacitor estiver carregado, perderá sua carga; se estiver descarregado, receberá carga. Logo, é necessário que toda leitura de célula seja seguida de uma escrita, para restaurar os valores de carga anteriormente retidos na célula. Além disso, um problema comum é existência de corrente de fuga (*leakage current*) [24] nos capacitores, o que reduz a quantidade de carga armazenada e requer recargas[7] periódicas, para não prejudicar o sensoriamento das *bit-lines*, mesmo quando uma linha de memória não tenha sido utilizada por um longo tempo. Com a mi-

mais complexo [34] envolvendo tensões, correntes ou resistências de referência para se obter o nível lógico correto.

[6]Ao longo dos anos a forma de se construir uma célula DRAM se modificou para se adequar ao escalonamento das geometrias de cada novo processo de fabricação. No passado, fora utilizado o capacitor de fosso. Atualmente, usa-se o capacitor de pilha (*stack capacitor*), que é fabricado com o empilhamento dos materiais condutores e isolantes, daí, o nome de pilha. Para continuar a miniaturização, uma nova forma de fabricação do capacitor poderá ser utilizada, no formato de pilar (*pillar capacitor*) [51].

[7]Em referência ao termo em inglês *refreshes*.

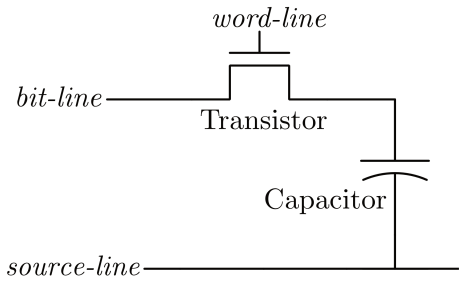


Figura 2.4: Circuito representativo de uma célula DRAM.

niaturização das células de DRAM, a quantidade de carga capaz de ser armazenada pelo capacitor vem reduzindo, tornando mais crítico o problema de retenção e aumentando a frequência de *refreshes*, o que, por sua vez, aumenta o consumo energético [29].

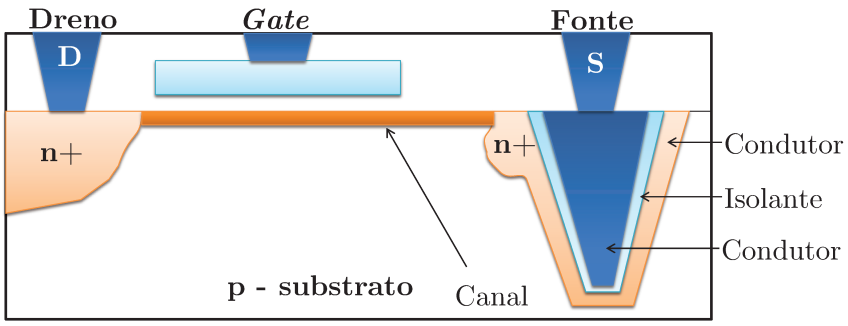


Figura 2.5: Esquemático de uma célula DRAM projetada com o chamado capacitor de fosso (*trench capacitor*). Adaptado de [34].

Uma observação importante relacionada à DRAM é que sua durabilidade é maior do que 10^{15} escritas [7, 59], considerada ilimitada para efeitos práticos [43]. Portanto, essa questão não é um problema. Além da escrita a cada leitura e da escrita de *refresh*, existe uma terceira escrita necessária: na DRAM implementa-se um mecanismo chamado de *scrub*, cuja função é temporariamente verificar se existem células que não possuem o valor correto e corrigi-las. Esses erros podem ocorrer quando há um decaimento significativo de cargas do capacitor entre dois *refreshes*, pois a leitura do valor degenerado de uma célula pelo mecanismo de *refresh* levará a restauração de um valor incorreto pelo segundo *refresh*.

A despeito de existirem, atualmente, soluções para manter o tempo de retenção de cargas nas células DRAM adequado, com a escalabilidade dos processos de fabricação, a busca por grande capacidade de armazenamento requisitará o descobrimento de novas tecnologias capazes de lidar com este problema [51].

2.2 Tecnologias de memórias não-voláteis emergentes

Espera-se que, nos próximos anos, novas tecnologias supram a crescente necessidade de armazenamento de informação. É desejável que, além de compatíveis com o tradicional processo de fabricação CMOS, idealmente, estas memórias exibam algumas qualidades: alta densidade (*terabits*/cm^2); não-volatilidade (mais do que 20 anos de retenção dos dados); ciclos de leitura e escrita com poucos nanossegundos; alta durabilidade (milhões de bilhões de ciclos de leitura e escrita); baixo consumo de energia (menor do que qualquer tecnologia atual); e baixo custo [34].

Esta seção cita as principais tecnologias de memórias emergentes, explicando de maneira sucinta seus funcionamentos, qualidades e desafios a serem enfrentados.

2.2.1 ReRAM

Resistive RAM (RAM resistiva) embora seja uma definição muito genérica na qual diversos tipo de componentes eletrônicos se encaixam, atualmente é quase univocamente referenciada como a memória constituída de Memristors. Memristor, abreviação de *memory resistor*, é considerado o quarto elemento fundamental da teoria de circuitos elétricos. Previsto teoricamente em 1971 e construído recentemente nos laboratórios da HP [2], podem funcionar como dispositivos de memória não-volátil.

Existem diversos materiais que podem ser usados para construir um memristor que é constituído de um material isolante colocado entre dois eletrodos. Seu funcionamento depende de uma alta tensão, ou corrente, ou alto campo elétrico para polarizar o material isolante de modo à conduzir ou não. Os materiais constituintes são agrupados em categorias dependendo do mecanismo de mudança de estado [39]; por exemplo, para oxidação e redução química, um dos materiais isolantes comumente usado é o dióxido de titânio (TiO_2) [21, 39, 59].

O núcleo do dispositivo de armazenamento, que está entre dois eletrodos (Figura 2.6), tem duas regiões: dopada e não-dopada. A região dopada de comprimento w, pode se estender até o comprimento D de todo o núcleo, bem como pode ter comprimento praticamente nulo. O número de dopantes (TiO_{2-x}) aumenta ou diminui dependendo da tensão aplicada, que provoca oxidação ou redução dos íons com a movimentação dos elétrons de um eletrodo ao outro. Quando praticamente todo o comprimento D é ocupado por TiO_{2-x} (condutor [39]) há baixa resistividade na região nuclear [21]. Outrossim, quando o comprimento D é ocupado por TiO_2 (não-condutor) há alta resistividade na região nuclear.

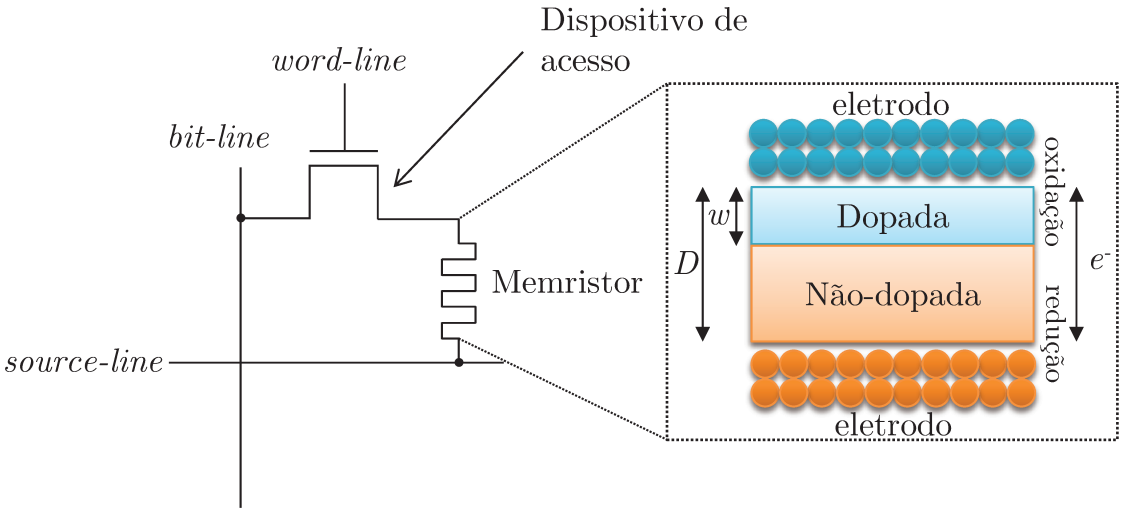


Figura 2.6: O circuito e o dispositivo de armazenamento (memristor) são ilustrações com base nas informações retiradas de [2, 21, 39, 59].

A escrita pode ser visualizada (por meio da Figura 2.6) da seguinte forma: depois da ativação do dispositivo de acesso, dependendo dos sinais aplicados à *bit-line* e *source-line*, haverá migração de elétrons de um eletrodo ao outro. Dependendo do tipo de material que constitui cada eletrodo, o eletrodo inferior (alaranjado) pode ser visto como um cátodo e o superior (azul) como um ânodo. Quanto mais este receber elétrons, mais haverá moléculas de TiO_{2-x}, que são deficientes de átomos de oxigênio e altamente condutoras. Já aquele, ao receber elétrons, provoca o processo inverso de redução das moléculas oxidadas, aumentando o número de TiO_2, altamente resistivas. Na leitura, o sinal aplicado à *bit-line* terá a tensão variada, dependendo se as cargas elétricas terão ou não facilidade em perpassar o memristor. Tal variação será identificada pelo circuito de sensoriamento. Como a leitura não estimula nem a oxidação, nem a redução, ela não é destrutiva; além disso, os elétrons migrados para um eletrodo, por lá ficarão até serem repelidos por um forte campo elétrico, garantindo às células não-volatilidade.

Uma das grandes vantagens do memristor é a área de sua célula, muito pequena em relação às outras tecnologias, mesmo quando comparado àquelas atualmente no mercado [39, 59]. Embora a durabilidade – contabilizada por ciclos de programação e desprogramação da célula – divirja nalgumas fontes na literatura, estando entre 10^4 [2] e 10^{12} [59], memristors são sem dúvida fortes candidatos a se estabelecerem como uma nova tecnologia de armazenamento, principalmente, pelas células diminutas e pela possibilidade de escalar o processo de fabricação [59], bem como a utilização de células MLC [39].

2.2.2 FeRAM e FETRAM

Ferroeletric RAM (RAM ferroelétrica) é um tipo de memória cujo o dispositivo de armazenamento é um capacitor, assim como na célula de DRAM (1T-1C), mas ao contrário desta, o material utilizado entre as placas condutoras é classificado como ferroelétrico. A vantagem desse material sobre o dielétrico é que a polarização permanece quando não há campo elétrico, dando a característica de não-volatilidade ao material [55]. Apesar do nome, o material ferroelétrico não é constituído de ferro, mas de cristais como PZT (Titanato Zicornato de Chumbo) que apresentam histerese sob campos elétricos, como materiais ferromagnéticos sob campos magnéticos, portanto, a similaridade do nome.

Observando a Figura 2.7(a) de um elemento de memória de uma FeRAM, a escrita é feita enviando sinais tanto pela *bit-line* quanto pela *sense-line*[8], dependendo do valor que se deseja armazenar. Após o dispositivo de acesso ser acionado pela *word-line*, a escrita é feita, ou aplicando pulso de tensão na *bit-line* e polarizando o material do capacitor em uma orientação, ou na *sense-line* e polarizando o material na orientação inversa. A leitura é feita aplicando uma tensão sobre a *sense-line* [39] (ou *bit-line* [34]), após a *word-line* ativar o transistor. Se a polaridade no capacitor estiver de acordo com a tensão da *sense-line*, uma corrente muito pequena flui pelo capacitor, se a polaridade for discordante, a tensão da *sense-line* irá inverter a polarização atual, fazendo um grande fluxo de corrente fluir e ser detectada pelo circuito de sensoriamento. Tanto a leitura quanto a escrita podem ser realizadas abaixo dos 50 ns [39].

O grande inconveniente da célula da FeRAM é que, igualmente à DRAM, a leitura é destrutiva, visto que em ambos os casos explicados acima, a leitura resulta em uma mesma polarização, requerendo uma escrita após a leitura para recuperar os valores armazenados. Embora a durabilidade da célula esteja por volta de 10^{12} ciclos de escrita, por conta da leitura destrutiva, o tempo de vida efetivo[9] das células é menor. A FeRAM também enfrenta problemas de miniaturização para novos processos de fabricação, por conta da área necessária para manter um número de cargas que satisfaça o sensoriamento e pela alta temperatura necessária para cristalizar os materiais ferroelétricos. Apesar disso, existem aplicações reais com a FeRAM como memória em etiquetas RFID [39].

Das e Appenzeller publicaram em [16] um novo tipo de memória chamada FETRAM (FeFET-RAM[10]). No lugar de um capacitor ferroelétrico, tem-se um transistor ferroelétrico (célula de memória 1T-1T). O transistor parece-se como um transistor nMOS comum, à exceção do isolante entre o *gate* e o canal, que é um material ferroelétrico. Também há uma certa similaridade com o transistor utilizado na memória *Flash*, no que concerne as operações de leitura e escrita.

[8] Também denotada como *plate-line* [39].

[9] Isto é, número de operações de escrita requisitadas à memória pelo sistema.

[10] Sigla em inglês para *ferroelectric field-effect transistor RAM*.

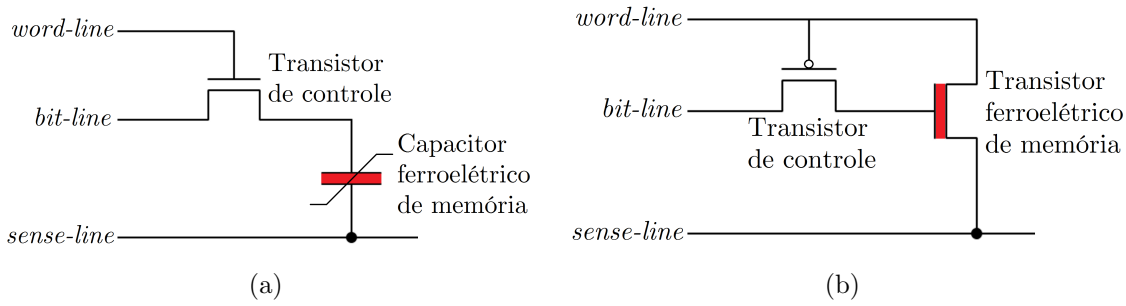


Figura 2.7: (a) Circuito representativo do elemento de memória FeRAM. (b) Circuito representativo do elemento de memória FETRAM[†]. Figuras adaptadas de [16].

Na Figura 2.7(b) é mostrado um esquemático do circuito do elemento de memória. Note que a *bit-line* no transistor ferroelétrico está ligada ao *gate*. Quando a *word-line* possui tensão baixa, o transistor de controle está ativo e o transistor ferroelétrico está desativado, porque dreno e fonte – conectado à *sense-line* – estão na mesma tensão. Com um pulso elétrico na *bit-line*, orienta-se a polarização do material ferroelétrico, conforme a tensão do pulso: positiva ou negativa. A leitura é executada elevando a tensão na *word-line* a qual, embora desative o transistor de controle e corte o efeito da *bit-line*, permite um fluxo de corrente pelo transistor de memória. Esse fluxo será alto ou baixo, dependendo da polarização do ferroelétrico. Assim, a variação será mensurada pelo circuito de sensoriamento através da *sense-line*.

Deve-se notar que, diferentemente da célula de uma FeRAM, a leitura não é destrutiva, pois a corrente que flui pelo capacitor na leitura não altera a polarização do material ferroelétrico. Informações importantes como energia e latência das operações de leitura e escrita, durabilidade e escalabilidade do processo de fabricação ficaram faltando no artigo de Das e Appenzenller. Contudo, se as propriedades como durabilidade e a latência de escrita e leitura forem de iguais qualidades à célula da FeRAM, a FETRAM tem um futuro promissor. Além disso, o material utilizado no transistor de memória, o ferroelétrico orgânico PVDF-TrFE, possui baixa temperatura de cristalização, o que é uma vantagem no seu processo de fabricação.

2.2.3 MRAM e STT-MRAM

Magnetic RAM tem como dispositivo de armazenamento um material isolante, como o óxido de alumínio (Al_2O_3), posicionado entre duas placas de material ferromagnético,

[†]No artigo [16], não é utilizado um pMOS e sim um transistor nMOS em *deplation mode*, isto é, tem V_T negativa. Contudo, pela descrição do funcionamento do circuito percebe-se que há uma incoerência, uma vez que é descrito que se utiliza de 15 V para desligar o transistor de controle, o que discorda com a teoria [44, 53].

denominado MTJ (*magnetic tunnel junction*). O MTJ pode ser visto de maneira simplificada[11] na Figura 2.8. A camada fixa é mantida sempre com uma orientação magnética fixa. A camada livre pode receber duas orientações magnéticas: uma oposta (antiparalela) à camada fixa e outra direta (paralela). Quando as camada superior e inferior estão igualmente (paralelamente) polarizadas, a passagem sobre o isolante por tunelamento necessita de baixa energia, o que representa um maior fluxo de cargas e pode ser visto com uma baixa resistividade do material. Já quando as camadas superior e inferior estão diferentemente (antiparalelamente) polarizadas, o tunelamento pelo isolante requer alta energia, representando baixo fluxo de cargas, logo, alta resistividade.

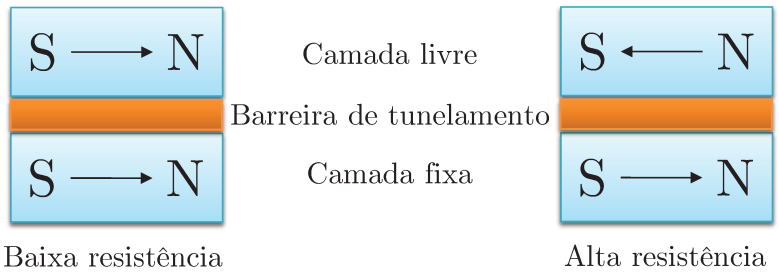


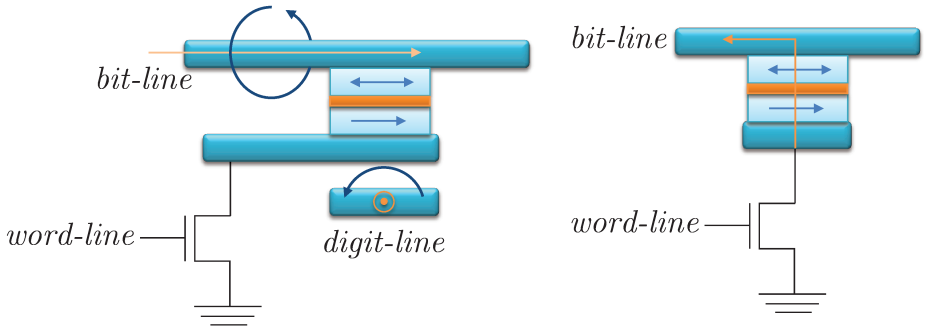
Figura 2.8: Esquemático simplificado do MTJ com as três principais camadas: *free layer*, camada livre; *tunnel barrier*, barreira de tunelamento; e *fixed layer*, camada fixa.

O elemento de memória de uma MRAM pode ser visto na Figura 2.9(a). Com a *word-line* ativando o dispositivo de acesso, a leitura na célula é feita medindo fluxo de corrente – ou a resistência – que acomete a *bit-line*, atráves do circuito de sensoriamento. Aplicando uma tensão na *bit-line*, haverá pouco ou nenhum fluxo de cargas, se o MTJ estiver com material em orientação antiparalela, ou manterá um alto fluxo de corrente pela *bit-line*, se o MTJ estiver com orientação paralela.

A escrita é feita desativando o transistor de acesso e, dessa forma, o fluxo de corrente pela *bit-line* não perpassa pelo MTJ. Como a variação de corrente elétrica produz um campo magnético, a corrente da *bit-line* ou da *digit-line* são responsáveis pela orientação dos materiais ferromagnéticos. A *digit-line* [34] (ou *write word-line* [39]) é acionada para polarizar a camada fixa, sempre na mesma direção. A polarização da camada livre fica a cargo do sinal aplicado à *bit-line*.

A MRAM possui muitas qualidades: a durabilidade de uma célula MRAM ultrapassa

[11] É comum haver de 8 à 10 camadas de material ferromagnético, anti-ferromagnético, dielétrico, entre outros compondo a pilha de camadas que formam o MTJ [39].



(a) Elemento de memória MRAM.

(b) Elemento de memória STT-MRAM.

Figura 2.9: Figuras adaptadas de [26].

a capacidade de 10^{15} escritas [7]; nota-se que a corrente de leitura que flui pelo MTJ não interfere na orientação magnética dos materiais ferromagnéticos. Logo, a leitura não é destrutiva; soma-se o fato que a orientação dos materiais ferromagnéticos são permanentes, o que garante não-volatilidade à memória; e, por fim, tanto o tempo de leitura e escrita estão abaixo dos 10 ns.

Não obstante, a MRAM possui células muito grandes (área 4 vezes maior do que uma célula DRAM produzida na mesma escala do processo de fabricação [7]) e consome alta quantidade de energia na escrita [39]. Por isso, uma nova forma de programar o MTJ, utilizando o efeito STT (*spin-transfer torque*) tem chamado muito a atenção. O efeito STT é a polarização do material ferromagnético através da passagem de uma corrente com cargas de *spin* polarizado, ocorrendo transmissão ou reflexão [39] do *spin* dos elétrons para o material ferromagnético, causando orientação magnética do material.

Pela célula da Figura 2.9(b), nota-se que a *digit-line* não é mais necessária para programar. Na escrita, basta que o dispositivo de acesso esteja ativo e a corrente a passar pelo MTJ esteja polarizada[12]. Como mostrado na figura – pela linha alaranjada que passa ao alongo do MTJ –, uma corrente de elétrons de *spin* polarizado fluem do mais baixo potencial para o maior, polarizando a camada livre (superior) do dispositivo de armazenamento. Aqui, o mais baixo potencial é *bit-line*, por isso a corrente flui em sua direção. Já a leitura na STT-MRAM é feita de maneira análoga à MRAM.

A área da STT-MRAM é significativamente menor quando comparada com a MRAM, e mantém bons tempos de leitura e escrita [7]. Já a durabilidade pode ser menor, devido ao estresse da passagem da corrente de escrita pelo material [51], mas se mantém alta [39]. A STT-MRAM tem grande potencial, muito embora ainda sejam necessários muitos

[12]Isto pode feito passando a corrente através de uma fina camada de material magnetizado [57].

esforços de engenharia para se adequar à indústria: escalabilidade para atingir custo por *bit* competitivo; redução do tamanho da célula, limitado pelo transistor que deve prover a larga corrente necessária para escrita [48]; entre outros problemas de fabricação [39].

2.3 PCM

Memória de mudança de fase, *Phase-change memory*, também chamada de OUM (*ovinic unified memory*[13]), de PCRAM ou de PRAM – ambas as siglas significam *Phase-change random-access memory* –, é uma tecnologia emergente de memória, como aquelas descritas na seção anterior, e que tem chamado a atenção por conta da baixa latência de leitura (e escrita, se comparada à *Flash*), da boa durabilidade e da escalabilidade para processos de fabricação nanométricos. Sua qualidades a fazem não apenas uma potencial substituta das memórias *Flash*, como também das memórias dinâmicas. Esta possibilidade fomentou ótimos trabalhos de pesquisa nos últimos anos [23, 29, 40, 43].

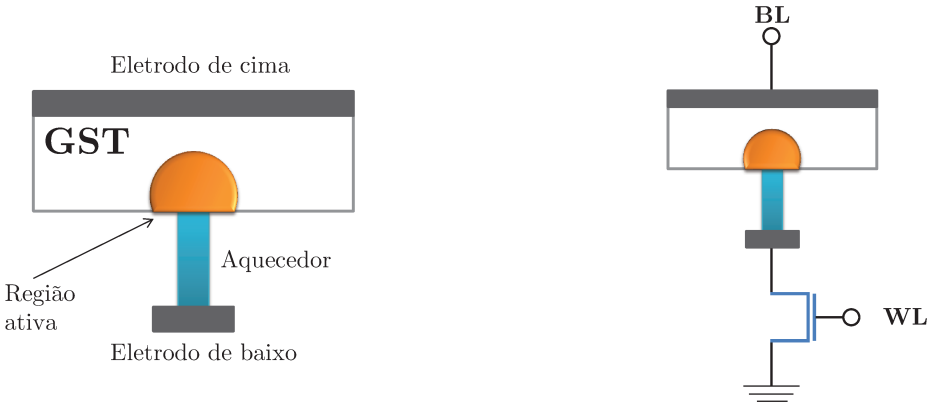
Um elemento de memória PCM consiste de uma liga calcogênica[14] formada, em geral, por germânio, antimônio e telúrio, $Ge_xSb_yTe_z$ (GST), que pode exibir dois estados físicos com características completamente distintas, podendo-se alternar entre eles facilmente[15]. O estado amorfo (ou fase amorfa) é caracterizado por baixa reflexividade ótica e alta resistividade elétrica (como o vidro), enquanto o estado cristalino (ou fase (poli)cristalina) possui alta reflexividade e baixa resistividade. Embora a mudança na reflexividade possa ser tanto quanto aproximadamente 30%, a mudança na resistividade chega a cinco ordens de grandeza (10.000.000%) [42]. A descoberta do calcogênico $Ge_2Sb_2Te_5$ diminuiu o tempo de cristalização, dos mais de 10 μs da liga anterior usada em PCM, para menos de 100 ns [3, 42], tornando-o um material promissor.

Em uma célula de memória PCM (Figura 2.10(a)), os principais constituintes são: os eletrodos (*bottom electrode* e *top electrode*), contatos metálicos por onde entram e saem sinais elétricos; o aquecedor, que tem o papel de transformar a energia elétrica em térmica para aquecer a parte do GST que ele está em contato; e a região ativa, porção do cristal GST que sofre modificações de estado com o calor recebido do aquecedor. A passagem de uma fase à outra ocorre com o aquecimento do material calcogênico, feito através da aplicação de pulsos de corrente elétrica específicos na *bit-line*, vista na Figura 2.10(b). Com a ativação do transistor nMOS por meio da *word-line*, os pulsos provenientes da *bit-line* geram uma diferença de potencial no elemento de memória, provocando o fluxo

[13] A tecnologia é licenciada pela Ovonyx [34].

[14] Um calcogênio é qualquer elemento da família do oxigênio, exceto ele mesmo. Neste caso, o telúrio é o elemento que pertence a família do oxigênio.

[15] Esse tipo de material já é conhecido e utilizado há algum tempo, por exemplo, em CDs e DVDs regraváveis.



(a) Célula PCM. (b) Elemento de memória PCM.

Figura 2.10: Figuras adaptadas de [18].

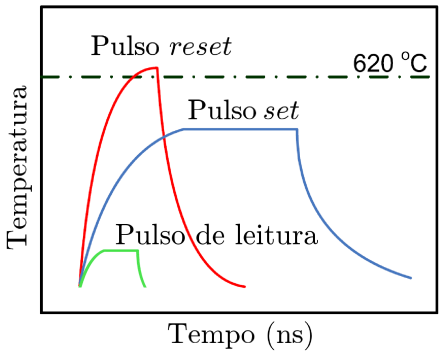
de elétrons da fonte do transistor para o dreno, conectado ao eletrodo de baixo, por onde os elétrons alcançarão o *heater*, que será aquecido, modificando a região ativa da liga. Na Figura 2.11(b) é mostrada uma fotografia, obtida pela técnica de microscopia eletrônica de varredura, de um elemento de memória real, que usa um dispositivo de acesso nMOS.

A escrita é realizada por meio de duas operações: *set* e *reset*. A operação *set* consiste em pulsos longos de potência moderada, longa duração e descontinuados gradativamente (Figura 2.11(a)), de forma que a temperatura do material é elevada até pouco abaixo do seu ponto de fusão, sendo sustentada tempo suficiente para ordenação atômica quando, então, esfria-se paulatinamente mantendo a estrutura ordenada, isto é, o estado cristalino (de baixa resistividade). Na operação *reset*, pulsos rápidos de alta potência (alta amplitude) são injetados e abruptamente descontinuados, aquecendo a liga para além do ponto de fusão quando, então, esta sofre resfriamento brusco, fixando-a no estado fundido, amórfico e de alta resitividade elétrica. Deve-se notar que essas alterações incorrem apenas na região ativa.

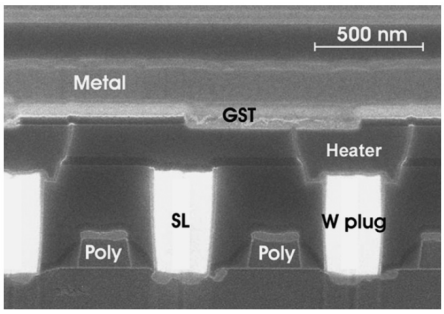
A leitura da célula é feita aferindo a queda de tensão nas *bit-lines*; quando a célula se encontra na fase cristalina, a tensão aplicada à *bit-line* reduz em função do caminho que a própria célula de memória e o dispositivo de acesso[16] oferecem. Logo, se a célula estiver no estado amorfo, a alta resistência dificulta a movimentação das cargas e a variação da tensão não é sentida no circuito de sensoriamento da memória. Note que o pulso de leitura é de baixa amplitude e rápida duração, como visualizado na Figura 2.11(a).

Observa-se, pois, pela Figura 2.11(a), que leitura é mais rápida do que a escrita e consome menos energia. No protótipo de PCM de 4 Mbits construído em [6], o tempo

[16]Pode ser um transistor FET, como na Figura 2.10(b), ou um transistor bipolar de junção [29] ou um diodo [27].



(a) Descrição da temperatura pelo tempo do sinais de escrita e leitura.



(b) Elemento de memória PCM real.

Figura 2.11: (a) A temperatura de 620 °C é o ponto de fusão do material. Adaptada e modificada de [58], com informações de [6]. (b) Fotografia obtida por um microscópio eletrônico de varredura (MEV), originalmente publicada em [6]. O conector (*plug*) de tungstênio conecta o dreno do transistor nMOS ao aquecedor (*heater*). A SL (*source line*) fornece uma tensão de referência, e está conectada na fonte do transistor. O *gate* do transistor é indicado por *poly*. A região ativa da liga GST está em contato como aquecedor através de um fosso (*trench*) no material dielétrico. Sobre a liga, está o metal que conecta a célula à *bit-line*.

de leitura obtido foi de 45 ns, com tensão e corrente aplicadas na *bit-line* de 0,4 V e 80 μA, respectivamente. Analisando, novamente, a Figura 2.11(a), pode-se deduzir que a operação *set* domina a latência de escrita e a operação *reset* o custo energético. Nesse mesmo trabalho, o conjunto corrente e tensão para a operação *set* foi de 1,5 V e 300 μA, enquanto para a operação *reset* foi de 2,7 V e 600 μA, com latências logradas de 150 ns e 50 ns, respectivamente.

Em função da ampla diferença de resistividade apresentada pelo material calcogênico, há a possibilidade de caracterizar mais do que um *bit* por célula – como nas células de memória *Flash* – com a discretização dos níveis de resistência. Quanto à diferença entre PCM MLC e SLC, em termos práticos, para a primeira há uma maior complexidade do circuito de sensoriamento e na forma como é realizada a escrita. Atualmente, até 4 *bits* por célula são comportados em PCM MLC [39].

2.3.1 Principais desafios

Escritas em PCM geram estresse termomecânico nos constituintes da célula, resultando em ciclos de dilatação e contração que acabam por levar uma célula a falhar permanentemente. O resultado da falha é a constância em um dos estados: cristalino ou amorfo.

Embora os mecanismos de falhas não estejam completamente entendidos [28], especula-se que a célula falhe no estado cristalino devido a difusão do eletrodo sobre o calcogênico [3], provavelmente causado pelo aquecimento exacerbado gerado pela aplicação da alta corrente elétrica. Por outro lado, a célula falha no estado amorfo pela delaminação[17] da região de contato entre o calcogênio e o aquecedor [3].

Não obstante, a duração média de uma célula PCM é da ordem de 10^8 escritas em situações reais[18] (*chips* fabricados) [30], o que é uma durabilidade 1000 vezes maior daquela lograda pela memória *Flash*. Ademais, a escrita é 1000 vezes mais rápida – mesmo considerando o pior cenário: a operação *set* – do que da memória *Flash* [34], com a vantagem da escrita poder ocorrer individualmente nos *bits*, sem a necessidade de se apagar o conteúdo anterior, como na memória *Flash*, e muito menos de se apagar conjuntos de células.

Claramente, a PCM é uma substituição vantajosa das memórias *Flash*. Entretanto, ao que parece, enquanto a indústria conseguir miniaturizar o processo de fabricação destas, aquela será ainda uma alternativa custosa e sem produção em massa. A despeito disso, a perspectiva sob a PCM é boa o bastante para ser cotada como substituta da DRAM, visto que as projeções de escalabilidade do processo de fabricação não são as melhores para a memória dinâmica. O ITRS (*International Roadmap for Semiconductors*) relatou que há trabalhos que utilizando nanotubos de carbono como eletrodos mostraram escalabilidade abaixo de 5 nm para o processo de fabricação da PCM [51].

Para que, de fato, a PCM seja um solução viável como substituta da DRAM, é necessário mitigar suas diversas desvantagens frente a DRAM: baixa durabilidade, alta latência de leitura e escrita e alto consumo energético. Nas próximas subseções aprofundar-se-á essas questões com as soluções propostas na literatura.

2.3.2 Estudos sobre consumo de energia e latência

Embora alguns estudos comparativos mostrem grande desvantagem da PCM em relação à DRAM nos aspectos de consumo de energia e latência das operações de leitura e escrita [29, 41], há diversas vantagens da PCM sobre a DRAM. Um sistema computacional com sua memória principal sendo não-volátil tem a possibilidade de desligar completamente regiões da memória em momentos de ociosidade, reduzindo o consumo de energia, estático ou dinâmico. Elimina-se também a necessidade de *refreshes*. Além disso, a leitura em uma célula PCM não é destrutiva como na DRAM, evitando a necessidade de uma escrita de restauração do valor lido.

[17] Separação em camadas de um material composto.

[18] O relatório do ITRS (*Internacional Technology Roadmap for Semiconductors*) de 2011 [51] relata que há trabalhos experimentais que alcançaram a durabilidade de 10^{11} escritas.

Em função dessas qualidades complementares, alguns trabalhos da literatura [18,29,41] projetaram sistemas de memória híbridos, os quais se valem de ambas as qualidades em uma espécie de mutualismo. As propostas de um *buffer* DRAM permeando a comunicação entre memória principal PCM e a *cache* do processador, apresentadas em [29, 41], acabaram se tornando praxe na literatura [4, 25, 40, 45].

Sistemas híbridos têm fornecido boas soluções; Lee et al. conseguiu, com a adição de um *buffer* de DRAM em seu sistema PCM, praticamente torná-lo equivalente a um sistema de memória com DRAM, eliminando o consumo de energia 2,2 vezes maior da PCM e acelerando o desempenho do sistema em 33% [29]. Dhiman et al. mostrou uma redução de 30% no consumo energético de sua memória, constituída de 1 GB de DRAM e 3 GB de PCM, em relação à arquitetura convencional [18]; e Qureshi et al. exibe um sistema de memória formado por 1 GB de DRAM e 32 GB de PCM com desempenho muito próximo a um sistema com 32 GB de DRAM, mas consumindo metade da energia necessária.

Outro viés na literatura é explorar as assimetrias temporais e energéticas das operações de leitura e escrita na PCM. Cho e Lee em [11] propuseram um esquema (*Flip-N-Write*) que chama a atenção pela simplicidade: para um dado endereço de memória, se uma palavra de dados – a ser escrita – modificar menos do que metade dos *bits* desse endereço, ou seja, se distância de Hamming[19] entre a palavra atual na memória e a nova for menor ou igual a metade do número de *bits* da palavra, então, ela será gravada na sua forma original, caso contrário, será escrita com todos seus *bits* invertidos. Logo, sempre metade ou menos *bits* serão escritos, reduzindo o consumo de energia.

Similarmente, mas bem mais elaborado, em [58] Xu et al. descrevem uma técnica que realiza a escrita de uma palavra de dados com os *bits*, ora originais, ora invertidos; dependendo se a escrita produzirá um maior número de operações *reset* ou *set*, uma vez que a operação *reset* consome mais energia, é preferível que ela ocorra com menor frequência. Diferentemente do trabalho de Cho e Lee (em [11]), pode-se modificar mais *bits* na escrita do que a metade do tamanho da palavra. Porém, ainda assim, é obtido uma economia de energia de até 60%. Para ambas as técnicas, a recuperação da informação na memória se dá através de um *bit* extra gravado concomitantemente à palavra de dados. Esse *bit* é um, se a palavra foi escrita de maneira invertida, e zero, caso contrário. Logo, basta que o controlador de memória leia-o para saber se realizará uma operação de inversão de *bits* antes de enviar a informação, por exemplo, ao processador.

Utilizando a assimetria temporal entre leitura e escrita, Qureshi et al. em [35] aumentam a performance de leitura da PCM, bloqueando escritas e as reagendando para dar maior vazão ao atendimento das requisições de leitura. De maneira complementar a esse trabalho, novamente, Qureshi et al. em [38] viram nas assimetrias das operações

[19]Número de *bits* nos quais duas palavras binárias se diferenciam.

set e *reset* uma oportunidade de propor uma melhoria da técnica anterior. Quando uma linha na memória *cache* é alterada, a linha correspondente na memória principal perde a validade. Nesse momento, o sistema de memória proposto aplica a operação *set* em todas células da linha inválida, na memória principal. Dessa forma, quando a escrita de volta[20] da memória *cache* de fato for solicitada, apenas a operação *reset* será executada, como é uma operação rápida, a escrita da linha é acelerada. O resultado final dos trabalhos é um aumento no desempenho do sistema de memória de 34%.

2.3.3 Estudos sobre desgaste e segurança

A limitação do número de escritas da PCM requer uma modificação na forma de funcionamento da memória principal. Para uma memória dinâmica, inexiste o problema de escritas repetitivas em um único endereço. Contudo, em uma PCM, um único endereço sendo constantemente modificado levará ao desgaste total das células por ele endereçadas. Estas falhas invalidarão a escrita naquela endereço e, por conseguinte, as próximas escritas na página.

Uma forma de evitar o problema descrito acima é nivelar o desgaste, espalhando-o por toda memória. Para tanto, faz-se necessário redirecionar escritas muito localizadas para outros endereços menos frequentemente escritos. Esse nivelamento chama-se *wear-leveling* e já é muito comum em memórias *Flash* [8]. Enquanto para memórias *Flash* é mais interessante o *wear-leveling* de blocos, para PCM, granularidades mais finas – como linhas – são mais interessantes. Todavia, quanto menor é a unidade de memória que se deseja nivelar o desgaste, maior é o número de unidades à serem reendereçadas, bem como o tamanho em *bits* desses endereços. Dessa forma, o modo tradicional de remapeamento de endereço – como feito com tabela de páginas da memória ou a com TLB – exige um *overhead* de espaço muito grande.

O trabalho de Qureshi et al. em [40] traz uma técnica inovadora e simples para fazer o nivelamento de desgaste das linhas de memória. Em vez de usar uma tabela – como a TLB – para fazer o remapeamento de endereços, determina-se pequenos conjuntos de linhas da memória e, para cada um deles, há um par de contadores destinados a calcular o redirecionamento dinamicamente. Os registradores são atualizados quando o número de escritas no conjunto atinge um valor pré-definido, isso não impede a existência de endereços frequentemente escritos, mas espalha essas escritas às várias linhas.

Não obstante, isso é insuficiente para uma PCM como memória principal, pois ela ainda acrescenta preocupações com ataques, que em uma DRAM não teriam efeito: a incidência ininterrupta de escritas até que uma linha falhe e leve a memória ao colapso. Note que a técnica supracitada é determinística, basta que se descubra os valores dos registradores,

[20] Do inglês *writeback*.

que o redirecionamento não impedirá um ataque a uma determinada linha. Dessa forma, também em [40] são propostos alguns meios de aleatorização do remapeamento. Com isso, além de aumentar o tempo de vida de uma memória sobre ataque em 4 a 5 ordens de magnitude, o algoritmo alcançou 97% de eficiência do que seria a melhor situação teórica de nivelamento de desgaste, na qual realmente não há linha de memória com maior desgaste do que outra.

Conquanto a técnica de Qureshi et al. tenha se tornado um argumento de sustentação para outros trabalhos na literatura [23, 43], de maneria que supor que todas escritas serão igualmente distribuídas pela memória não seja uma premissa refutável, a técnica é sujeita à ataques bem conhecidos como o ataque baseado no paradoxo do aniversário [47]. No intento de melhorar sua técnica, Qureshi et al. em [36] inseriu um mecanismo adaptativo para diversos tipos de ataques mantendo a performance em 97% do valor teórico máximo.

Em [45], Seong et al. não conseguem atingir o mesmo desempenho de Qureshi et al., apenas 81,2% contra os 97%, apesar disso, sua proposta abrange mais tipos de ataque, como *side-channel attack*. Há outros trabalhos propostos na literatura sobre *wear-leveling*, como [9]; entretanto, a falta de análise sobre segurança, diminui a chance da proposta ser utilizada em uma PCM.

2.3.4 Estudos sobre recuperação de falhas

É necessário perceber que mesmo nivelando o desgaste equitativamente sob uma memória, se houver uma célula que, por algum motivo (como falha no processo de fabricação), não suporte tantas escritas quanto qualquer outra na memória, toda a memória estará comprometida. De forma imprescindível, há de haver recursos e ou mecanismos que isolem a falha e garantam a integridade da informação armazenada. Ao contrário de uma falha em DRAM, que é comumente não permanente (chamada de *soft-error*), em PCM a falha é irrecuperável.

Diferentemente de memórias voláteis, em memórias não-voláteis, o tipo de erro mais comum é o *hard-error*: permanente e irrecuperável, muito embora suplantável. Dar condições de funcionamento a um dispositivo com erros permanentes está intimamente ligado à durabilidade do dispositivo. Isto é, permitir que uma PCM continue funcionando, mesmo com erros permanentes, a mantém economicamente viável, pois assiste o fabricante contra pequenas eventualidades do processo de fabricação e, ainda que não atinja o tempo de vida de uma DRAM, diminui sua desvantagem em comparação a esta.

Toda informação pode ser resguardada por um código corretor de erros[21]. O que um código desses faz é gerar metainformação a partir da informação original, de maneira que juntas, a informação original pode ser recuperada. Contudo, preservar toda integridade

[21] Ou código de correção de erros.

da informação original requer redundância e os meios de transmissão e de armazenamento são recursos escassos. Por isso, a metainformação é limitada e a capacidade de recuperar corretamente uma informação deve ser criteriosamente definida, dependendo do meio físico onde a informação é transferida ou armazenada.

Diversos trabalhos na literatura têm propostos novas formas de recuperação de falhas para PCM. Alguns utilizam os código corretores da teoria da informação para, a despeito da falha não ser transiente, recuperar a informação correta na leitura de uma linha de memória. Outros propõem novas formas de se recuperar de falhas, baseando-se em características próprias da PCM.

Em [23] Ipek et al. propõem DRM (*Dynamically replicated memory*), um mecanismo de recuperação de falhas a partir da reciclagem de páginas de memória com células falhas. Duas páginas com erros são emparelhadas logicamente de maneira que não haja sobreposição da localização de seus erros; tal que uma página sirva de *backup* à outra, quando uma escrita for endereçada na localização de um erro. No artigo é demonstrado um resultado 40 vezes melhor do que o tradicional código de correção de erros usado em DRAM, o SECDED (*Single error correction and double error detection*).

Todavia, Schechter et al. em [43] tomam considerações diferentes de Ipek et al., e chegam em resultados que contradizem a superioridade do mecanismo apresentado em [23], isto é, o SECDED apresentou um desempenho melhor do que a técnica DRM. O ponto crítico é que as considerações assumidas por Schechter et al. – no caso, uma memória com SECDED só desativa uma página após o segundo erro, enquanto para Ipek et al. isso acontece logo após a correção do primeiro erro – foram adotadas por outros trabalhos relacionados na literatura [46, 60], e o trabalho de Ipek et al. acabou sendo ignorado nas comparações de resultados desses trabalhos posteriores. Especula-se pela durabilidade aquém de um código de correção de erros bem estabelecido como o SECDED.

O trabalho de Schechter et al. apresenta excelentes resultados com uma técnica de recuperação de falhas bem simples e exclusiva para memórias não-voláteis, a ECP (*Error-Correcting Pointers*). Alcançando o número de bilhões de escritas, o trabalho argumenta que códigos corretores de erros, para memória com restrições na durabilidade, incorrem em desgaste desnecessário das células de memória. Na ECP, a metainformação de um código de correção de erros tradicional é substituída por células sobressalentes e apontadores (ou ponteiros); quando uma falha acontece em uma linha, uma célula sobressalente passa a ser utilizada, sendo toda escrita destinada à célula falhada redirecionada – através dos ponteiros – à sobressalente.

Seong et al. [46] também apresentam uma técnica pensada para memórias não-voláteis: a SAFER (*Stuck-At-Fault Error Recovery*). Muito mais elaborada e complexa, requer um menor tamanho de metainformações e há possibilidade de se recuperar de mais erros do que a ECP, o que na prática trouxe resultados de durabilidade melhores daqueles

apresentados em [43]. Sua principal característica é a gravação de grupos de *bits* da palavra de dados invertidos ou não; a inversão depende se o *bit* que deveria ser gravado sobre uma célula falha terá um casamento de valor ou não. A despeito de sua maior durabilidade, e talvez pela sua complexidade, o mecanismo não foi utilizado em trabalhos posteriores na literatura, como [25, 37].

A técnica FREE-p (*Fine-grained Remapping with ECC and Embedded-Pointers*) apresentada por Yoon et al. em [60], de certa forma alinhada com a técnica de Ipek et al., também recicla páginas com falhas. Na verdade, recicla-se as linhas falhas de páginas de memória, tornando-as ponteiros para linhas incólumes de outras páginas. Diferentemente da DRM, que usa paridade apenas para indicar falhas, Yoon et al. usam um código de correção de erros robusto e, somente quando esse falhar, as linhas serão reaproveitadas como ponteiros. O resultado é uma técnica com desempenho melhor do que aquela proposta por Schechter et al.

Qureshi propôs em [37], PAYG (*Pay-As-You-Go*), uma maneira de organizar técnicas de correção de erros, como ECP e SAFER, de modo que haja redução do *overhead* de *bits* de metainformação. A partir da análise do modelo analítico da ECP, ele percebeu que mais do que 95% das linhas, no máximo, necessitavam apenas da correção de um erro. Dessa forma, construindo a organização do sistema de memória de forma que as linhas tivessem capacidade de corrigir apenas um erro, com os demais erros corrigidos sob demanda, obteve um tempo de vida maior do que a técnica ECP.

2.3.5 Estudos com *multi-level cell*

Outra natureza de dificuldades são as escritas em células PCM MLC. Embora a leitura não se diferencie entre células PCM *multi-level* e *single-level*, a escrita em células MLC é executada em estágios. No projeto de uma PCM SLC, tempo e corrente necessários para escrever são designadas a partir da medição da distribuição de resistividade das células de memória; por exemplo, quando um pulso *reset* é aplicado, este deve deixar as células alvo com uma resistividade alta, obedecendo um limite mínimo de resistência, se algumas células ficarem abaixo do limite, há necessidade de aumentar, ou a amplitude, ou o tempo, de exposição das células ao pulso. Analogamente, se o pulso é suficiente, é possível tentar reduzir um dos parâmetros (como feito em [28]) até que se chegue à mínima amplitude ou ao mínimo tempo.

Desse modo, a escrita em uma linha de memória MLC é complexa, os pulsos em cada *bit-line* deveriam ser diferenciados, o que é inviável de se fazer. Assim, um pulso *reset* inicial é seguido de vários pulsos *set*, conformando em uma escrita incremental, o que requer a utilização do mecanismo de escrever-e-verificar[22] no qual, após cada escrita, mensura-se

[22]Do inglês *write-and-verify* [35] ou *program-and-verify* [25].

se os valores de resistência esperados foram alcançados. Caso isso não aconteça, uma nova tentativa é realizada, até que a linha possua o valor desejado. Claramente, nessa situação incorrem alta latência de escrita e alto consumo de energia, além do fato de se fazer múltiplas escritas, para uma escrita efetiva de dados, reduzindo mais rapidamente a durabilidade das células.

Como existem, em uma mesma linha, células que demandam mais escritas do que outras, Jiang et al. em [25] propõem uma mecanismo limitante no número escritas necessário. O número de escritas se torna aquele no qual as poucas células que ainda não atingiram o valor alvo podem ser ignoradas, com a utilização de um código corretor de erros; ou seja, as células que demandariam mais passos de escrita são vistas como *bits* de informação errados e a informação original pode ser recuperada a partir do código de correção armazenado na linha. O resultado foi uma redução na latência de escrita de 57%.

Embora se diga que não há necessidade de se preocupar com *soft-errors* em PCM [43], existe uma forma desse gênero de erro que acontece em PCM, quando se trata de células *multi-bit*. No material calcogênico, estados intermediários nos quais o material se apresenta parcialmente ordenado (ou desordenado) atomicamente, ao longo do tempo sofrem o efeito de *drift*, que é a desorganização do material no estado cristalino (e ordenado).

Na verdade, quanto mais houver desordenação no estado do material, mais desordenado ele tende a ficar. Acontece que dividir de modo exato a diferença de resistência apresentada pelos dois estados extremos – estado muito cristalino e estado muito amorfo –, pelo número de *bits* que se deseja representar na célula, poderá trazer problemas de reconhecimento da informação armazenada; uma vez que informações representadas por estados muito desorganizados (amorfos) sofrerão *drifts* mais significativo, podendo ultrapassar o nível de resistência (ou seja, o estado) que caracteriza aquela informação. Como consequência, ao passar do tempo, uma célula poderia representar uma informação que difere da originalmente escrita.

Em [4] Awasthi et al. propõem usar o mecanismo de *scrub* – análogo ao presente na DRAM – com o objetivo de evitar *soft-errors* na memória. Periodicamente, as linhas de memória são lidas, de modo que, quando um código de correção de erros indica erros em alguma linha, uma reescrita é feita para corrigir a falha. Nesse caso, a utilização de códigos corretores é a mais indicada, apesar disso, no próximo capítulo se discute a redução do tempo de vida da PCM que esses códigos podem provocar, quando utilizados sobre *hard-errors*.

2.4 Síntese

Neste capítulo foram apresentadas algumas tecnologias de memórias emergentes que tem se destacado pela intensiva pesquisa tanto na indústria quanto na academia. A Tabela

2.1 enquadra aquelas que se apresentam como candidatas mais fortes a serem a próxima tecnologia de memória, que substituirá ou a memória *Flash*, ou a DRAM, ou ambas.

Na Seção 2.3, apresentou-se sobre alguns desafios para PCM e vários trabalhos elaborados para suplantar tais desafios. Porém, neste trabalho, o principal foco são os estudos de recuperação de falhas em PCM, nos quais apresenta-se técnicas de correção de erros. Na Tabela 2.2 sumariza-se quais dessas técnicas serão estudadas de modo aprofundado no próximo capítulo. Na tabela, algumas dessas técnicas, como a ECP e a SAFER, não utilizam códigos de correção de erros, pois possuem mecanismos desenvolvidos especificamente para a característica de não-volatilidade e duração limitada da PCM.

Tabela 2.1: Sumário de tipos de memórias emergentes. Os dados foram compilados de [6, 7, 39, 58, 59], buscando-se coerência e imparcialidade.

	ReRAM	**FeRAM**	**STT-MRAM**	**PCM**
Estágio	Em pesquisa	Consolidada	Protótipo	Protótipo
Elemento	Memristor	C. ferroelétrico	MTJ	Liga GST
Durabilidade	10^4 - 10^{12}	10^{12}	10^{15}	10^8
Área(F^2)	< 5	4-15	< 20	4-16
Leitura (ns)	< 10	< 50	< 20	< 50
Escrita (ns)	10	< 50	< 20	< 500
Energia (pJ)	3	-	< 2	25

Tabela 2.2: Sumário das técnicas de correção de erros.

	ECP	DRM	SECDED	SAFER	FREE−p
Código de correção	Sem código (usa mecanismo próprio)	Paridade	Hamming + Paridade	Sem código (usa mecanismo próprio)	BCH

Capítulo 3

Modelos e métodos

Neste capítulo apresenta-se a modelagem matemática e a análise das técnicas de correção de erros estudas. Ademais, são apresentadas construções de modelos analíticos para duas dessas técnicas. Primeiramente, é feita explicação do que é a probabilidade de *bit-flip*, como é modelada e obtida experimentalmente. Este é o fator que permite modelar a alteração dos *bits* de uma linha de memória, levando em conta a diferenciação entre dados e metainformação. Em seguida, é elucidado as hipóteses e o ferramental matemático utilizados na construção do simulador para predição do tempo de vida de uma memória, sob aplicação de uma técnica.

Finalmente, são apresentadas as técnicas de correção de erros e a modelagem da probabilidade de *bit-flip* de cada uma, além de como esta é inserida no modelo de simulação. Após a descrição do modelo de cada técnica, simulações da modelagem proposta são defrontadas com os resultados tradicionalmente apresentados na literatura, nos quais a probabilidade de *bit-flip* tem influência irrisória e não há distinção da frequência de alterações dos dados e da metainformação na escrita.

Ao final do capítulo, duas formas de modelagem analítica probabilística são exibidas, construindo-se três modelos analíticos, dois para técnica ECP e um para a técnica SEC-DED. Para as outras técnicas é discutido os motivos pelos quais não se alcançou modelos viáveis.

3.1 Modelagem da probabilidade de *bit-flip*

A probabilidade de *bit-flip*[1] refere-se à probabilidade de troca de valor dos *bits* de uma palavra (ou linha) em um endereço de memória com a escrita de uma nova palavra (ou linha). Considerando que o evento de troca de valor de um único *bit* é independente,

[1] A tradução aproximada para o português seria probabilidade de "inversão de *bits*". Por conta de não ser uma tradução direta, neste trabalho será mantido *bit-flip*.

pode-se tomar a escrita de uma nova palavra de n *bits* na memória como a ocorrência de uma série de n eventos independentes com probabilidade p de que cada um ocorra. Assim, considerando a escrita de um *bit* como sucesso, se o valor do *bit* foi invertido depois da escrita, ou fracasso, se o valor não foi invertido; é possível modelar o fenômeno da escrita de uma palavra na memória por uma distribuição binomial. Logo, o número esperado de trocas no valor dos *bits* de um endereço de memória quando uma nova palavra ou uma sequência de novas palavras forem escritas é $n \cdot p$.

A hipótese da independência entre os valores dos *bits* é válida estatisticamente quando se assume a aleatoriedade das palavras de dados e a generalidade do universo de programas, isto é, casos de programas cujos dados produzidos tenham correlação entre *bits* não se encaixam no modelo e não foram levados em conta nos modelos de sistemas de memória encontrado nos mais importantes trabalhos da literatura [11,23,43,58]. Porém, a hipótese não é universal. Outra importante consideração feita na literatura é a de que um *bit* tem probabilidade de 50% de trocar de valor em uma escrita [37,43,46,60]. Estatisticamente, pode-se assemelhar a escrita de um *bit* ao lançamento de uma moeda, assim, o resultado de um evento é um valor dentre dois possíveis. Portanto, é concluível – com base no modelo de distribuição binomial – que o valor médio de *bits* invertidos em uma escrita de palavra de n *bits* seja metade do tamanho da palavra.

Conquanto esses dois argumentos sejam pilares para os trabalhos citados, é parte do trabalho científico questioná-los. E se, por um lado, a análise da correlação entre *bits* depende de se analisar o funcionamento de um dado programa, por outro, a análise da taxa média de *bit-flips* depende apenas da obtenção das palavras de dados a serem escritas na memória. Claramente, isto não é uma tarefa simples, mas é independente de qual é o programa e, desse modo, uma vez a metodologia definida, ela é reaplicável para qualquer programa.

Para se analisar a veracidade do argumento de que um *bit* tem probabilidade de *flip* de 50%, toma-se o modelo da distribuição binomial ainda como válido para a escrita de uma palavra na memória, ou seja, a escrita é dada por uma sequência de eventos cuja probabilidade p de cada evento agora é desconhecida. Então, mensura-se a quantidade de *bit-flips* ocorridos a cada palavra de dados escrita na memória, durante a execução de um ou vários programas. Com palavras de tamanho fixo n, a taxa média de *bit-flips* $\tau \mid 0 \leq \tau \leq n$ é o valor médio esperado da distribuição binomial, *i.e.*, $\tau = n \cdot \rho$. Assim, deduz-se que $\rho = \tau/n$.

Embora continue a ser citado como probabilidade de *bit-flip*, ρ é um valor estatístico[2], resultante da média de uma amostra da população de programas. Após obitdo o valor de ρ, este é utilizado como parâmetro nos modelos de simulação, na tentativa de se buscar

[2]Valor estatístico da probabilidade de *bit-flip* é uma apropriação semântica, toda vez que for utilizada será denotado a letra grega ρ, caso contrário, se não for o valor estatístico, será usado p.

uma verossimilhança entre as simulações e um sistema de memória.

3.1.1 Probabilidade de *bit-flip* experimental

Nesta seção elucida-se como é feito a obtenção da probabilidade de *bit-flip* experimental que será estuda e discutida no Capítulo 4.

Forma de obtenção

Uma forma de obter as palavras de dados durante a execução de um programa é instrumentar sua execução. A instrumentação pode ser em tempo de execução – instrumentação dinâmica – ou codificada junto do código fonte, sendo o código de instrumentação inserido durante a programação ou no momento da compilação – instrumentação estática. *PIN Tools* [14] é um ferramental de instrumentação dinâmica, disponibilizada pela Intel e que permite construir ferramentas em C/C++ para instrumentar a execução de um programa.

Neste trabalho, para analisar e contar o número de *bit-flips* que ocorreram a cada palavra de dados escrita na execução de um programa, utilizou-se uma *PIN Tool* construída para simular uma memória *cache* de processador. As instruções de acesso à memória, emitidas no decorrer da execução de um programa instrumentado, tinham o tipo (escrita ou leitura) e o endereço analisados para, então, o simulador realizar o processamento similar àquele que seria feito pela memória *cache* de um processador.

A contagem de *bit-flips* é feita quando uma palavra de dados é removida da memória *cache* de instrumentação para ser gravada na memória principal. Todo endereço que possui dados nessa *cache* tem uma cópia da palavra da última atualização na memória principal, pois, lendo-se essa cópia e a comparando *bit* à *bit* com a nova palavra de dados a ser escrita no endereço, determina-se o número de *bits* i que têm seus valores trocados após a escrita. Por conseguinte, na posição i, de um vetor (o qual mantém a frequência de ocorrência de cada quantidade de *bit-flips*) de índices de 0 à n, incrementa-se o valor atual em uma unidade (veja o algoritmo da Figura 3.1). Ao final da execução de um programa, é obtida a totalidade de *bits* que tiveram seus valores invertidos, por meio de arquivos de *profile*.

Seja Q_i a quantidade de palavras que tiveram i *bit-flips*, onde $0 \leq i \leq n$, logo $\sum_{i=0}^{n} Q_i$ representa o total de escritas na memória principal e $\sum_{i=0}^{n} i \cdot Q_i$ é total de *bit-flips*. Daí:

$$\tau = \frac{\sum_{i=0}^{n} i \cdot Q_i}{\sum_{i=0}^{n} Q_i} \tag{3.1}$$

```
Resultado: Computação da frequência de bit-flip de um programa.
enquanto o programa em execução não terminar faça
    se cache solicitou uma escrita-de-volta então
        palavraDados ← ExtraiPalavraLRU();
        aux ← palavraDados ⊕ ObtémPalavraCópia(palavraDados.Endereço);
        i ← ContaUns(aux);
        Q_i ← Q_i + 1;
        GravaPalavraMemória(palavraDados);
    fim
fim
```

Figura 3.1: Algoritmo de obtenção da taxa de *bit-flips*.

Lembrando que τ é a taxa média (obtida através da média ponderada) de *bit-flips* de um ou vários programas – podendo representar uma amostra da população de programas –, a partir dele é calculado ρ, o valor estatístico da probabilidade de *bit-flip* (usando a $\rho = \tau/n$).

Experimento

Para a obtenção de um valor de ρ significativo, o conjunto de programas analisados tem de ser representativo. Por esse motivo, foi selecionado o *benchmark suite* SPEC CPU2006 [15]. A despeito do número pequeno de programas, o SPEC CPU2006 possui um repertório diferenciado, abrangendo vários tipos de aplicações, de forma que é plausível supor padrões de acesso à memória – principalmente, de escrita de dados – diversificados, o que é de interesse para os objetivos deste trabalho. Ademais, sua adoção é bem ampla na literatura estudada [11, 29, 40, 46].

O SPEC CPU2006 permite diversas configurações, dentre elas a utilização de *PIN Tools* durante sua execução e a paralelização da execução de cada *benchmark*. Contudo, como a simulação da memória *cache* insere um alto *overhead* no tempo de execução de um programa, o tempo demandado para toda execução do *suite* é muito longo, com os recursos de computação disponíveis (a *suite* completa chegava a executar por semanas). Sendo que cada *benchmark* do SPEC possui uma ou mais entradas e a execução delas ocorre sequencialmente, foi necessário criar *scripts* que permitissem a execução em paralelo das entradas de cada programa. Desse modo, utilizando os mesmos recursos computacionais, reduziu-se o tempo execução do SPEC inteiro para alguns dias.

Com a execução dos *benchmarks* foi obtido um conjunto de arquivos de *profile* dos quais contou-se o total de escritas na memória e o total de *bit-flips*, de modo que foi possível determinar τ, taxa média de *bit-flips*, e, portanto, calcular ρ para o SPEC CPU2006. Todo

os detalhes sobre os arquivos de *profile*, *software* e infraestrutura utilizados para executar o experimento estão em [22], bem como os códigos que puderam ser disponibilizados.

3.1.2 Aplicação da probabilidade *bit-flip*

A probabilidade de *bit-flip* é um dos parâmetros inseridos nos modelos de simulação desenvolvidos para simular o tempo de vida de uma memória. A construção de um modelo de simulação depende de algumas hipóteses e parâmetros gerais, somando-se a eles, posteriormente, o comportamento idiossincrático de cada técnica de correção de erros modelada. A seguir são descritas as principais hipóteses e seus argumentos, que sustentam o modelo de simulação utilizado neste trabalho.

3.1.2.1 Modelo de tempo de vida

Técnicas de correção de erros são comumente usadas para manter a integridade dos dados. Nas memórias não-voláteis elas fazem algo a mais: prolongam o tempo de vida. Os principais estudos realizados [23, 37, 43, 46, 60] para PCM mostram esse fato. Para avaliar cada técnica e compará-las entre si, é necessário estabelecer um modelo base. A partir dos trabalhos citados, os seguintes critérios são utilizados como característica de uma PCM:

- as células não possuem tempo de vida iguais;
- uma escrita modifica apenas os *bits* que terão seus valores alterados;
- cada célula armazena apenas um *bit* (memória com SLC); e
- existe um algoritmo de *wear-leveling* perfeito.

Uma vez que o processo de fabricação é imperfeito, existem variações no tempo de vida das células [1, 4, 25, 42]. Para corresponder à realidade, supõe-se, no modelo, que o tempo de vida das células da memória formam uma população com distribuição gaussiana de média μ e desvio padrão σ. Devido a essa abordagem, a existência da capacidade de recuperação de falhas na memória é fundamental. Falando-se de uma quantidade de bilhões de células – isto é, memórias com capacidade maior do que 1 G*bit* – existe uma probabilidade não desprezível de haver células cujo tempo de vida é zero, ou seja, células completamente inutilizáveis logo após a fabricação.

Escritas em memórias voláteis costumam afetar todos os *bits*, mesmo aqueles que não terão seus valores alterados. Contudo, em memórias não-voláteis como a PCM, alterar um *bit* reduzirá seu tempo de vida desnecessariamente. Supõe-se, então, que a memória realiza uma leitura antes de uma escrita, para identificar quais *bits* devem ser alterados e, desse modo, modificando apenas estes; tal esquema é conhecido como escrita diferencial (*differencial write* [25, 45]) ou leitura-antes-da-escrita (*read-before-write* [10, 20, 60]). Apesar de se poder argumentar contra sua utilização por aumentar o tempo total de escrita

acrescendo o tempo de uma leitura, em PCM uma leitura é mais rápida do que uma escrita, como exposto no Capítulo 2. Ou seja, nesse caso, o tempo de escrita absorveria o tempo de leitura. Ademais, existe uma redução no consumo de energia [58], bem como uma melhora no tempo de vida da memória [61], utilizando esse esquema de escrita.

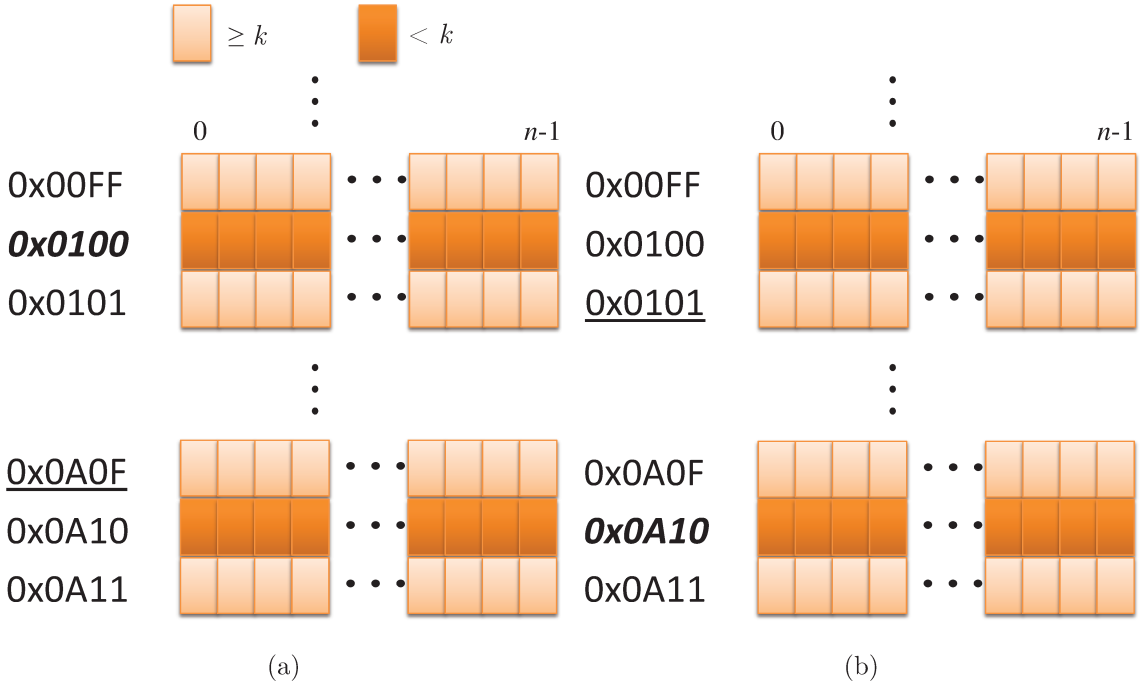


Figura 3.2: Visualização conceitual de *wear-leveling*. As células mais claras receberam pelo menos k escritas e as células mais escuras possuem menos do que k escritas. (a) Uma escrita a ser realizada é endereçada à linha de memória 0x0A0F (sublinhado); por ela ter mais do k escritas, a escrita é redirecionada à linha 0x0100 (negrito e itálico) que possui menos do que k escritas, embora o endereço 0x0A10 seja o endereço subsequente ao 0x0A0F, o mapeamento foi realizado de modo aleatório. (b) Após a escrita anterior, uma nova escrita direcionada à linha de memória 0x0101 foi mapeada para a linha 0x0A10. Porém, poderia ter sido mapeada para o endereço 0x0100, ou qualquer outro que tivesse menos do k escritas.

Muito embora uma das grandes vantagens de uma PCM seja a possibilidade de classificar diferentes níveis de resistividade na célula como estados, possibilitando usá-la como *multi-bit*, as técnicas de correção de erros estudadas não são genéricas a ponto de poderem ser aplicadas em quaisquer tipos de células. Considere, por exemplo, duas técnicas citadas no Capítulo 2, SAFER e ECP; enquanto a aplicação da técnica ECP tanto em memória com SLC quanto com MLC é direta, porque não importa o conteúdo da célula, o mesmo não é possível dizer sobre a aplicação da SAFER, uma vez que, neste caso, recu-

perar informação exata de uma célula falha não é possível: além de complicar seu método de funcionamento (que será explicado posteriormente), se a célula, ao falhar, estiver em um estado intermediário, por conta de *drift* [4], o estado da célula poderá mudar com o decorrer do tempo, tornando impossível confiar na leitura das células com falhas, o que é essencial para a técnica SAFER. Logo, no modelo adotado neste trabalho se considera apenas um *bit* por célula.

A última hipótese do modelo proposto, também encontrada nos trabalhos estudados, considera que a memória simulada possui um algoritmo de *wear-leveling* perfeito, ou seja, em um sistema hipotético usando PCM, o controlador da memória, ou a MMU, ou algum *hardware* projetado é capaz de distribuir uniformemente as escritas entre as células de memória. Para uma memória, ter um nível de desgaste perfeito significa que nunca haverá duas escritas em determinado endereço, em um intervalo de tempo menor do que aquele necessário para escrever uma única vez em todos outros endereços da memória.

O *wear-leveling* perfeito é diferente do que é exibido na Figura 3.2(b), onde o endereço 0x0100 acabou de receber uma escrita, mas continua com menos do que k escritas. Ou seja, no *wear-leveling* perfeito as linhas da memória ou têm k escritas ou têm $k-1$ escritas; quando a primeira $(k+1)$-ésima escrita ocorrer, todas as outras linhas hão de terem sofrido k escritas. Em termos práticos, no modelo, essa consideração permite simular o tempo de vida de uma memória de maneira fidedigna, sem para tanto utilizar o tempo real de vida; isso se dá, pois sabendo o que ocorre entre duas escritas distintas, não é necessário simular os acontecimentos entre elas, apenas aplicar os efeitos do que deveria acontecer.

3.1.2.2 Simulação do modelo de tempo de vida

O modelo proposto é simulado através de um simulador de tempo de vida cujos principais parâmetros de entrada são tamanho de linha de memória[3] N, tamanho de página (número de linhas por páginas) N_L, tamanho de memória (número de páginas) N_P, tempo de vida médio μ, desvio padrão σ e o tipo de técnica a ser simulada. Antes da simulação ocorrer é gerada uma memória de maneira aleatória, isto é, gera-se uma matriz de inteiros a partir da distribuição gaussiana com média μ e desvio padrão σ, de forma que cada célula da matriz representa um tempo de vida de uma célula da memória. Da matriz, cria-se um vetor com a sequência temporal de durabilidade de cada célula da memória simulada, que é um vetor ordenado de maneira crescente pelo valores dos tempos de vida. Depois desse *set-up*, a simulação é iniciada.

A memória, pois, tem capacidade (em *bits*) $C_M = N \cdot N_L \cdot N_P$. A simulação procede percorrendo o vetor ordenado e termina quando atingido um critério de parada e/ou o

[3] O simulador não trabalha com palavras de memória, mas linhas de memória. É pouco comum em um processador moderno uma palavra isolada ser gravada na memória. O que de fato acontece é toda uma linha da memória *cache*, na qual a palavra se encontra, ser gravada integralmente na memória principal.

limite tolerável de erros por cada técnica. Observe que o tempo de simulação é proporcional ao tamanho do vetor. Também note que duas células vizinhas k e $k+1$, por exemplo, embora próximas no vetor, podem possuir localizações na memória tão distante quanto possível, bem como se k possui um tempo de vida x, não necessariamente, $k+1$ possui $x+1$, e sim, $x+c_k$, onde $c_k \mid c_k \in \mathbb{N}$, é a diferença do tempo de vida das duas células. Dessa forma, como cada elemento do vetor é tratado em um passo da simulação, quando acontece a passagem do passo k para o $k+1$, significa que foram feitas $w = c_k \cdot N_L \cdot N_P$ escritas na memória. Levando em conta a existência de *wear-leveling* perfeito, todas as células da memória que possuem tempo de vida maior ou igual à $x + c_k$, terão sofrido $x + c_k$ escritas até aquele passo. Conclusivamente, as restantes, com tempo de vida menor do que $x + c_k$, terão falhado entre os passos 0 e $k+1$. Veja uma instância da situação ilustrada na Figura 3.3.

Enquanto a técnica é capaz de suportar falhas que ocorrem nas linhas e/ou páginas durante a simulação, ela progride; contudo, a despeito da memória ir ou não à pique com a incidência de falhas, o número de escritas entre dois passos pode diminuir à medida que linhas ou páginas falham irrecuperavelmente. Dessa forma, o número de escritas que acontecem entre os passos k e $k+1$ da simulação não é w como supracitado e, sim, $w - c_k \cdot N_L \cdot F_P(k)$, onde $F_P(k)$ é o número de páginas que falharam até o passo k. Note que assume-se que a primeira linha a falhar na página torna esta inválida.

A cada passo da simulação incrementa-se o total de escritas e registra-se esse valor em um arquivo de *profile*. O valor total de escritas na memória no instante k da simulação é dado pela Equação 3.2. Para saber o tempo de vida total da memória simulada, isto é, o número total de escritas, basta calcular $W_T(\omega)$, sendo ω o passo seguinte ao qual a memória definhou. Como N_L é uma constante na equação, pode ter seu valor computado posteriormente usando o arquivo de *profile*; fatorando-a da equação se reduz computação desnecessária, acelerando a simulação.

$$W_T(k) = N_L \cdot \sum_{i=1}^{k} c_i \cdot \Big(N_P - F_P(i)\Big) \tag{3.2}$$

Apesar desse cálculo estar correto, ele não é o utilizado, pois difere do que fora apresentado na literatura. Como feito em [43], usa-se o fator de desgaste (*wear-rate*). Basicamente, ele representa o declínio porcentual do número de escritas conforme o desgaste aumenta. A diferença entre a Equação 3.4, usada na simulação, e a Equação 3.2 está na substituição de $N_P - F_P(k)$ pela equação 3.3. O valor $W_T(k)$, então, passa a ser logrado do arquivo de *profile* ao multiplicar-se o valor registrado a cada passo da simulação por $N_L \cdot N_P$, como será mostrado mais adiante.

Define-se *wear-rate* no passo k da simulação por

$$f_{wr}(k) = f_{wr}(k-1) \cdot \frac{N_P - F_P(k)}{N_P - F_P(k-1)} \tag{3.3}$$

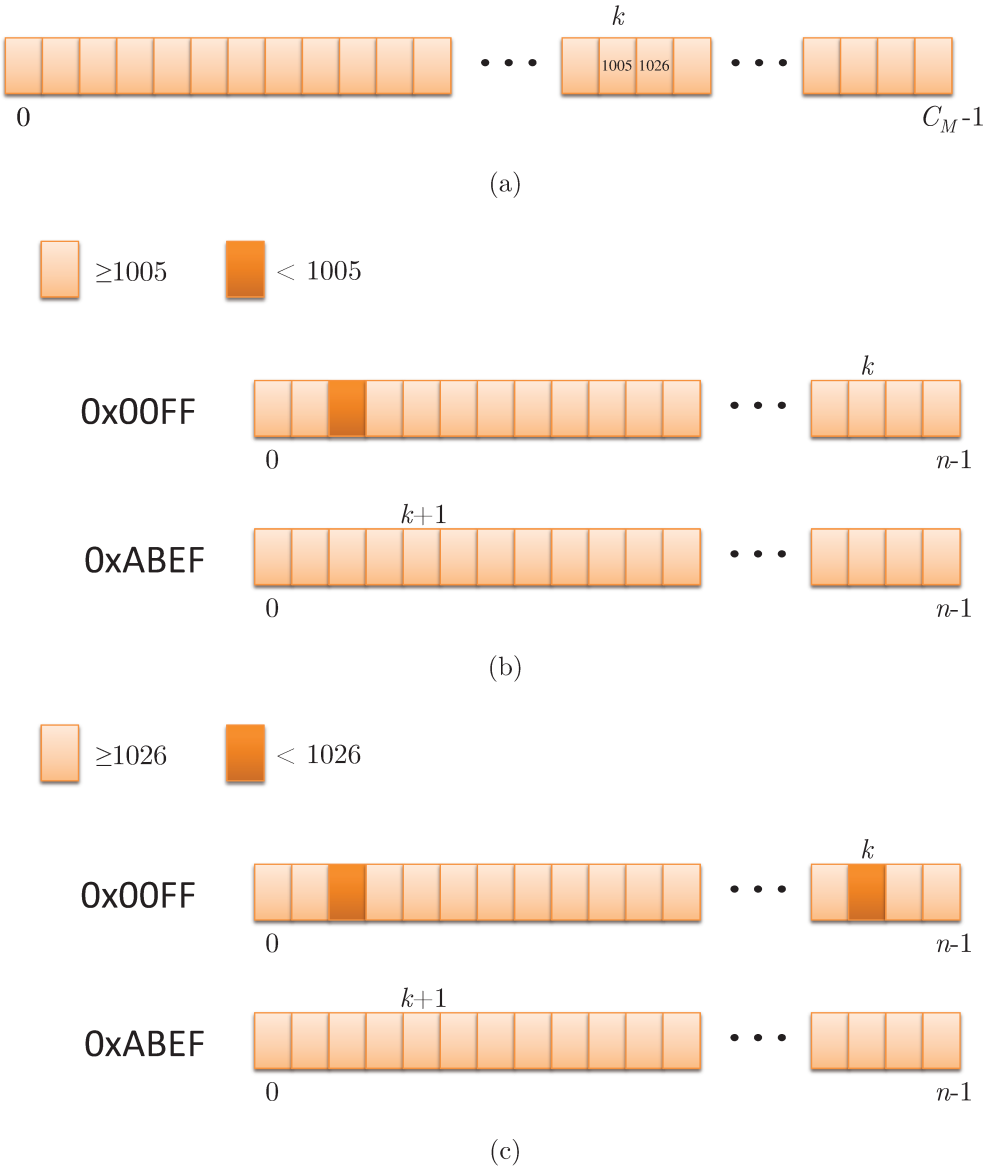


Figura 3.3: (a) O vetor ordenado contendo os tempos de vida das células da memória simulada; na posição k do vetor a célula suporta até 1005 escritas, e a célula $k+1$, ao lado, 1026. (b) A célula k está na linha localizada no endereço 0x00FF e a célula $k+1$ está na linha localizada no endereço 0xABEF; as células mais escuras nessa situação indicam aquelas definhadas por não suportarem mais do 1004 escritas. (c) Na simulação o passo k foi executado e agora será a vez do passo $k+1$. A célula k agora está inutilizada e todas as células claras já sofreram 1025 escritas e estão prestes a sofrer a 1026ª escrita‡.

‡Nota-se que a técnica simulada nesse caso suporta algumas falhas nas linhas de memória.

Logo,

$$W(k) = \sum_{i=1}^{k} c_i \cdot f_{wr}(i) \tag{3.4}$$

E, finalmente,

$$W_T(k) = W(k) \cdot N_L \cdot N_P \tag{3.5}$$

Sendo $f_{wr}(0) = 1$ e $F_P(0) = 0$, uma vez não existem falhas contabilizadas antes da simulação iniciar; as Equações 3.2 e 3.5 são equivalentes. Pois:

$$\begin{aligned}
W_T(k) &= W(k) \cdot N_L \cdot N_P \\
&= \left[\sum_{i=1}^{k} c_i \cdot f_{wr}(i)\right] \cdot N_L \cdot N_P \\
&= \left[\sum_{i=1}^{k} c_i \cdot \cancel{f_{wr}(0)}^{1} \cdot \prod_{j=1}^{i} \frac{N_P - F_P(j)}{N_P - F_P(j-1)}\right] \cdot N_L \cdot N_P \\
&= \left[c_1 \cdot \left(\frac{N_P - F_P(1)}{N_P - F_P(0)}\right) + c_2 \cdot \left(\frac{\cancel{N_P - F_P(1)}}{N_P - F_P(0)} \cdot \frac{N_P - F_P(2)}{\cancel{N_P - F_P(1)}}\right) + \cdots \right. \\
&+ \left. c_k \cdot \left(\frac{\cancel{N_P - F_P(1)}}{N_P - F_P(0)} \cdots \frac{N_P - F_P(k)}{\cancel{N_P - F_P(k-1)}}\right)\right] \cdot N_L \cdot N_P \\
&= \left[c_1 \cdot \left(\frac{N_P - F_P(1)}{N_P}\right) + c_2 \cdot \left(\frac{N_P - F_P(2)}{N_P}\right) + \cdots + c_k \cdot \left(\frac{N_P - F_P(k)}{N_P}\right)\right] \\
&\cdot N_L \cdot N_P \\
&= N_L \cdot \left[c_1 \cdot \Big(N_P - F_P(1)\Big) + c_2 \cdot \Big(N_P - F_P(2)\Big) + \cdots + c_k \cdot \Big(N_P - F_P(k)\Big)\right] \\
&= N_L \cdot \sum_{i=1}^{k} c_i \cdot \Big(N_P - F_P(i)\Big)
\end{aligned}$$

Por fim, para apurar o número de escritas que ocorrem entre dois passos da simulação, tem-se de adicionar a probabilidade de *bit-flip*. Como explicado na Seção 3.1, ela representa a probabilidade de uma palavra na memória ter seus *bits* invertidos após uma escrita; por exemplo, suponha $p = 50\%$ e uma palavra de memória de 64 *bits*, isso significa que quando uma palavra ou sequência de palavras de dados forem escritas em um mesmo endereço, espera-se que a média de *bits* invertidos seja 32. Consequentemente, o desgaste proporcionado por essa sequência é metade do que seria se todos os *bits* fossem invertidos a cada escrita, resultando em tempo de vida duas vezes maior para tal endereço.

Argumentativamente, pode-se dizer que, no caso onde sempre certa palavra é escrita no mesmo endereço, 32 *bits* nunca terão seu tempo de vida reduzidos (pois essa palavra

afeta sempre os mesmo *bits*), tornando uma falácia o fato de que suportar-se-ia duas vezes mais escritas. Todavia, além de se considerar *wear-leveling* perfeito para os endereços de memória, também se leva em conta *wear-leveling* intra-linha [43], isto é, no modelo de memória simulado, supõe-se que todo endereço tem capacidade de rotacionar a sobreposição dos *bits* de uma palavra de dados a ser escrita com os *bits* da palavra no endereço[4], de forma que leve n escritas (para uma palavra de tamanho n) para acontecer novamente uma mesma sobreposição de *bits*. A Figura 3.4 exibe um esquemático dessa situação.

Não obstante, é possível ainda contra-argumentar dizendo que se as palavras de dados a serem escritas sofrerem o rotacionamento inverso, levar-se-ia novamente ao problema de desgaste localizado nalguns *bits*. No entanto, é preciso fazer aqui considerações: a primeira é a de que a rotação de uma linha pode ser aleatória e esporádica, mas isso, no modelo de simulação, é irrelevante; a segunda é a de que se assume que as palavras de dados de um programa são aleatórias. Além do mais, a probabilidade de duas palavras se diferenciarem por uma simples rotação é de $0,5^n$, pois a probabilidade de um *bit* da segunda palavra ser igual a algum da primeira é 0,5 e isso tem de se repetir n vezes. Logo, para três palavras, a probabilidade é de $0,5^{2n}$ e assim em diante. De fato, tal sequência de dados é improvável, pois ou são geradas por uma aplicação muito específica, ou se caracterizam como ataque.

Para concluir o raciocínio por trás do uso de *bit-flip* na simulação: quanto menor for a taxa de *flips* em escritas, maior será a duração da memória. Para uma probabilidade de *bit-flip* p, o número de escritas incidentes na memória para o passo k da simulação é

$$\mathbb{W}(k) = \frac{W(k)}{p} \tag{3.6}$$

Assim,

$$\mathbb{W}_T(k) = \mathbb{W}(k) \cdot N_L \cdot N_P \tag{3.7}$$

De forma que $\mathbb{W}$ e $\mathbb{W}_T$ são valores de escritas mais verossímeis, substituindo as Equações 3.4 e 3.5 pelas Equações 3.6 e 3.7, respectivamente, na computação da simulação.

[4]Uma das formas de se fazer isso é por meio de registradores de deslocamento acionados por contadores, localizados em cada da linha ou em um grupo de linhas da memória. Embora prático, há certos desafios neste tipo de projeto: aumento da lógica combinacional e sequencial dos *chips*; acréscimo de *overhead* de metainformação, por conta do armazenamento do estado de rotação dos registradores (isto é, não se pode perder a informação de quantos *bits* são necessários deslocar para recuperar o dado original).

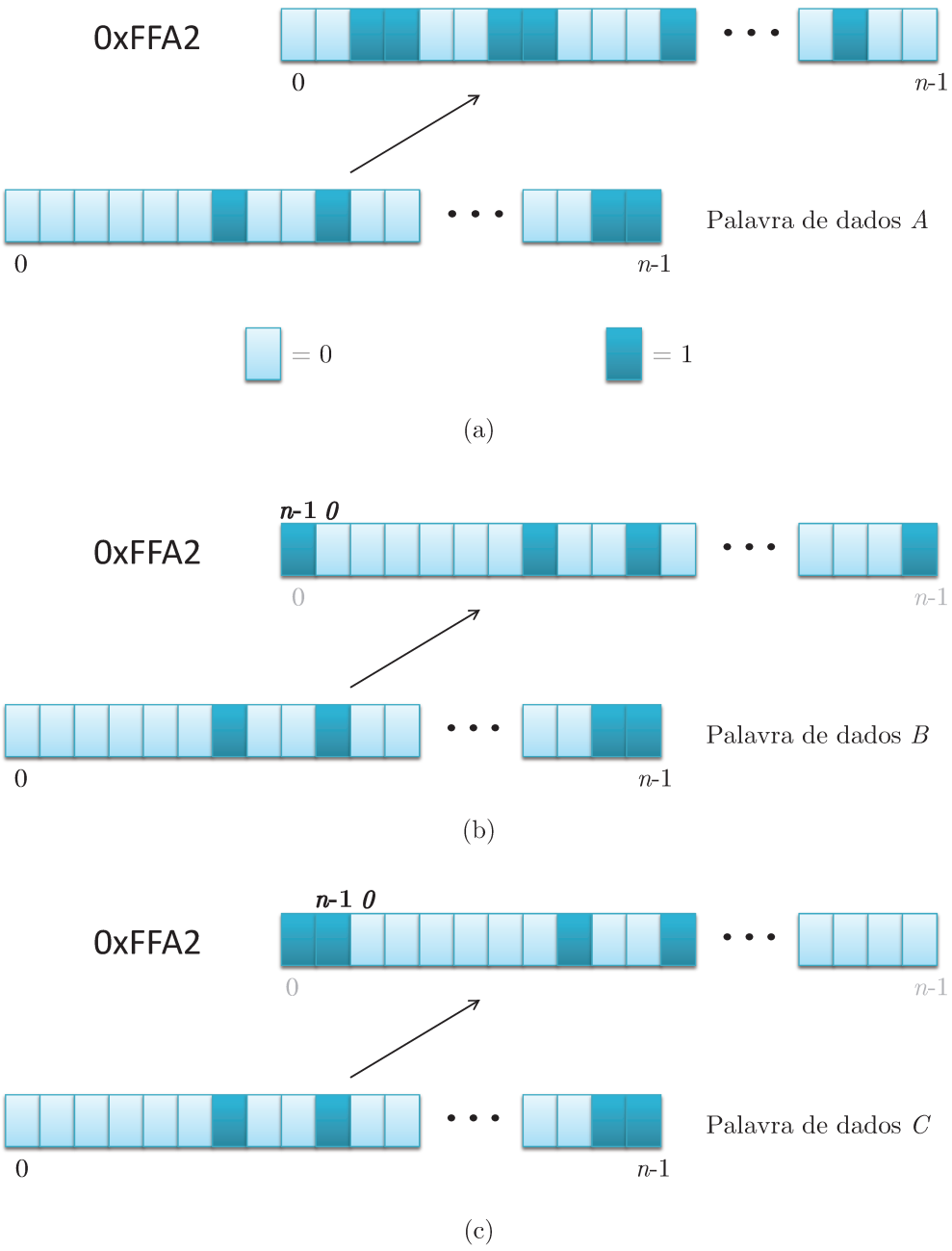


Figura 3.4: (a) No endereço 0xFFA2 existe uma palavra que será substituída pela palavra de dados A. (b) A palavra A foi rotacionada em uma posição e gravada no endereço 0xFFA2, a posição dos *bits* menos e mais significativos da palavra escrita estão em negrito, já dos *bits* originais estão em cinza claro; ademais, uma nova palavra B será escrita, coincidentemente, é a mesma escrita anteriormente. (c) A palavra B também foi rotacionada em uma posição, mas a partir da última rotação.

3.2 Probabilidade de *bit-flip* para técnicas de correção de erros

Quando uma memória é simulada no simulador descrito, não são realizadas as operações dos códigos corretores de erros, ao invés disso, apenas o efeito que eles teriam é levado em conta; isto é, se um código corretor corrige e erros, durante a simulação, uma linha de memória pode ter e *bits* com tempo de vida menor do que o *bit* analisado no passo atual da simulação, antes da linha ser considerada como falha. Em consequência disso, não é necessário adicionar os *bits* de metainformação à simulação. Todavia, um dos objetivos deste trabalho é justamente analisar o impacto incidente das técnicas de correção de erros no desgaste de uma PCM. Portanto, a forma de manter o simulador simples em seu funcionamento e fidedigno à realidade é ajustar as equações que contabilizam o tempo de vida (ou a quantidade de escritas).

Alguns trabalhos como [20, 25] expõem que a probabilidade de *bit-flip* é menor do que aquela normalmente usada na literatura, como implicação, menos *bits* de dados em uma escrita são modificados; porém, não se havia analisado ainda se isso também era válido para os *bits* de metainformação, especificamente, os *bits* relacionados aos códigos de correção de erros. Por isso, no modelo de simulação é adicionado o parâmetro Δ que informa a quantidade de *bits* extras de cada técnica simulada, informação utilizada no cálculo da probabilidade de *bit-flip* ajustada para as técnicas de correção de erros. Dessa forma, como a probabilidade de *bit-flip* influencia no número de escritas suportados por uma memória, é possível analisar os efeitos e a eficácia dos sistemas de recuperação de falhas.

3.2.1 ECP

Error-Correcting Pointers [43] é uma forma de corrigir erros em uma memória não-volátil através da substituição de células falhas por outras incólumes. Para isso, usa-se ponteiros para redirecionar a escrita daquelas para estas. Essa forma de correção elimina a necessidade de escrever na memória mais do que os n *bits* da informação, enquanto qualquer técnica que use alguma codificação para se recuperar de erros requer também a gravação dos *bits* de verificação.

Empregando-se o conceito de *wear-leveling* dentro de uma linha de memória, o desgaste na linha é diluído temporalmente com as células sobressalentes, usadas apenas na ocorrência de falhas permanentes. Ao rotacionar as gravações sobre elas, diminui-se a frequência de escritas no todo (Figura 3.5). Isso não ocorre em técnicas que usam codificação, pois os *bits* extras na linha são modificados a cada escrita para armazenar os *bits* de verificação.

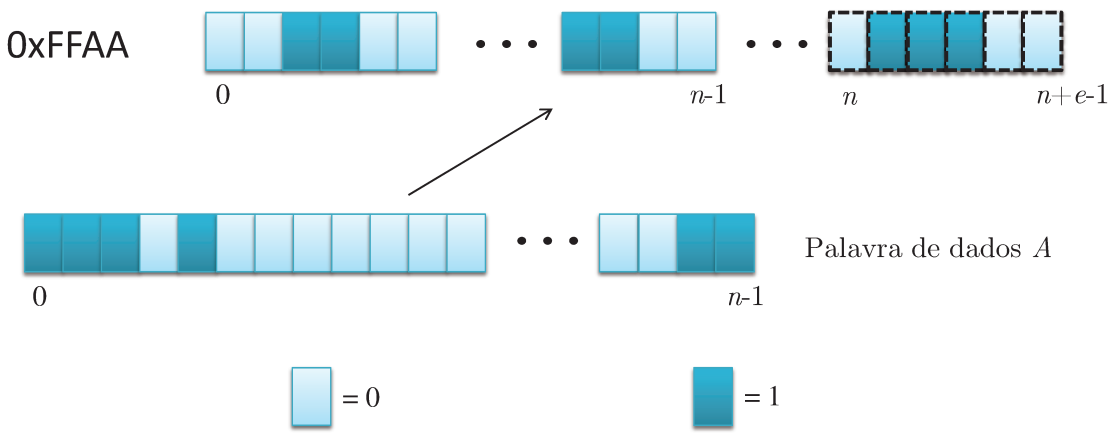


Figura 3.5: (a) No endereço 0xFFAA existe uma palavra que será substituída pela palavra de dados A. Os e *bits* sobressalentes (denotados pelo traço pontilhado) podem substituir e células com falhas. (b) A palavra A foi rotacionada em uma posição e gravada no endereço 0xFFAA; percebe-se que e *bits* são poupados em qualquer escrita, enquanto não há erros. A posição dos *bits* menos e mais significativos da palavra escrita estão em negrito.

Além dos *bits* sobressalentes da Figura 3.5, existem os *bits* que armazenam os ponteiros (com $\log_2(n)$ *bits*) de localização das células com falhas. Esses ponteiros são raramente escritos, sendo em duas situações modificados: quando uma célula falha e o ponteiro tem de indicar sua localização, ou quando isso acontece e o último ponteiro é alterado para apontar para próxima entrada ECP[5] a ser utilizada. Para uma linha com e entradas ECP (ECP_e) no pior caso o último ponteiro será escrito e vezes. Valor irrisório se comparado ao número de escritas que, em geral, uma linha pode sofrer. Além dos ponteiros, existe um *bit* na linha responsável por indicar se todas as entradas ECP já foram utilizadas. Dessa forma, o total de *bits* em uma linha com ECP_e é $N_{\text{ECP}_e} = n + e + e \cdot \log_2(n) + 1$.

Como os e ponteiros de tamanho $\log_2(n)$ e o *bit* indicador de uso de entradas ECP não possuem frequência de escritas – a probabilidade deles sofrerem escritas a cada passo de simulação é praticamente nula – considera-se suas probabilidades de *bit-flip* zero. Dessas

[5]Uma entrada ECP corresponde aos *bits* de ponteiro mais o *bit* sobressalente.

implicações, a probabilidade de *bit-flip* ajustada para a ECP_e é

$$P_{\text{ECP}_e} = p \cdot \frac{n+e}{N_{\text{ECP}_e}} \tag{3.8}$$

Apesar de se tentar modelar a probabilidade de maneira verossímil, algumas concessões são feitas; por exemplo, no simulador não é possível distinguir qual célula falha: pois não se discrimina qual é de dados ou de metainformação, com isso, não é possível subtrair de n ou de e as quantidades de células inutilizadas para dar maior precisão à modelagem, visto que um *bit* falho não sofre inversão e, portanto, deveria ser descontado na contabilidade de inversões. Embora esse fato seja indiferente para a Equação 3.8 – subtrair de n ou de e, tanto faz –, para outras técnicas, a probabilidade de *bit-flip* depende da função exercida pelo *bit*, *i.e.*, se armazena informação de dados ou de verificação. Portanto, a subtração de uma falha teria de ser precisa.

Para a simulação da técnica, a Equação 3.9 é usada para calcular o número total de escritas a cada passo da simulação. Por fim, na Tabela 3.1 mostra-se a comparação dos valores de p, sem ajuste (nominais), e P_{ECP_e} para ECP_6, considerando diferentes valores de p – ou seja, além do usual 50% não ajustado e utilizado na literatura [37, 43]. Observe que os valores ajustados são menores, devido à rotação e aos *bits* sem frequência de escrita.

$$\mathbb{W}(k) = \frac{W(k)}{P_{\text{ECP}_e}} \tag{3.9}$$

Para corroborar com os valores da tabela, foram feitas 1250 execuções de simulação de uma PCM com ECP de seis entradas[6] (ECP_6) com a seguinte configuração: $N = n = 512$, $\Delta = e = 6$, $N_L = 64$, $N_P = 256$, $\mu = 10^8$ e $\sigma = 2,5 \cdot 10^7$. Um arquivo de *profile* contendo número de escritas e porcentagem de páginas vivas na memória é gerado e alimentado com os resultados de cada execução da simulação. À exceção do parâmetro Δ, todos os outros manterão seus valores em qualquer uma das simulações subsequentes.

Como as porcentagens são fixas para uma determinado tamanho de memória, a média do número de escritas é calculada para cada porcentagem; o erro máximo (calculado pela equação de erro padrão [17]) obtido da média estimada de escritas para uma porcentagem é menor do que 0,2%, indicando que a técnica é pouco sensível à variação do tempo de vida das células das memórias geradas a cada simulação. Em termos reais, ela possui certa insusceptibilidade à variação do processo de fabricação. O maior erro acontece justamente na primeira porcentagem, por conta da variação do número de células que possuem tempo de vida nulo ou praticamente nulo, geradas a cada simulação.

[6] É a configuração mais usada em trabalhos relacionados por conta do *overhead* ficar próximo a 12,5%, valor usado amplamente como limite de tamanho para metainformações. É também o valor de *overhead* para um *bit* de paridade a cada *byte* ou do código SECDED(72,64). Mais detalhes, nas próximas subseções.

Tabela 3.1: Comparação entre valores de probabilidade de *bit-flip* com e sem ajuste para ECP_6. Dado $n = 512$ e $e = 6$. A terceira coluna é a redução porcentual da probabilidade ajustada.

p	P_{ECP_6}	%
10%	9,04%	9.6% ↓
20%	18,08%	9.6% ↓
30%	27,12%	9.6% ↓
40%	36,16%	9.6% ↓
50%	45,20%	9.6% ↓
60%	54,24%	9.6% ↓
70%	63,28%	9.6% ↓
80%	72,32%	9.6% ↓
90%	81,36%	9.6% ↓
100%	90,40%	9.6% ↓

Nas Figuras 3.6 à 3.15 são mostrados os gráficos da porcentagem de páginas vivas pelo número de escritas. As linhas tracejadas marcam o fim da simulação ou a falha da última página. Para as Figuras 3.6 à 3.10 os eixos são mantidos iguais em todos os gráficos para mostrar a influência da probabilidade de *bit-flip* na duração da memória, bem como nas Figuras 3.11 à 3.15, contudo, para estas últimas a abscissa tem escala diferente.

Trivialmente, quanto menor for a probabilidade de se modificar os *bits* a cada escrita, maior será a durabilidade da memória; embora a diferença no número de escritas entre as curvas seja maior no gráfico da Figura 3.6 – um pouco mais de 2 bilhões de escrita –, a relação entre as curvas de cada gráfico é constante, como apresentado na Tabela 3.1.

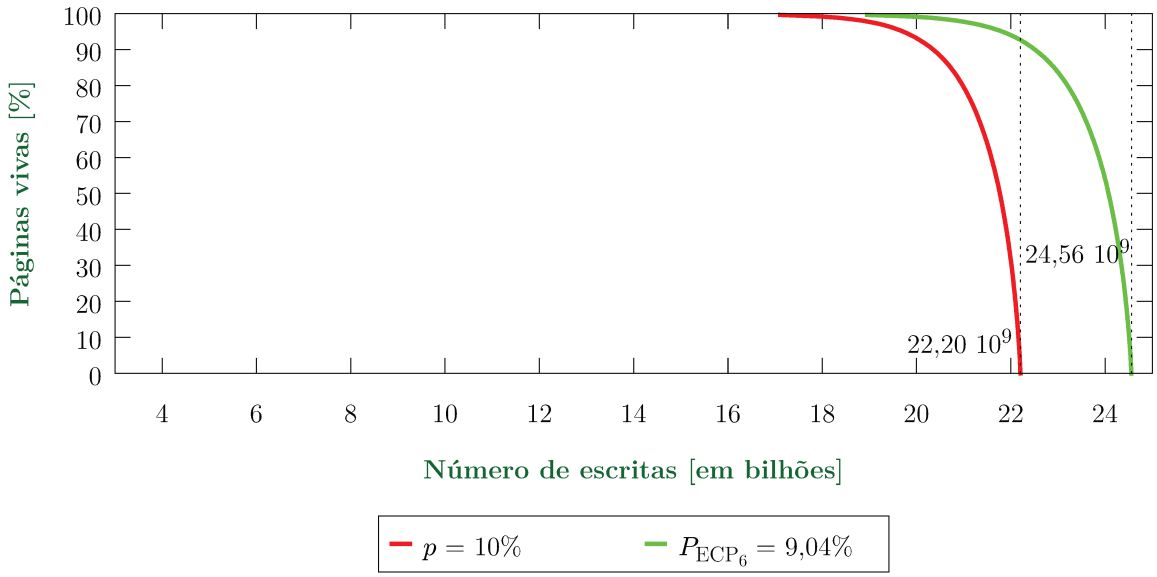


Figura 3.6: Resultados de simulações para uma memória com ECP_6, sendo $p = 0,10$ e $P_{\text{ECP}_6} = 0,0904$.

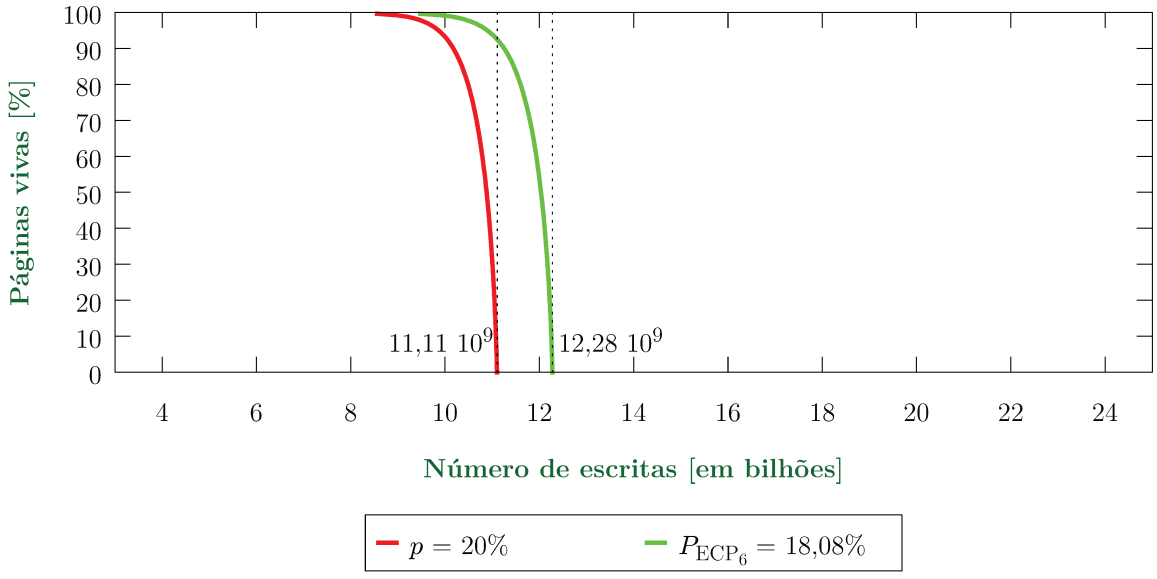


Figura 3.7: Resultados de simulações para uma memória com ECP_6, sendo $p = 0,20$ e $P_{\text{ECP}_6} = 0,1808$.

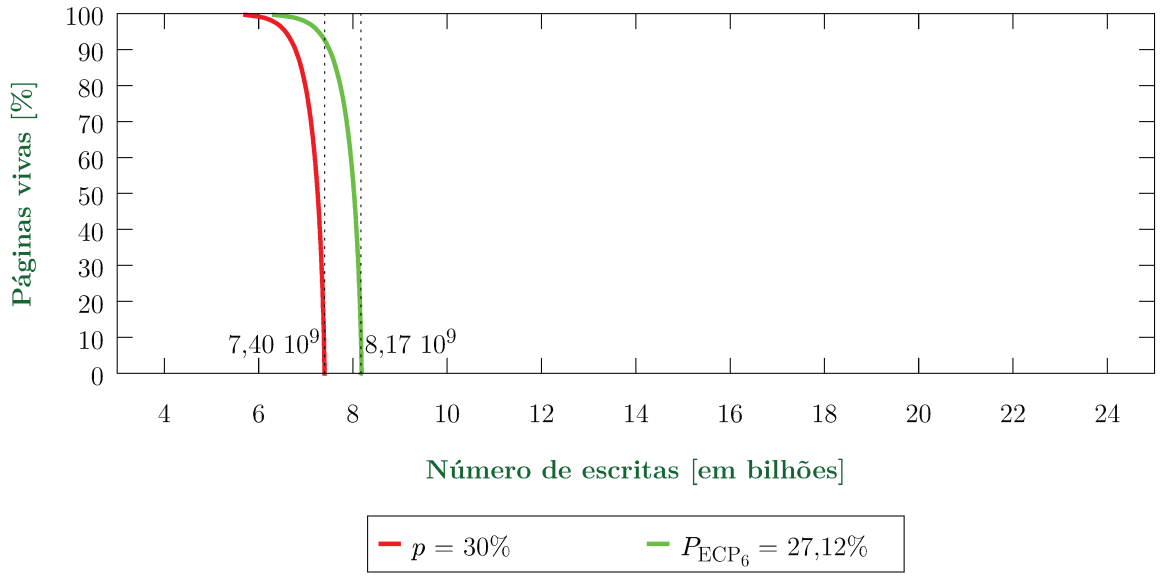


Figura 3.8: Resultados de simulações para uma memória com ECP_6, sendo $p = 0,30$ e $P_{\text{ECP}_6} = 0,2712$.

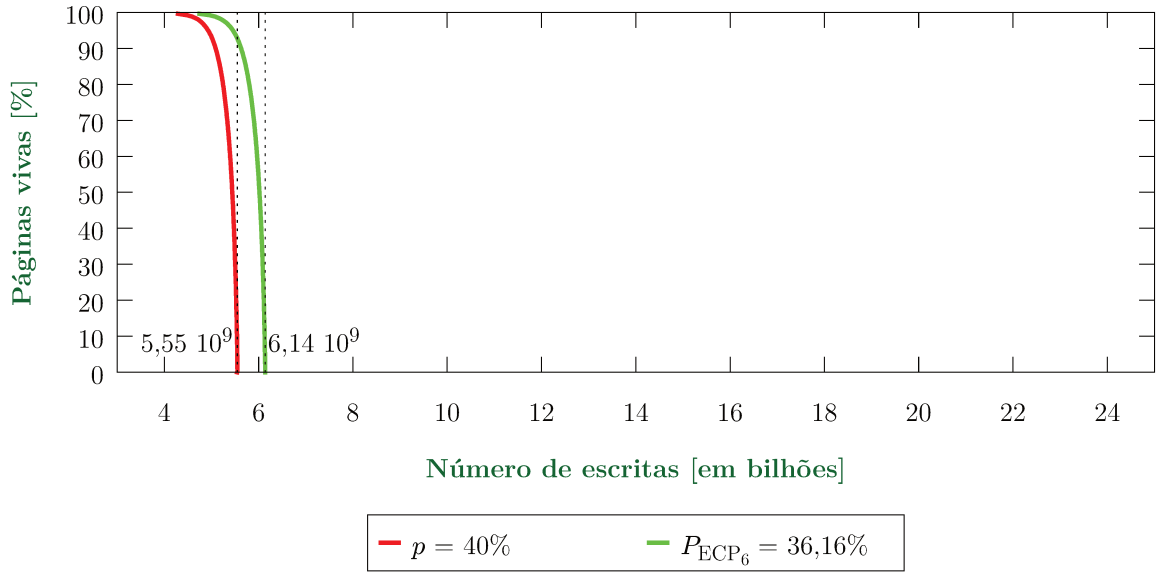


Figura 3.9: Resultados de simulações para uma memória com ECP_6, sendo $p = 0,40$ e $P_{\text{ECP}_6} = 0,3616$.

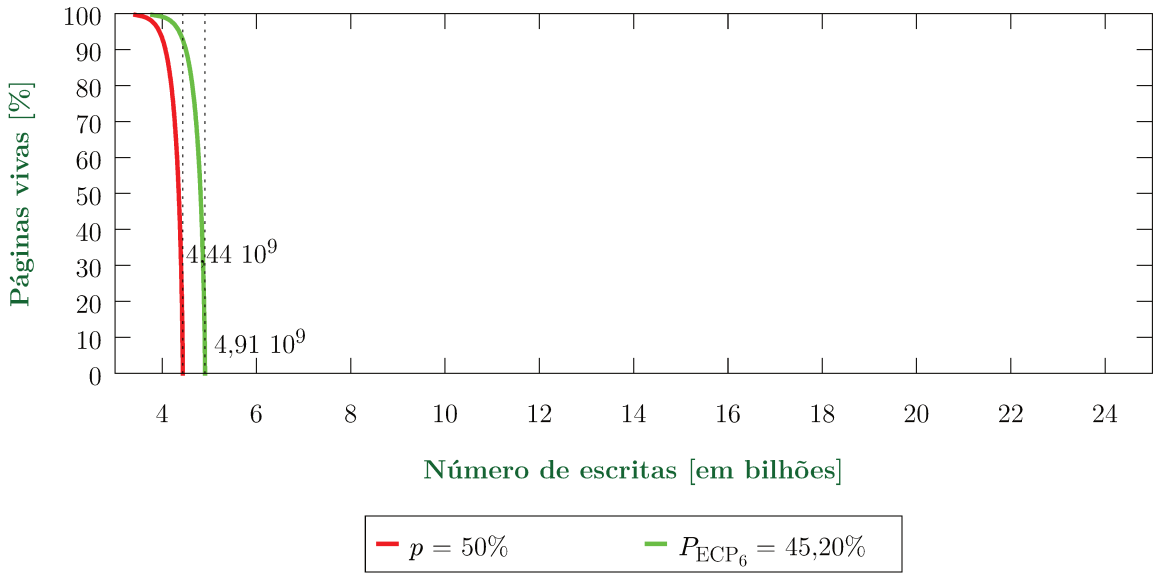


Figura 3.10: Resultados de simulações para uma memória com ECP_6, sendo $p = 0,50$ e $P_{\text{ECP}_6} = 0,4520$.

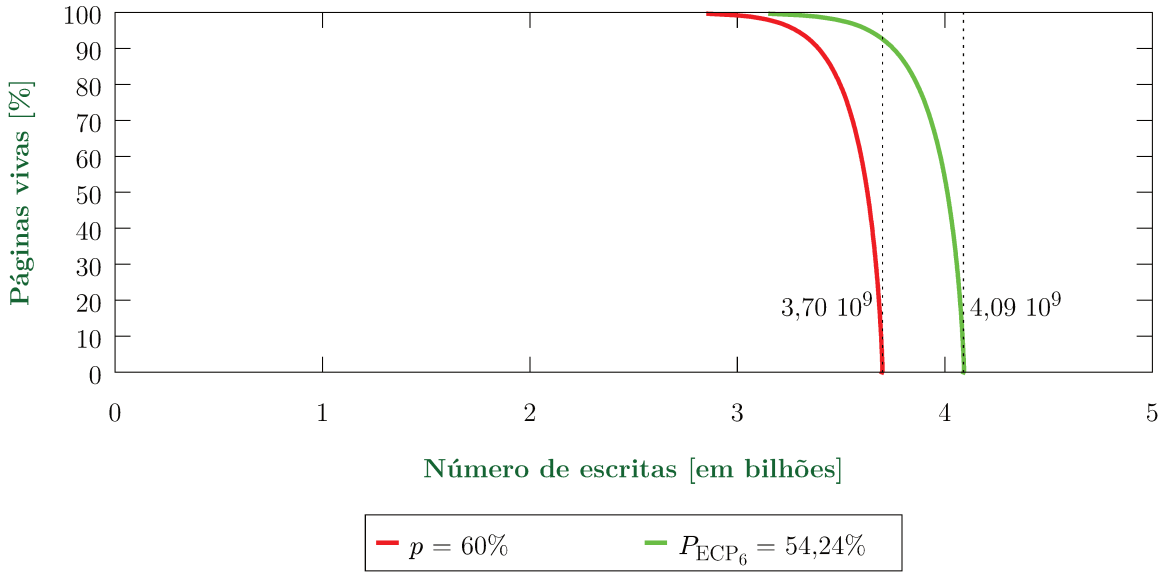


Figura 3.11: Resultados de simulações para uma memória com ECP_6, sendo $p = 0,60$ e $P_{\text{ECP}_6} = 0,5424$.

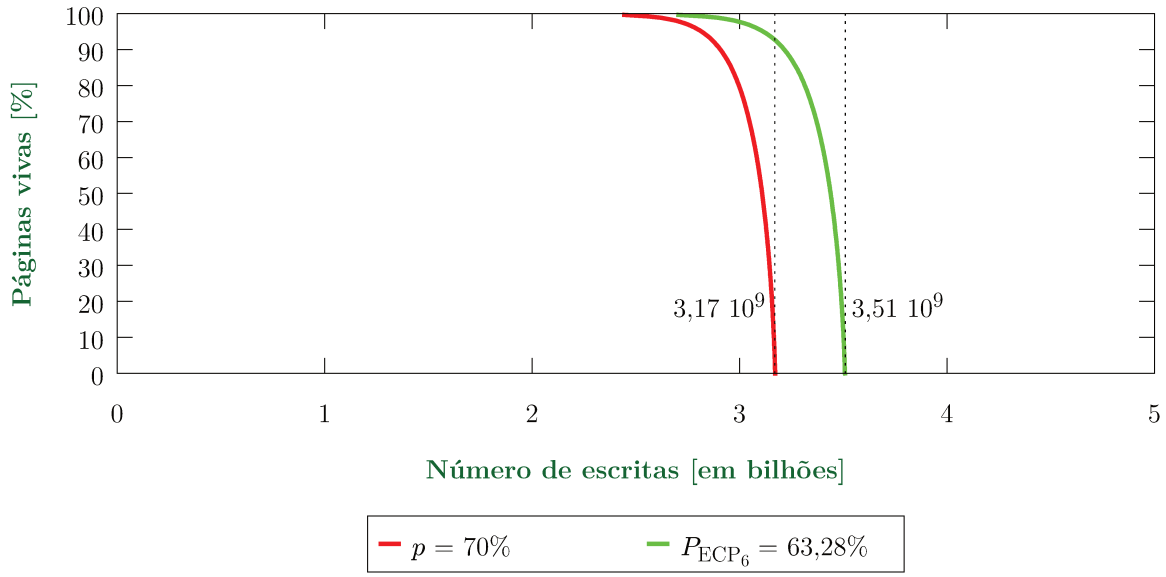


Figura 3.12: Resultados de simulações para uma memória com ECP_6, sendo $p = 0,70$ e $P_{\text{ECP}_6} = 0,6328$.

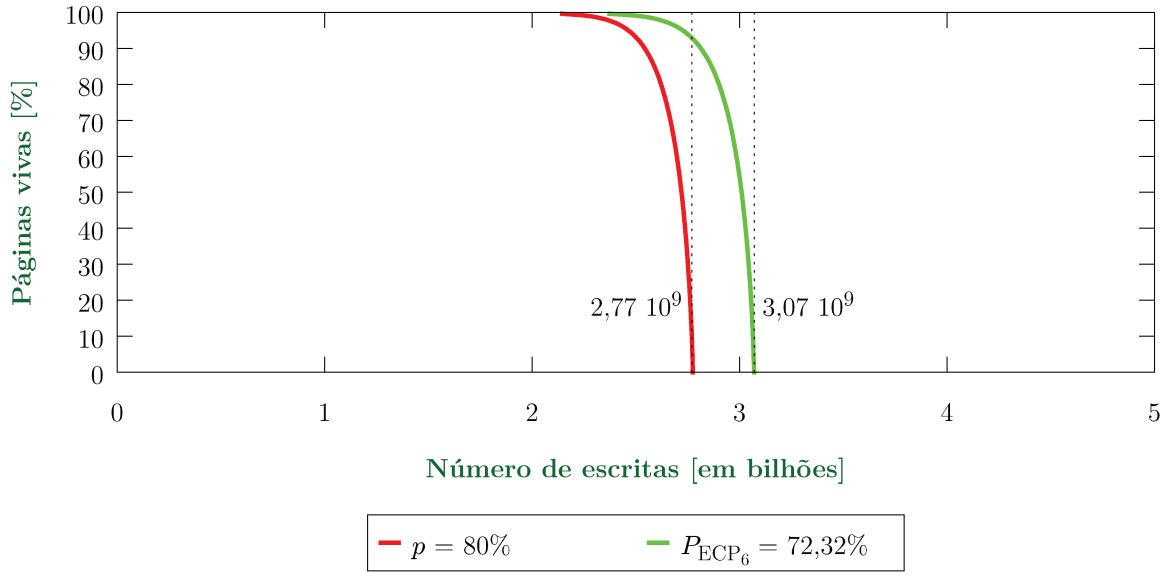


Figura 3.13: Resultados de simulações para uma memória com ECP_6, sendo $p = 0,80$ e $P_{\text{ECP}_6} = 0,7232$.

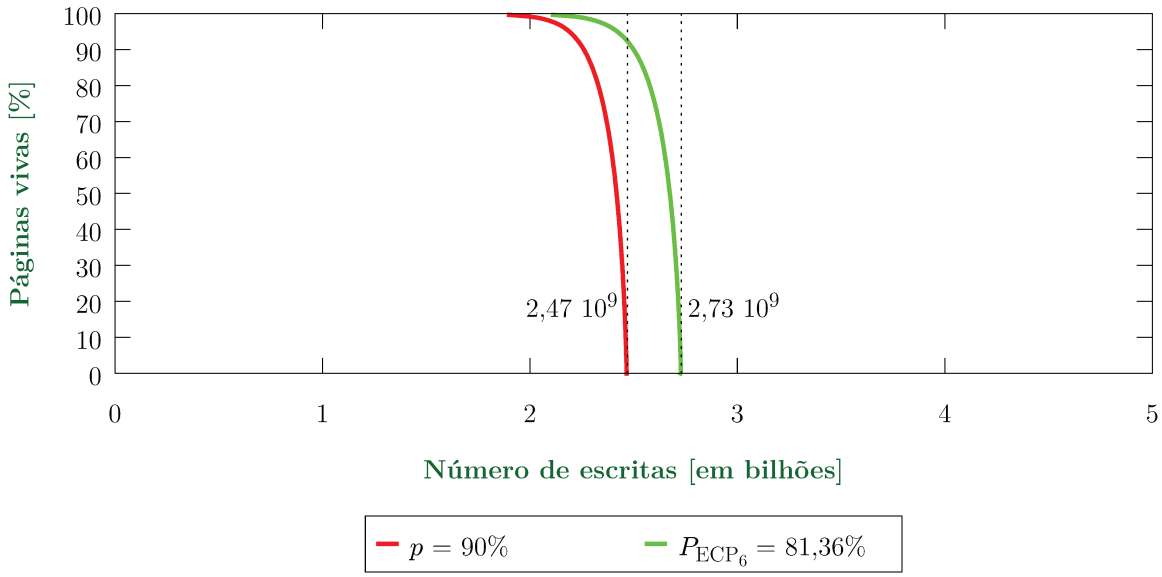


Figura 3.14: Resultados de simulações para uma memória com ECP_6, sendo $p = 0,90$ e $P_{\text{ECP}_6} = 0,8136$.

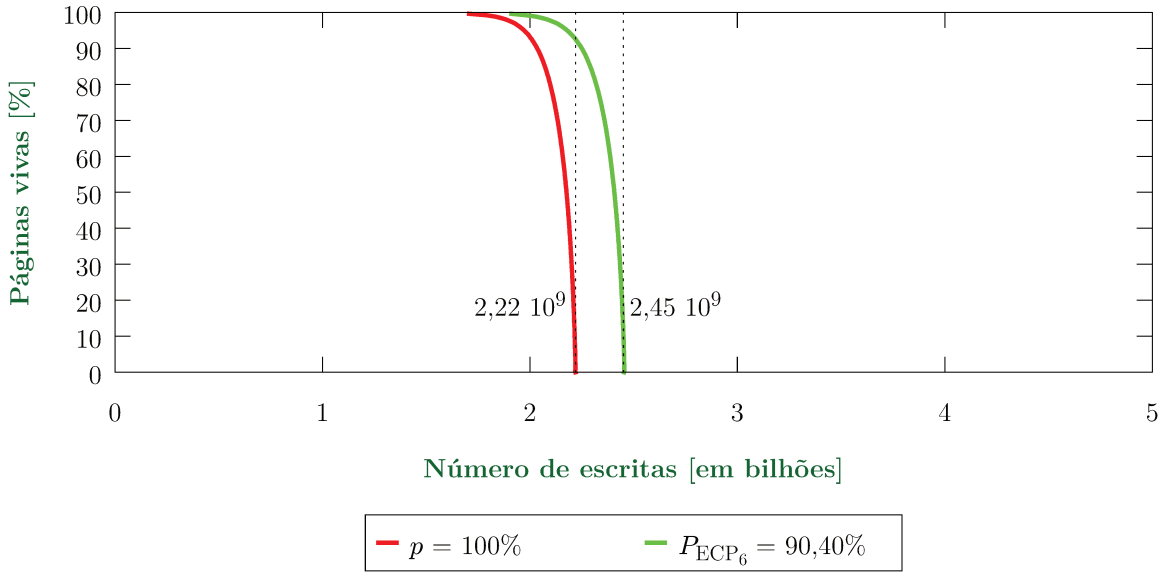


Figura 3.15: Resultados de simulações para uma memória com ECP_6, sendo $p = 1,00$ e $P_{\text{ECP}_6} = 0,9040$.

3.2.2 DRM

Dynamically Replicated Memory [23] é uma técnica que tem como principal objetivo recuperar páginas. Para tanto, ela precisa recuperar informação das linhas com erros e o faz usando a mais simples técnica existente de correção de erros: o cálculo da paridade. Para cada *byte* da linha existe um *bit* de paridade (12,5% de *overhead*). No total são $n/8$ *bits* de paridade em uma linha de memória, de tamanho n, cuja função é assegurar que as falhas são detectáveis.

A falha de um único *bit* invalida totalmente uma página; ela, daí, passa por um processo de reciclagem, sendo reutilizada assim que for possível emparelhá-la com outra página[7], desde que ambas não tenham *bytes* falhos concomitantemente localizados em um mesmo endereço interno. Assim, quando um *bit* de paridade sinaliza a falha de um *byte*, a escrita destinada a este é redirecionada à outra página, no mesmo endereço.

Como cada escrita de dados também incorre em escrita dos *bits* de paridade, todos os *bits* de uma linha estão sujeitos à *flips*. A despeito de existir *wear-leveling* intra-linha, os *bits* de paridade estão propensos a uma probabilidade de *bit-flip* diferente, devido à forma na qual são calculados. Sabe-se que um *bit* de paridade se modifica quando há um número ímpar de alterações nos *bits* de dados que ele resguarda; lembrando que a paridade é calculada pela operação $\oplus$ sobre todos os *bits* de dados (veja Figura 3.16).

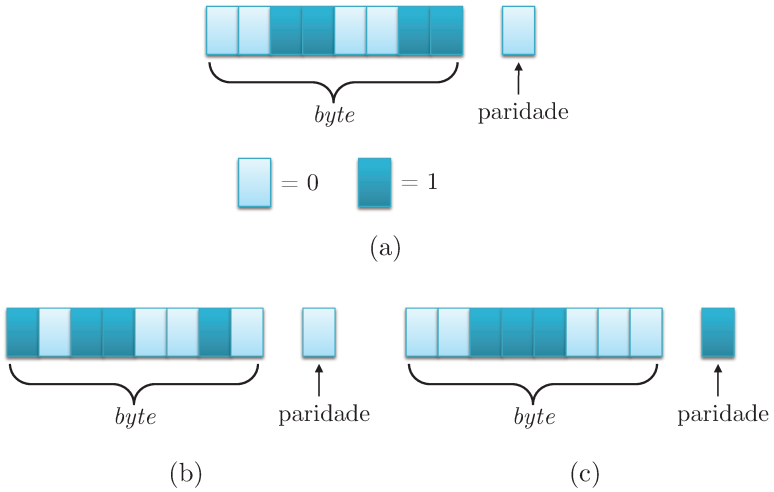


Figura 3.16: (a) Um *byte* e seu *bit* de paridade. (b) O *byte* sofre duas inversões, como o número de inversões é par, o *bit* de paridade não é modificado. (c) O *byte* passa por uma nova modificação mas agora de um número ímpar de inversões, portanto, a paridade é modificada.

[7]Um página pode ser emparelhada até atingir o limite de 160 falhas, após isto a página é descartada.

Dessa forma, o modelo de correlação entre os *bits* de dados e de paridade se dá pela probabilidade de acontecer uma quantidade ímpar de inversões na escrita dos dados. Logo, dado que n é sempre par,

$$P_{\text{PARIDADE}}(n,p) = \sum_{i=1}^{\frac{n}{2}} B(2\cdot i-1;n,p) \tag{3.10}$$

Onde $B(k;n,p)$ é a função densidade de probabilidade da distribuição binomial. Em suma, P_{PARIDADE} é a probabilidade de ocorrer um *flip* em n com probabilidade p, ou de ocorrerem três *flips* em n, ou de ocorrerem cinco *flips* em n e assim em diante. Seja e o número de *bits* de paridade para a técnica DRM, sendo que n/e são a quantidade de *bits* resguardados por um de paridade. A probabilidade de *bit-flip* para uma linha de memória é

$$P_{\text{DRM}} = \frac{e}{n+e}\cdot\left(p\cdot\frac{n}{e} + P_{\text{PARIDADE}}(n/e,p)\right)$$

n/e *bits* afetam um de paridade, sendo que isso ocorre e vezes na linha de memória que possui tamanho de $n+e$. Pode-se reescrever a equação acima como

$$P_{\text{DRM}} = p\cdot\frac{n}{n+e} + P_{\text{PARIDADE}}(n/e,p)\cdot\frac{e}{n+e} \tag{3.11}$$

Concluindo, o número total de escritas é calculado na simulação usando a equação 3.12 obtida substituindo a Equação 3.11 na Equação 3.6. Na Tabela 3.2 tem-se a comparação entre a probabilidade de *bit-flip* usada na literatura e a ajustada pela Equação 3.11.

$$\mathbb{W}(k) = \frac{W(k)}{P_{\text{DRM}}} \tag{3.12}$$

Diferentemente da ECP, existe um aumento na taxa de *bit-flip*, principalmente porque há uma contribuição muito forte da probabilidade P_{PARIDADE} a qual mantém-se muito próxima de 50% para quase todos valores de p (veja a Tabela 3.2). A grande consequência é a redução no tempo de vida da memória, visualizadas nos gráficos das Figuras 3.17 à 3.21, obtidos com os resultados de 1250 execuções de simulação de uma memória com os mesmos parâmetros da Seção 3.2.1, à exceção de $\Delta = e = 64$.

Para as probabilidades de *bit-flip* maiores do 50%, a probabilidade P_{PARIDADE}, por se manter em torno de 50%, contribui para a redução de P_{DRM}, como é visto nos gráficos das Figuras 3.22 à 3.26. Embora nessas condições o ajuste da probabilidade mostre uma situação mais favorável a utilização da paridade, deve-se observar que essas altas taxas de *bit-flip*, na verdade, estão reduzindo o tempo de vida da memória.

Dessa forma, a aplicação de técnicas como o *Flip-N-Write* – apresentada em [11] e explicada no Capítulo 2 – nas palavras de dados exclui a possibilidade de haver taxas de

Tabela 3.2: Comparação entre valores de probabilidade de *bit-flip* com e sem ajuste para DRM. Dado $n = 512$, *bits* de paridade $e = 64$. A segunda coluna é a probabilidade de *bit-flip* para um *bit* de paridade. A quarta coluna é a redução ou o aumento porcentual que a probabilidade de *bit-flip* sofre ao ser ajustada. Note que, quando a taxa de inversão de *bits* atinge 100%, a probabilidade de *bit-flip* da paridade é nula, resultado inerente à modelagem através da distribuição binomial.

p	P_{PARIDADE}	P_{DRM}	%
10%	41,61%	13,51%	35,10% ↑
20%	49,16%	23,24%	16,20% ↑
30%	49,96%	31,21%	4,03% ↑
40%	49,99%	41,11%	2,70% ↑
50%	50,00%	50,00%	0,00% –
60%	49,99%	58,89%	1,85% ↓
70%	49,96%	67,77%	3,19% ↓
80%	49,16%	76,57%	4,29% ↓
90%	41,61%	84,62%	5,98% ↓
100%	0,00%	88,89%	11,11% ↓

bit-flip maiores do que 50%, resultando em um aumento do tempo de vida da memória, porém, recaindo nas situações evidenciadas pelas Figuras 3.17 à Figuras 3.21 de maior desgaste.

O número de escritas em torno de 70% das páginas vivas tem alto desvio padrão, resultando em uma falta de confiabilidade no valor obtido (o erro mínimo logrado foi 70%); mas ao alcançar 60% das páginas, os erros se estabilizam abaixo de 1% e caminham para menos do que 0,2% conforme o número de páginas diminui. Essa instabilidade é inerente à técnica, pois ela não se recupera das primeiras falhas até conseguir emparelhar duas páginas, fazendo a variabilidade no tempo de vidas de células de cada execução afetar os primeiros emparelhamentos e, por conseguinte, a estabilidade do número de páginas vivas no início da simulação.

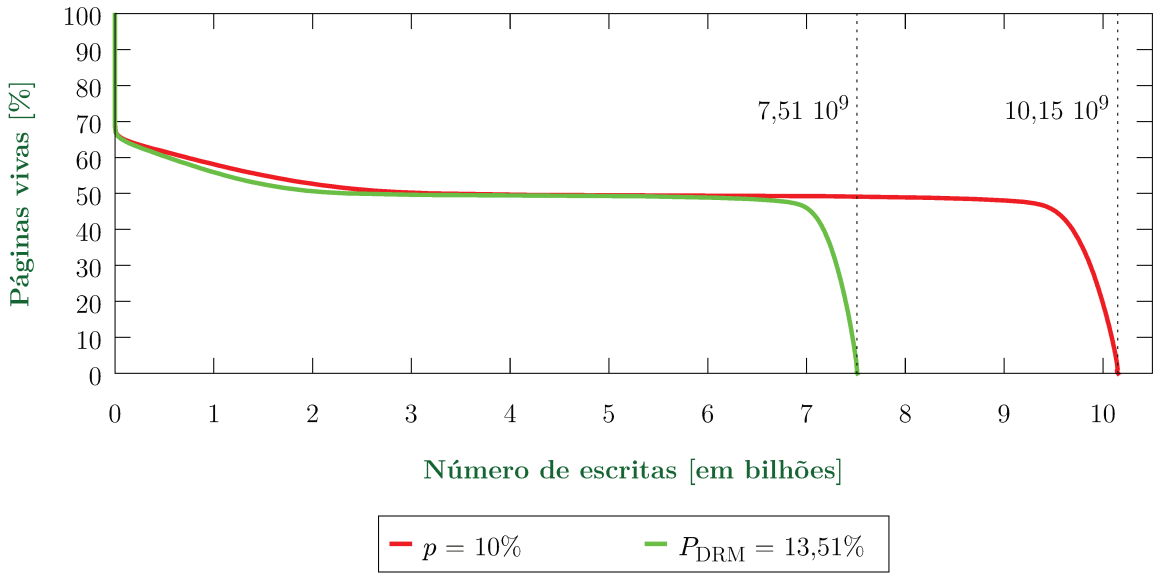


Figura 3.17: Resultados de simulações para uma memória com DRM, sendo $p = 0,10$ e $P_{\text{DRM}} = 0,1351$.

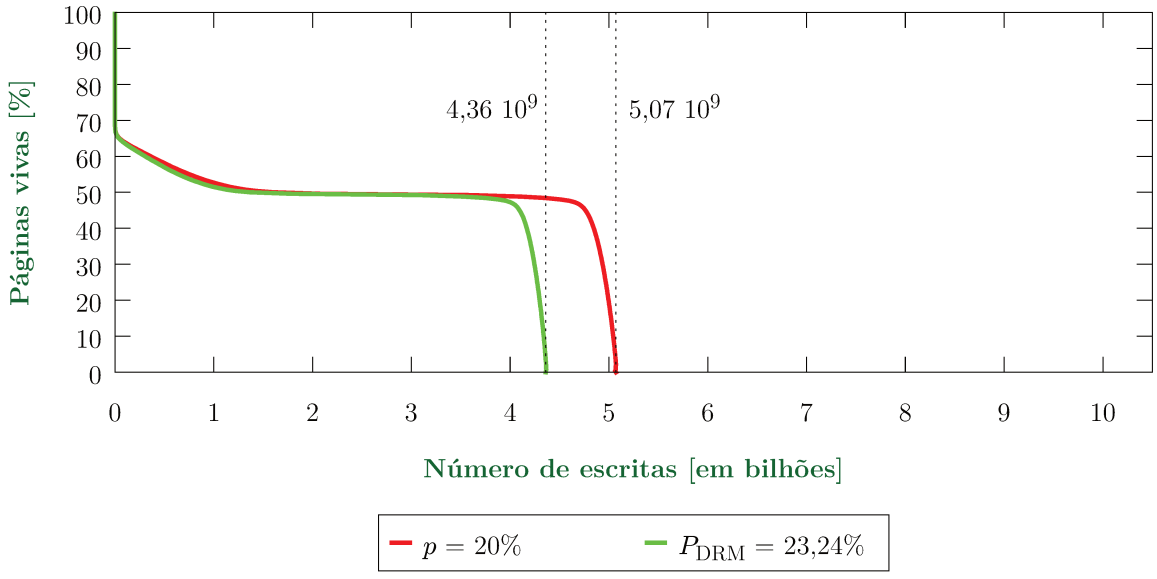


Figura 3.18: Resultados de simulações para uma memória com DRM, sendo $p = 0,20$ e $P_{\text{DRM}} = 0,2324$.

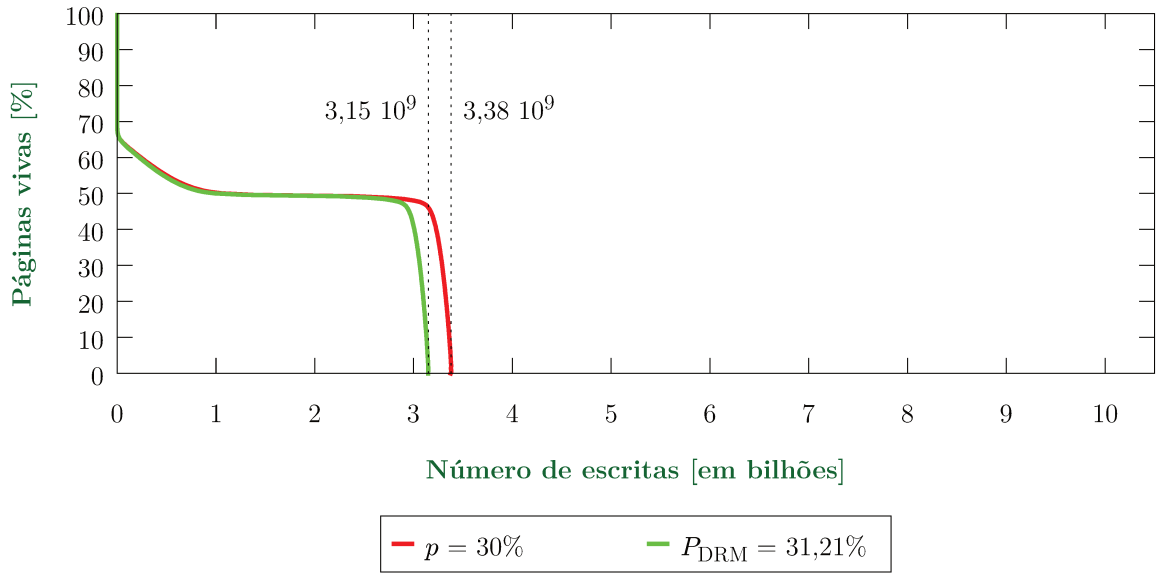


Figura 3.19: Resultados de simulações para uma memória com DRM, sendo $p = 0,30$ e $P_{\text{DRM}} = 0,3121$.

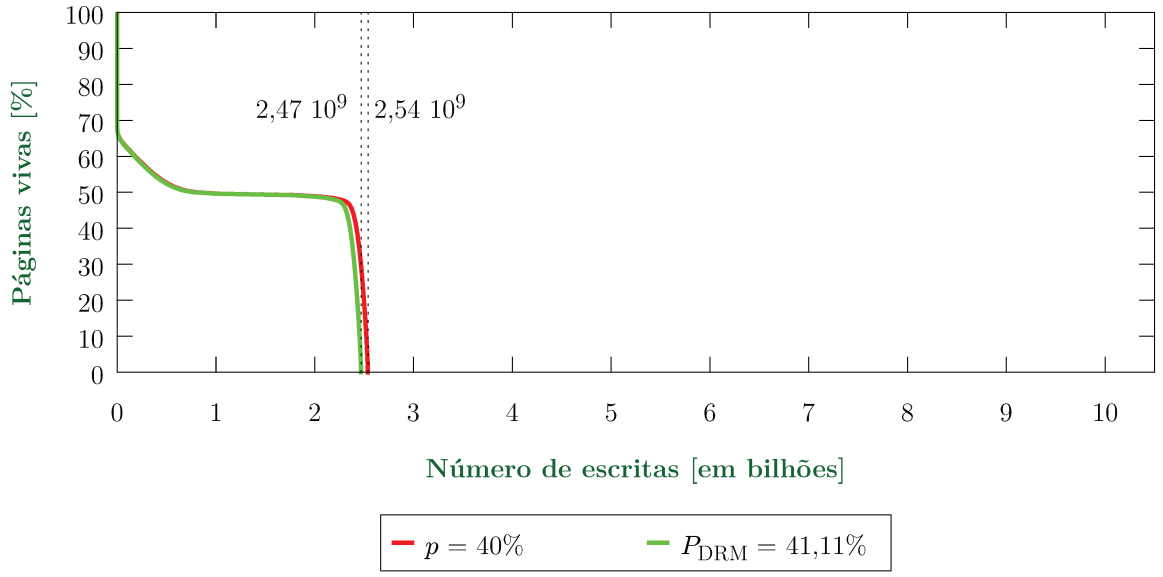


Figura 3.20: Resultados de simulações para uma memória com DRM, sendo $p = 0,40$ e $P_{\text{DRM}} = 0,4111$.

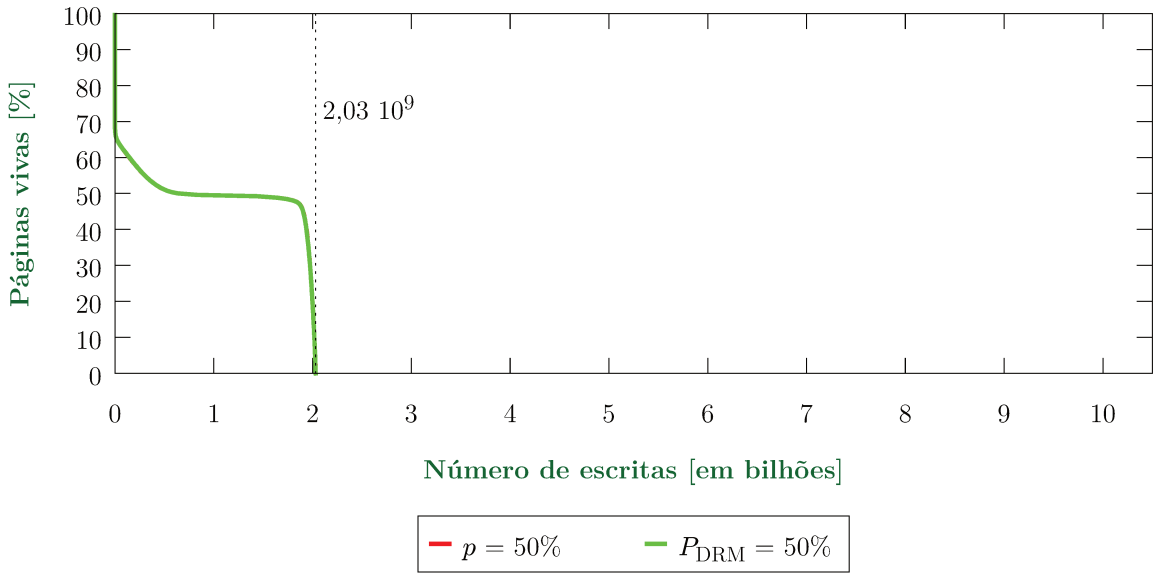


Figura 3.21: Resultados de simulações para uma memória com DRM, sendo $p = 0,50$ e $P_{\text{DRM}} = 0,5000$.

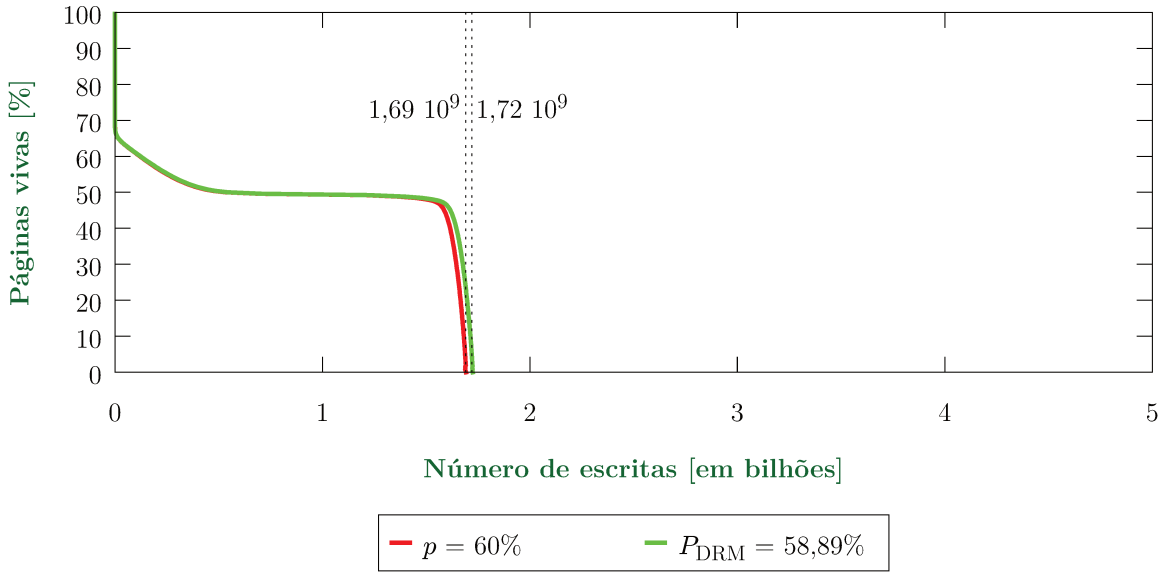


Figura 3.22: Resultados de simulações para uma memória com DRM, sendo $p = 0,60$ e $P_{\text{DRM}} = 0,5889$.

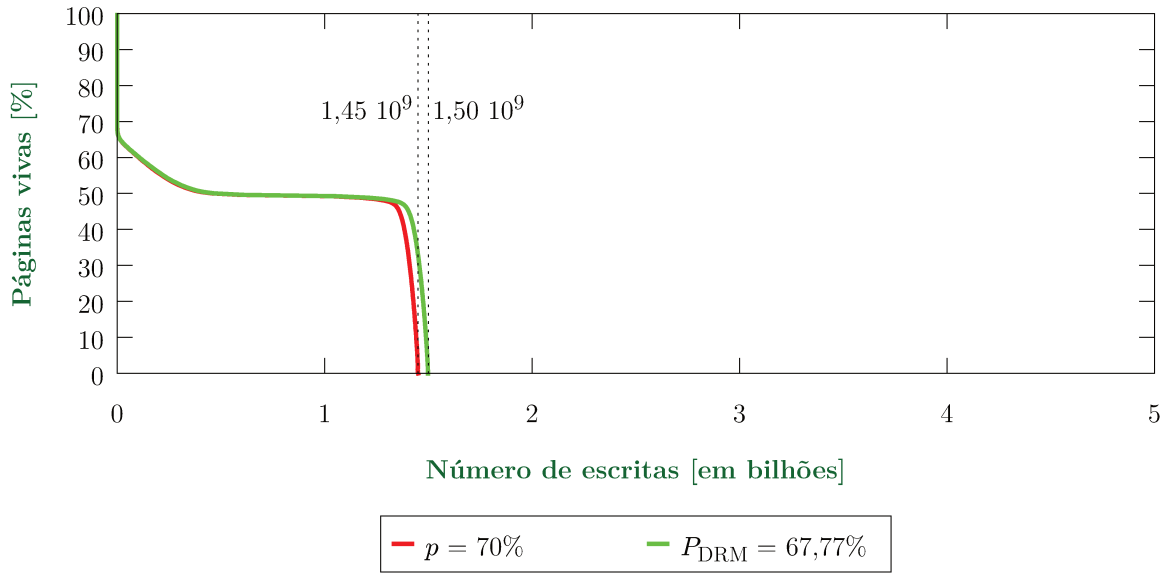


Figura 3.23: Resultados de simulações para uma memória com DRM, sendo $p = 0,70$ e $P_{\text{DRM}} = 0,6777$.

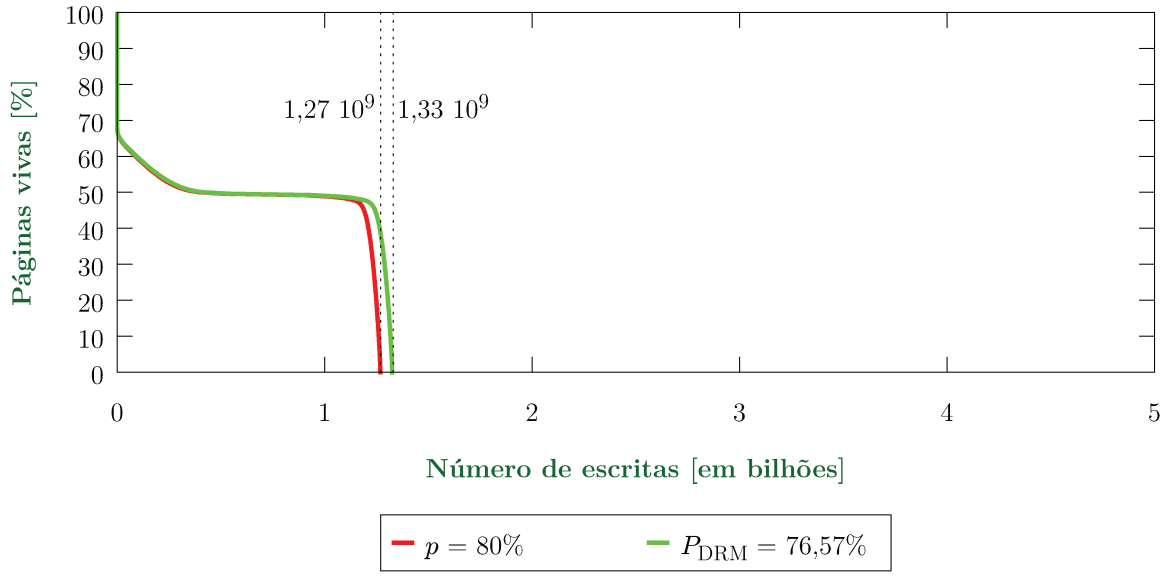


Figura 3.24: Resultados de simulações para uma memória com DRM, sendo $p = 0,80$ e $P_{\text{DRM}} = 0,7657$.

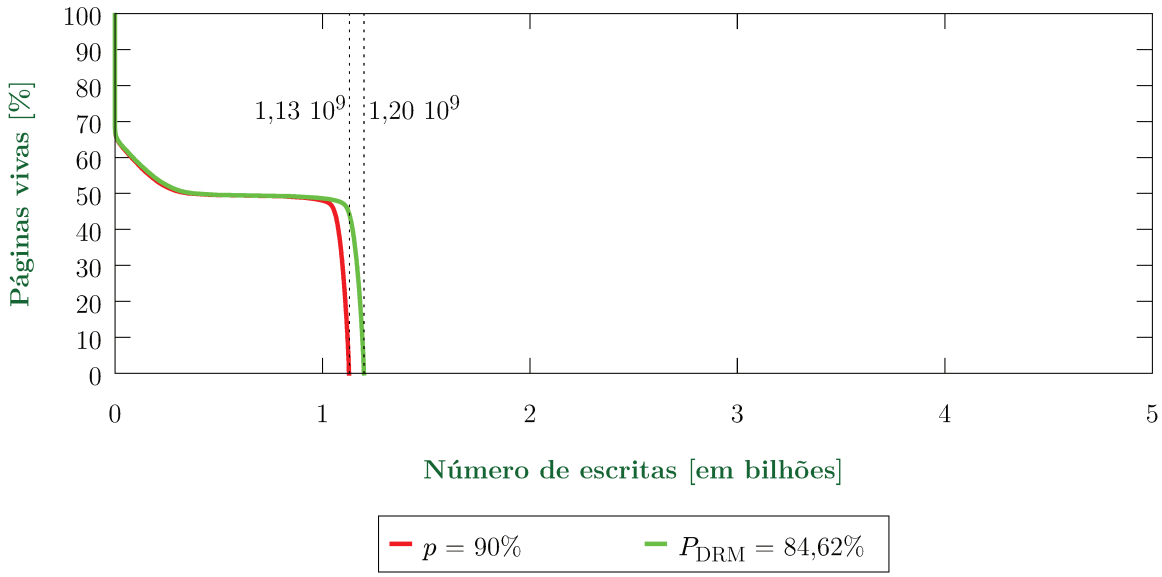


Figura 3.25: Resultados de simulações para uma memória com DRM, sendo $p = 0,90$ e $P_{\text{DRM}} = 0,8462$.

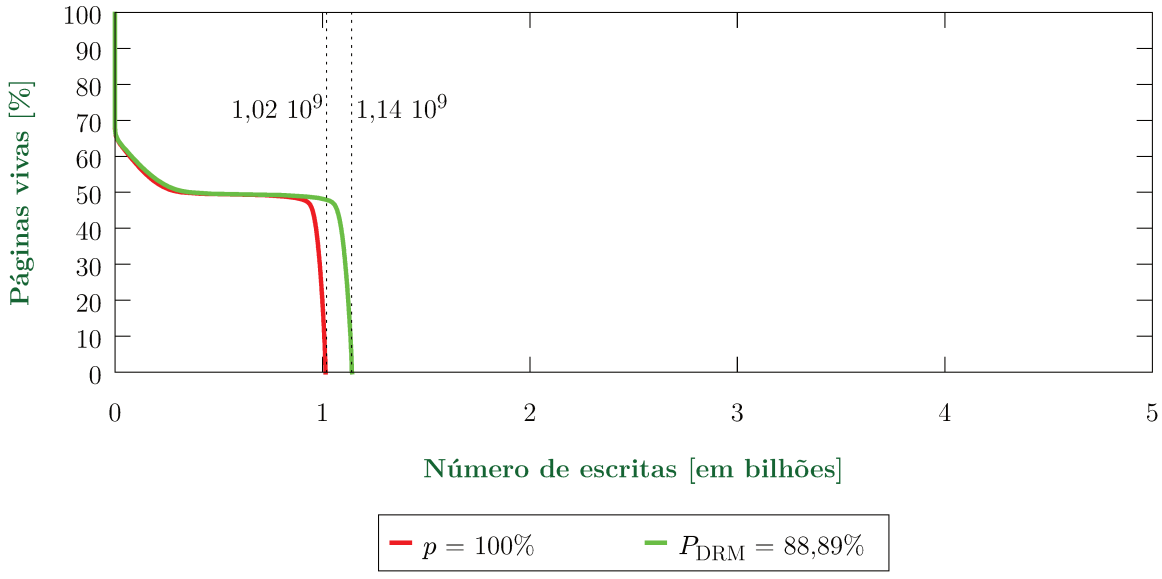


Figura 3.26: Resultados de simulações para uma memória com DRM, sendo $p = 1,00$ e $P_{\text{DRM}} = 0,8889$.

3.2.3 SECDED

A junção de um código de Hamming com um *bit* de paridade adicional, conhecido como *Single Error Correction and Double Error Detection* [24], é o código de correção de erros usualmente empregado em DRAM. Pela sua importância e antecipando algum trabalho futuro, analisou-se o comportamento da taxa de *bit-flip* de seus *bits* de verificação. O código de Hamming pode ser descrito como um conjunto de *bits* de paridade que cobrem redundantemente os *bits* de dados. Na Tabela A.1 temos um exemplo do que isso significa para uma palavra-código[8] de 31 *bits*, 26 *bits* de dados e 5 *bits* de paridade, *i.e.*, Hamming(31,26).

Em termos de parâmetros, o código de Hamming é definido da seguinte forma[9]:

Tamanho do código:	$n = 2^m - 1$
Número de *bits* de informação:	$k = 2^m - m - 1$
Número de *bits* de verificação:	$n - k = m$
Descrito por Hamming(n,k)	
$m \in \mathbb{N} \wedge m \geq 3$	

Na DRAM costuma-se usar $m = 7$ e, em vez de $n = 2^7 - 1 = 127$ *bits*, usa-se 71 *bits* de palavra-código, sendo que a diferença de $127 - 71 = 56$ *bits* são completados por zeros – em uma operação denominada *padding* – para que as operações sobre o código de 127 *bits* sejam corretamente executadas. Desses 71 *bits*, 7 são de paridade. Para formar o SECDED, acrescenta-se 1 *bit* de paridade cobrindo os 71 *bits* e, assim, forma-se o código SECDED(72,64). Na linha de memória na qual armazena-se mais do que uma palavra de dados, o código é aplicado à porções de 64 *bits*, mantendo seu *overhead* de 12,5% para a metainformação.

Na literatura pesquisada não foi encontrada uma forma de quantizar os *bits* cobertos por cada *bit* de paridade no código de Hamming, principalmente, para códigos que se utilizam de *padding*. Por isso, a Equação 3.13 foi deduzida e, a despeito da falta de uma prova formal, para fins validação são ilustradas as duas instâncias usadas neste trabalho no Apêndice A. Para n *bits* de palavra-código tem-se

$$k_p(j) = j \cdot \left(\left\lceil \frac{n-(j-1)}{j} \right\rceil - \left\lceil \frac{n-(j-1)}{2 \cdot j} \right\rceil \right)$$

$$q_p(j) = k_p(j) + \max\Big(0, n - (j-1) - 2 \cdot k_p(j)\Big) - 1 \tag{3.13}$$

Onde $q_p(j)$ fornece a quantidade de *bits* de dados cobertos para o j-ésimo *bit* da palavra-código, sendo $1 \leq j \leq n$. O i-ésimo bit de paridade está localizado na j-ésima

[8] Do inglês *codeword*. Significa a palavra composta pelos *bits* de dados mais os *bits* de verificação.
[9] Informações adaptadas de [31].

posição pela relação $j = 2^{i-1}$. A Tabela 3.3, construída a partir das Tabelas A.1 e A.2, mostra a quantidade de *bits* cobertos por cada *bit* de paridade dos códigos de Hamming(31,26) e Hamming(71,64), este último é uma apropriação da notação, pois ele não existe pela definição. Observe que, na palavra-código regular, existe uma constância na quantidade de *bits* de dados cobertos pelos de paridade, dado por $2^{m-1} - 1$, mas para a palavra-código irregular, os valores são distintos.

Uma vez obtidas as coberturas dos *bits* de paridade, usa-se a Equação 3.10 para se obter a probabilidade de *bit-flip* de uma palavra-código

$$P_{\text{HAMMING}} = \frac{1}{n} \cdot \left(k \cdot p + \sum_{i=1}^{m} P_{\text{PARIDADE}}(q_p(2^{i-1}), p) \right)$$

Agora, para o código SECDED a probabilidade de *bit-flip* se dá por

$$P_{\text{SECDED}} = \frac{1}{n+1} \cdot \left(n \cdot P_{\text{HAMMING}} + P_{\text{PARIDADE}}(n, P_{\text{HAMMING}}) \right) \tag{3.14}$$

Note que a Equação 3.14 é por bloco, mas vale para a linha de memória inteira, pois multiplicar P_{SECDED} pelo número de blocos e dividir pelo tamanho da linha não altera seu valor, porque o tamanho da linha é $n+1$ multiplicado pelo número de blocos. Na Tabela 3.4 mostra-se a probabilidade de *bit-flip* sem ajuste e o aumento (ou redução) incorrido pelo ajuste. Nessa tabela, deve-se notar que a contribuição da codificação de Hamming para o aumento da probabilidade é um pouco mais incisiva do que da paridade, como visualizado na Tabela 3.2. Além disso, o *bit* extra de detecção agrava mais a situação, uma vez que $P_{\text{PARIDADE}}(n, P_{\text{HAMMING}})$ é 50% para todos os casos.

As Figuras 3.27 à 3.31 são resultados da execução de 1250 simulações de uma PCM configurada com $\Delta = 64$ e com os outros parâmetros como de praxe. Considerando que

Tabela 3.3: Quantidade de *bits* cobertos pelos códigos de Hamming(31,26) e Hamming(71,64). Vale observar que os valores excluem o *bit* de paridade, o qual sempre cobre a si mesmo; i indica o i-ésimo *bit* de paridade, coberto pelo *bit* de paridade h_j, localizado na j-ésima posição da palavra código, tal que $j = 2^{i-1}$.

Hamming(31,26)			Hamming(71,64)		
i	h_j	$q_p(j)$	i	h_j	$q_p(j)$
1	h_1	15	1	h_1	35
2	h_2	15	2	h_2	35
3	h_4	15	3	h_4	35
4	h_8	15	4	h_8	31
5	h_{16}	15	5	h_{16}	31
-	-	-	6	h_{32}	31
-	-	-	7	h_{64}	7

uma linha de memória tem b blocos SECDED, $n = (N+\Delta)/b$ e $k = N/b$. Igualmente, para as técnicas anteriores, o total de escritas é dado pela Equação 3.15, lograda substituindo-se p, na Equação 3.6, pela Equação 3.14:

$$\mathbb{W}(k) = \frac{W(k)}{P_{\text{SECDED}}} \tag{3.15}$$

Analisando as Figuras 3.32 à 3.36, as considerações a serem feitas são as mesmas apresentadas em relação à técnica DRM, ou seja, a probabilidade de *bit-flip* para o código de Hamming não se distancia dos 50%, levando P_{SECDED} ter valor menor do que a probabilidade nominal quando esta é maior do que 50%. Todavia, na prática, essas altas taxas seriam amenizadas por algum mecanismo (como o *Flip-N-Write*) por conta do baixo número de escritas toleráveis em comparação com as probabilidades nominais, menores do que 50%.

O maior erro padrão para a média do número de escritas é aproximadamente 3% e depois se estabiliza para valores menores. É interessante notar que essa taxa de erro é bem maior do que aquela apresentada pela técnica ECP e que se refere a primeira porcentagem de páginas que falham. Como o código de correção SECDED(72,64) não é capaz de corrigir dois erros dentro do bloco de 72 *bits*, falhas no processo de fabricação onde duas células no bloco tenham durabilidade nula, provocam imediatamente a falha de toda a página. Com a execução de várias simulações, fica recorrente a alternância entre

Tabela 3.4: Comparação entre valores de probabilidade de *bit-flip* com e sem ajuste para SECDED. Dado que uma linha possui oito blocos de SECDED(72,64). A segunda coluna é a probabilidade de *bit-flip* para os *bits* de paridade do código de Hamming. Já a quarta coluna é a redução ou o aumento porcentual que a probabilidade de *bit-flip* sofre ao ser ajustada. Note que, quando a taxa de inversão de *bits* atinge 100%, a probabilidade de *bit-flip* do código de Hamming é nula, resultado inerente à modelagem através da distribuição binomial.

p	P_{HAMMING}	P_{SECDED}	%
10%	48,47%	14,29%	42,90% ↑
20%	49,80%	23,31%	16,55% ↑
30%	49,98%	32,22%	7,40% ↑
40%	49,99%	41,11%	2,77% ↑
50%	50,00%	50,00%	0,00% –
60%	50,00%	58,89%	1,85% ↓
70%	50,01%	67,78%	3,17% ↓
80%	50,20%	76,69%	4,13% ↓
90%	51,53%	85,70%	4,78% ↓
100%	0,00%	89,58%	10,42% ↓

os casos nos quais páginas inteiras são geradas falhas e o caso no qual nenhuma página é falha no início da simulação. Portanto, gerando esse erro mais acentuado. O que ressalta, de certa forma, a robustez da ECP.

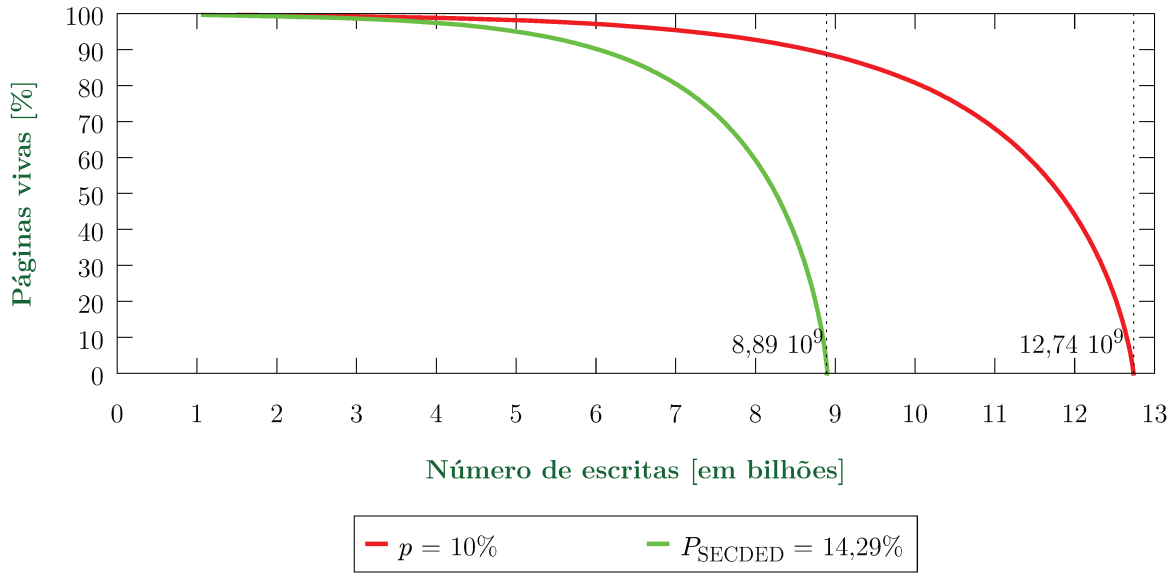


Figura 3.27: Resultados de simulações para uma memória com SECDED, sendo $p = 0,10$ e $P_{\text{SECDED}} = 0,1429$.

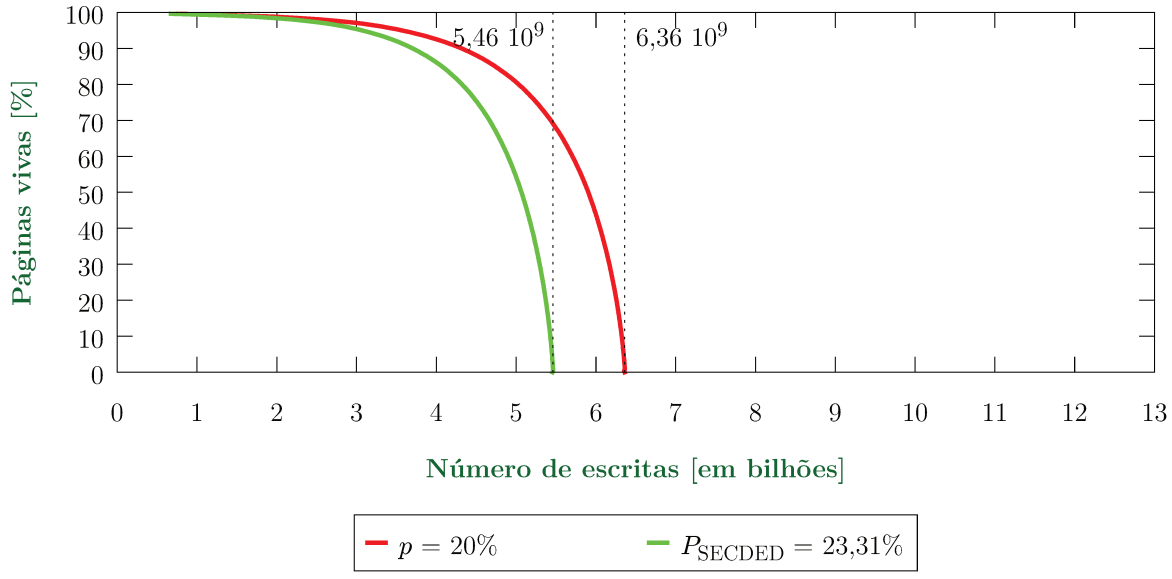


Figura 3.28: Resultados de simulações para uma memória com SECDED, sendo $p = 0,20$ e $P_{\text{SECDED}} = 0,2331$.

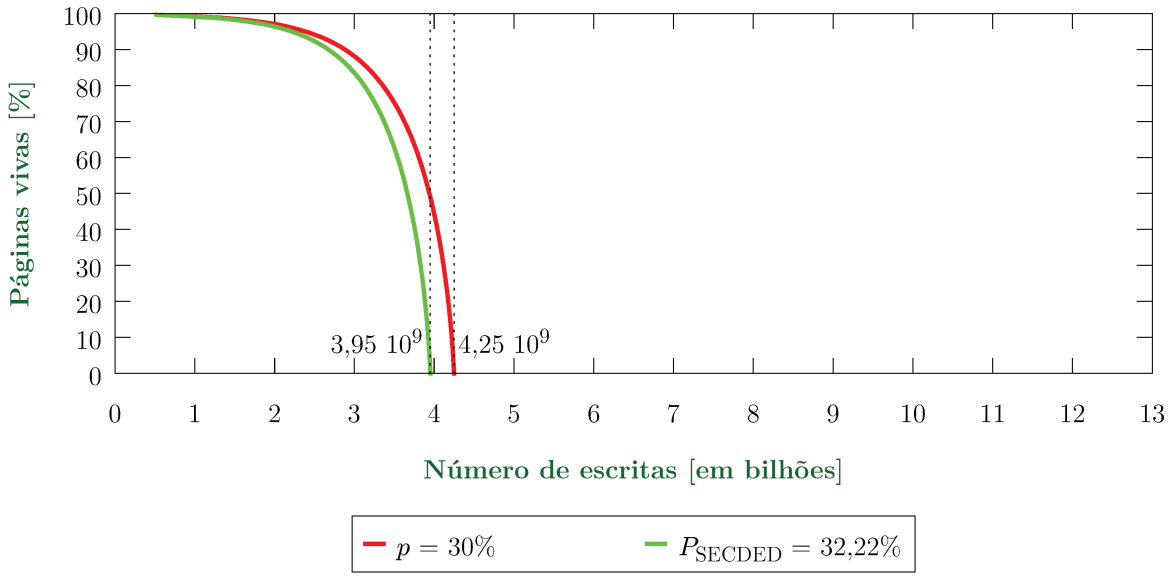


Figura 3.29: Resultados de simulações para uma memória com SECDED, sendo $p = 0,30$ e $P_{\text{SECDED}} = 0,3222$.

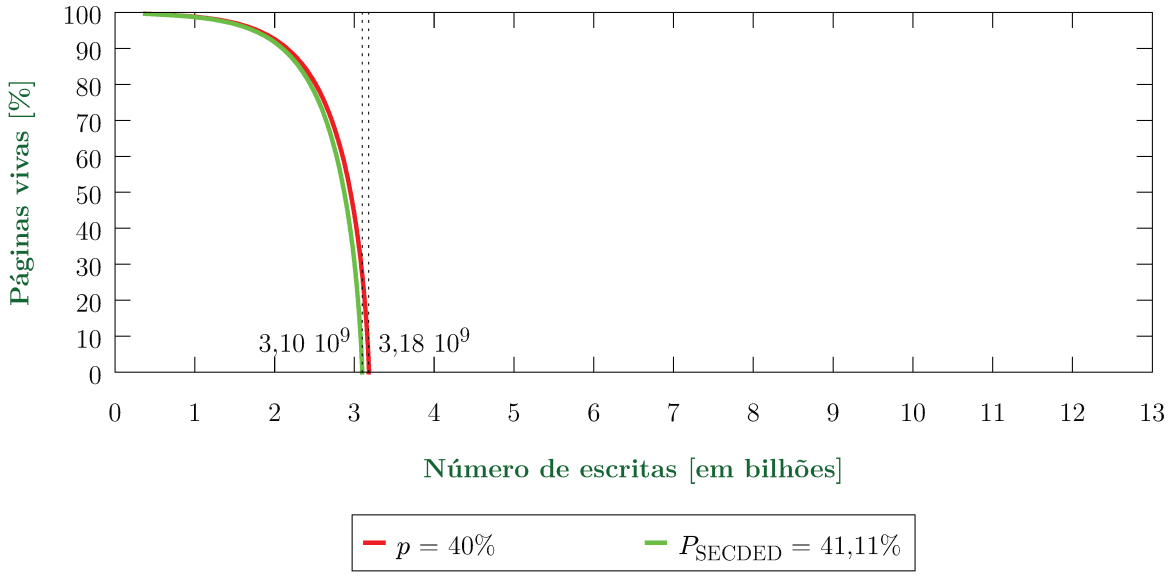


Figura 3.30: Resultados de simulações para uma memória com SECDED, sendo $p = 0,40$ e $P_{\text{SECDED}} = 0,4111$.

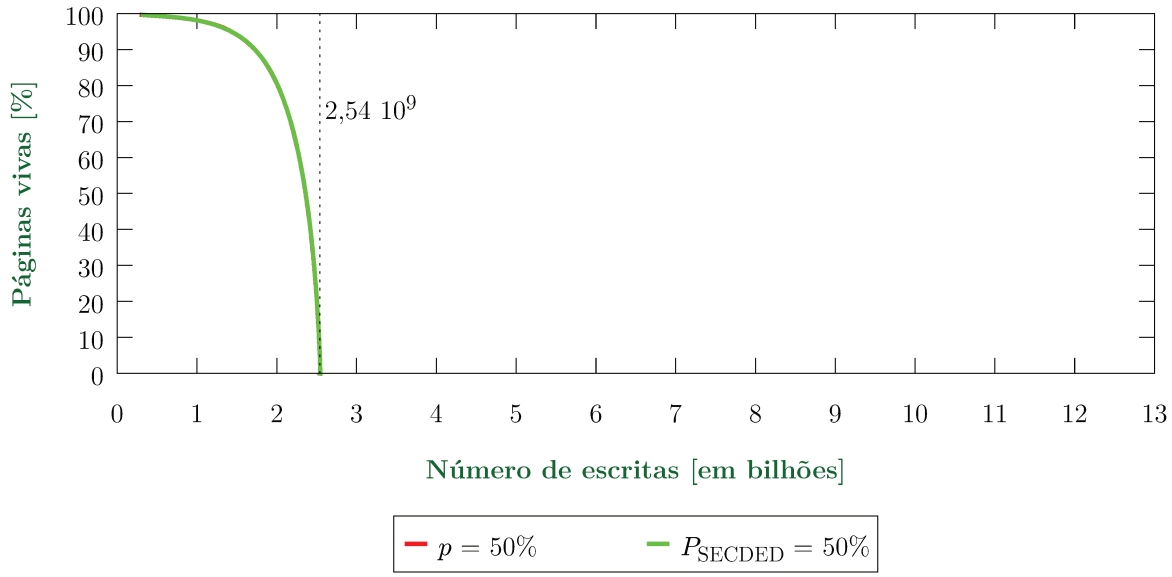


Figura 3.31: Resultados de simulações para uma memória com SECDED, sendo $p = 0,50$ e $P_{\text{SECDED}} = 0,5000$.

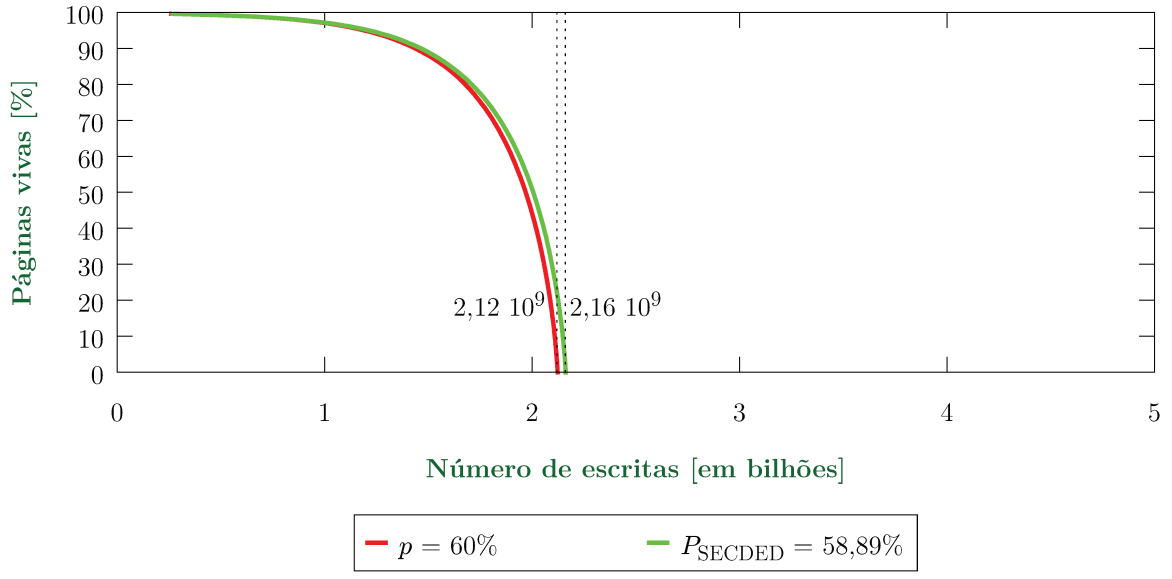


Figura 3.32: Resultados de simulações para uma memória com SECDED, sendo $p = 0,60$ e $P_{\text{SECDED}} = 0,5889$.

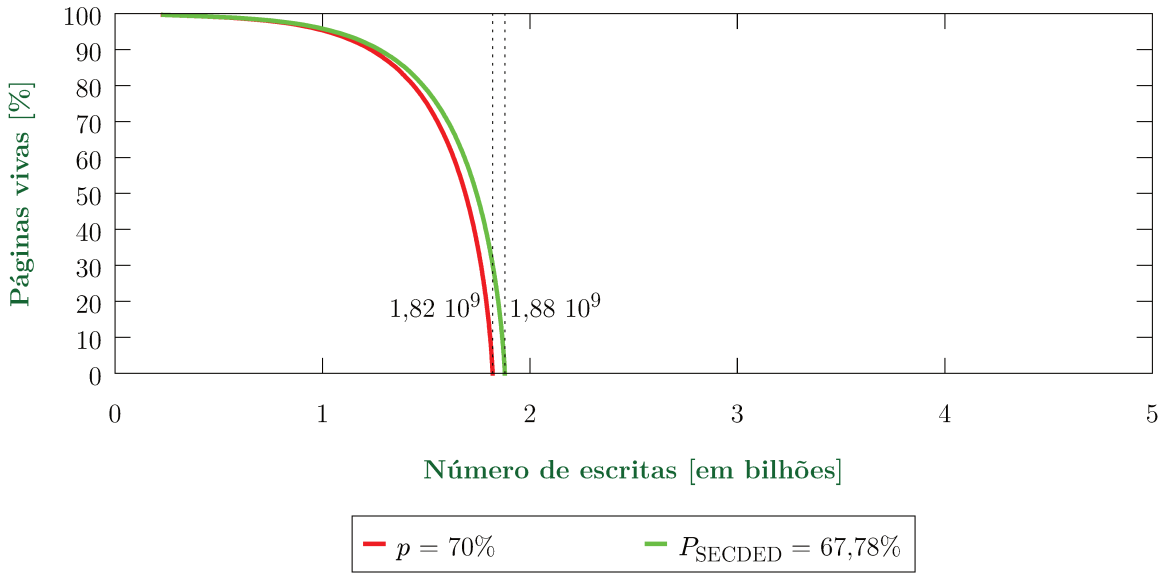


Figura 3.33: Resultados de simulações para uma memória com SECDED, sendo $p = 0,70$ e $P_{\text{SECDED}} = 0,6778$.

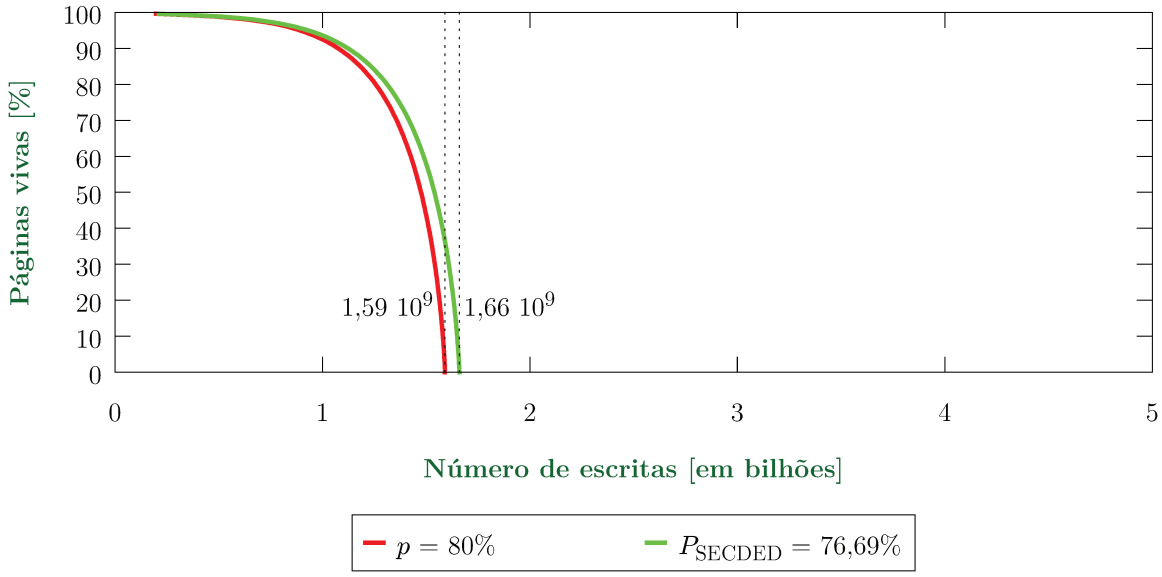


Figura 3.34: Resultados de simulações para uma memória com SECDED, sendo $p = 0,80$ e $P_{\text{SECDED}} = 0,7669$.

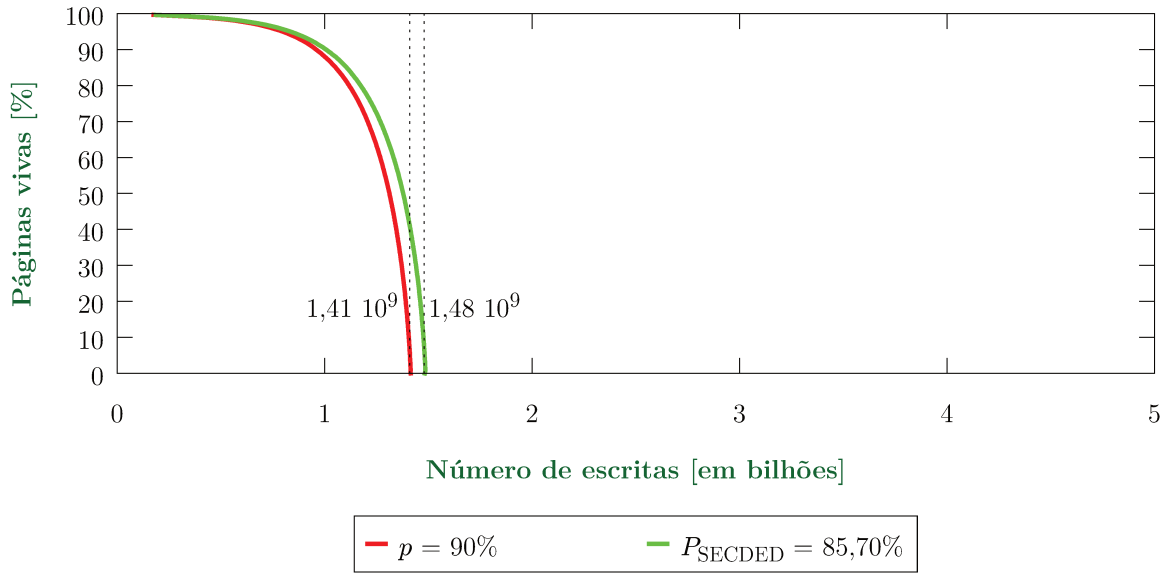


Figura 3.35: Resultados de simulações para uma memória com SECDED, sendo $p = 0,90$ e $P_{\text{SECDED}} = 0,8570$.

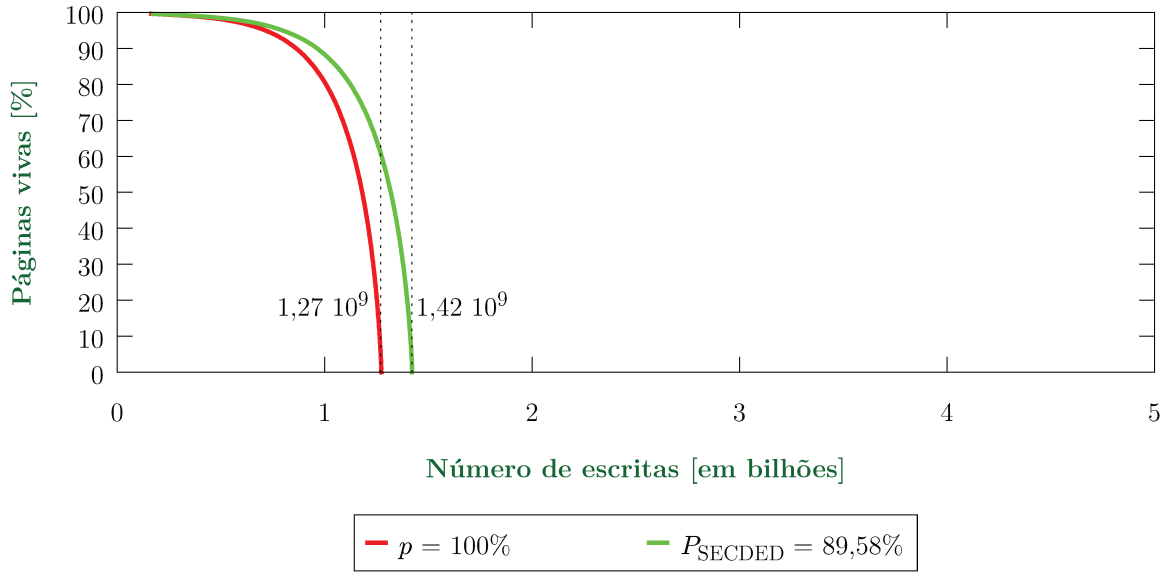


Figura 3.36: Resultados de simulações para uma memória com SECDED, sendo $p = 1,00$ e $P_{\text{SECDED}} = 0,8958$.

3.2.4 SAFER

A técnica SAFER [46] – acrônimo de *Stuck-At-Fault Error Recovery* –, assim como a técnica ECP, tem a peculiaridade de ser projetada especificamente para memórias não-voláteis. Entretanto, pode-se dizer que as semelhanças param nisso, pois enquanto a ECP é notoriamente simples, a SAFER é bem intrincada. Quando uma linha de dados está para ser escrita em uma memória com SAFER, ela poderá ter o valor de seus *bits* invertidos para que um deles case com o valor do *bit* falho na linha de memória, como ilustrado na Figura 3.37. Em um *bit* sobressalente, grava-se um quando invertida e zero, caso contrário, para indicar o estado da linha; permitindo, assim, que uma requisição de leitura seja corretamente atendida.

Embora a descrição acima não pareça um esquema complicado, no entanto, tratava-se de apenas um *bit*. Para que a linha corrija muito mais erros é necessário subdividi-la em grupos, de forma que cada um deles tenha apenas um *bit* com erro. Sob demanda, e dinamicamente, os grupos se reorganizam à medida que surgem novas falhas, isolando-as e, dessa forma, possuindo a capacidade de serem invertidos isoladamente.

Os grupos são formados combinando os endereços dos *bits* na linha; por exemplo, para uma linha de tamanho de 16 *bits* são necessários $\lceil \log_2 16 \rceil = 4$ *bits* para endereçar um *bit*. Considerando que a técnica corrige 4 falhas, são necessários 4 grupos e $\lceil \log_2 4 \rceil = 2$ *bits* para identificar um grupo. Dos 4 *bits* de endereçamento de *bits* na linha, para corrigir a falha no *bit* 14 (Figura 3.38), configura-se os grupos para serem formados pelos dois *bits* mais significativos dos *bits* de endereçamento. Ou seja, é uma escolha das $\binom{4}{2}$ possíveis formas de identificação dos grupos. Como depende da disposição das falhas na linha, podem existir mais de uma forma de identificação. A Figura 3.38 ilustra as situações desse exemplo.

Na Figura 3.38(a) o primeiro Campo de Partição indica qual é o *bit* menos significativo a formar a identificação dos grupos e o segundo campo indica o mais significativo. A palvavra que foi escrita teve de ter os *bits* pertencentes ao Grupo 3 invertidos, por isso o *bit* de inversão do Grupo 3 vale 1 (células azul-escuras). Na Figura 3.38(b), um erro recentemente ocorreu (indicado pelo bloco sólido) e a última palavra escrita tinha na posição 12 um *bit* diferindo do valor da falha de mesma posição na linha de memória. Portanto, os *bits* do Grupo 2 foram invertidos. Por conta dessa segunda falha, o primeiro Campo de Partição tem de ser fixado, fazendo o Contador de Partição ser incrementado. Esse incremento se dá de duas maneiras: existem mais do que 2^i falhas e i campos de partição tem de ser fixos, ou ocorreu duas falhas em um mesmo grupo, forçando o Campo de Partição (apontado pelo contador) ser fixado, evitando qualquer formação posterior de grupos que envolva essas falhas novamente.

Da descrição acima, tem-se k *bits* para indicar quando um grupo precisa ter seu valor

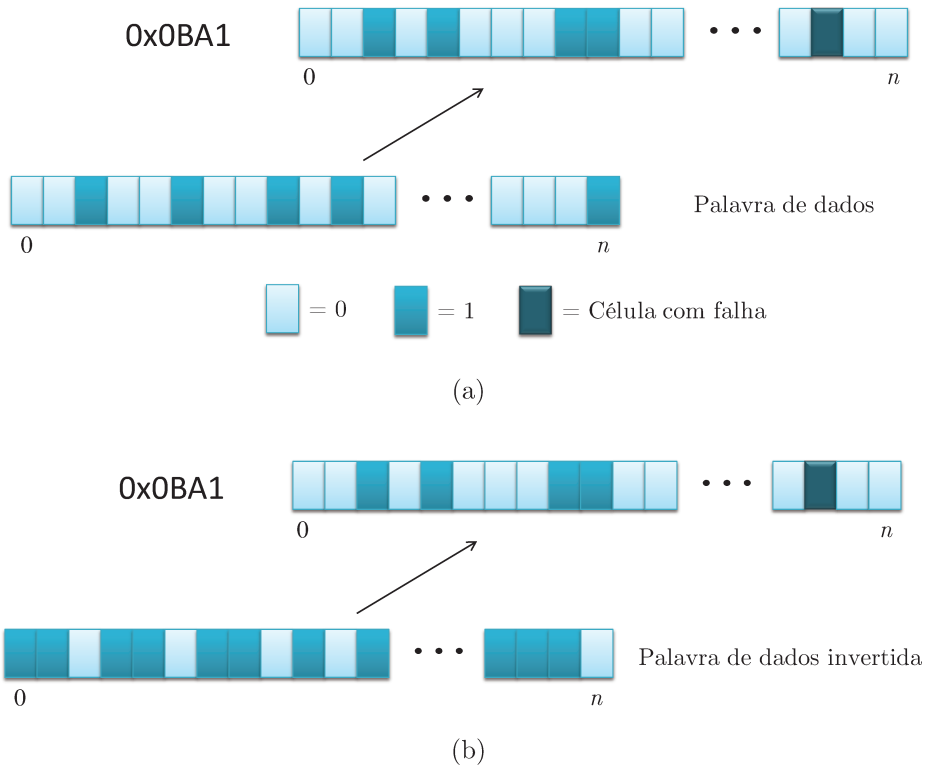


Figura 3.37: (a) O endereço 0x0BA1 possui uma falha no $(n-2)$-ésimo *bit*, o qual está fixo no valor 1. A palavra de dados a ser escrita possui o valor 0 na mesma posição que a falha se encontra. (b) A palavra de dados é invertida para que seu $(n-2)$-ésimo *bit* tenha seu valor casado com o *bit* falho no endereço 0x0BA1.

invertido para leitura da linha, $\lceil \log_2 k \rceil$ campos de partição, com $\lceil \log_2 \lceil \log_2 n \rceil \rceil$ *bits*, ou seja, o número de *bits* para indicar qual dos $\lceil \log_2 n \rceil$ *bits* de endereçamento da linha está sendo usado. Além disso, faltam os $\lceil \log_2 (\lceil \log_2 k \rceil + 1) \rceil$ *bits* para o contador de partições. Totalizando, uma linha de memória com SAFER exige $N_{\text{SAFER}} = n + k + \lceil \log_2 k \rceil \cdot \lceil \log_2 \lceil \log_2 n \rceil \rceil + \lceil \log_2 (\lceil \log_2 k \rceil + 1) \rceil$ *bits*.

Considerando uma memória cujas linhas possuem *wear-leveling* intra-linha, a probabilidade de *bit-flip* difere entre os grupos que possuem uma falha para os que não possuem. A probabilidade de inverter *bits* da linha a ser gravada depende de: se no grupo ao qual elas pertencem existe um *bit* com erro; e se o *bit* que deveria ser gravado sobre ele tem o mesmo valor ou não. Visto que um *bit* tem 50% de chance de ter o mesmo valor de uma falha, a probabilidade de que os *bits* do mesmo grupo e o *bit* de inversão sofram modificação é de 50%. Dado que os *bits* dos campos de partição e do contador de partições sofrem poucas modificações, no máximo igual ao número de erros corrigíveis – como os ponteiros em uma linha com ECP – toma-se que esses *bits* entre dois passos de simulação

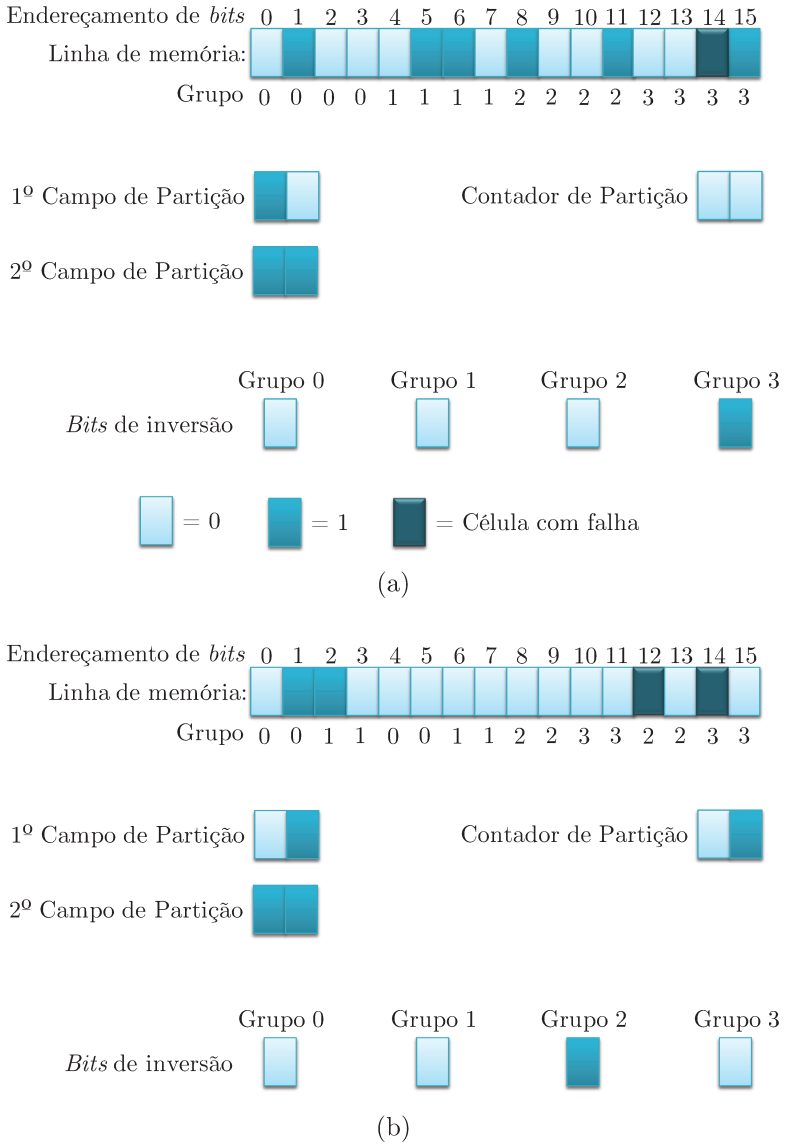


Figura 3.38: (a) Estados dos *bits* da linha de memória que possui uma falha no *bit* de posição 14. (b) Novo estado dos *bits* da linha de memória depois de uma falha na posição 12 e uma nova palavra escrita.

tem probabilidade de escrita nula. Portanto, a probabilidade de *bit-flip* de uma linha com SAFER é

$$P_{\text{SAFER}}(e) = \frac{\left(n + (k-e) - \left(\frac{n}{k} \cdot e\right)\right) \cdot p + \left(\frac{n}{k} \cdot e\right) \cdot \frac{1}{2} \cdot p + \left(\frac{n}{k} \cdot e\right) \cdot \frac{1}{2} \cdot (1-p) + e \cdot \frac{1}{2}}{N_{\text{SAFER}}} \tag{3.16}$$

Na Equação 3.16 é possível ver que a probabilidade de *bit-flip* depende do número de erros que a linha suporta. Utilizando *wear-leveling* intra-linha e supondo que a linha não tenha erros ($e = 0$), o desgaste da linha é distribuído – pela rotação – entre os n *bits* de dados mais os k *bits* que indicam a inversão de um grupo. Ao passo que os erros acontecem, os *bits* que indicam a inversão dos grupos com erros passam a ter probabilidade de 50% de terem seu valor invertido; por isso existe o último termo aditivo no numerador da equação, como também a expressão $k - e$ no primeiro termo aditivo. A fração n/k é a quantidade de *bits* em um grupo e com o aparecimento de erros o grupo deixa de ter probabilidade de inversão p, passando a ter 50% de chance de serem gravados com probabilidade p ou $1 - p$. Simplificando a Equação 3.16,

$$P_{\text{SAFER}}(e) = \frac{(n + k - e) \cdot p + \dfrac{n}{k} \cdot e \cdot \left(\dfrac{1}{2} - p\right) + e \cdot \dfrac{1}{2}}{N_{\text{SAFER}}} \tag{3.17}$$

Percebe-se que é impossível adaptar a Equação 3.6 para uma simulação de uma memória com SAFER. Primeiro, porque cada linha tem uma quantidade de erros, segundo, porque esses valores mudam a cada passo da simulação. Para resolver o primeiro problema, calcula-se a média aritmética do número de erros nas linhas a cada passo da simulação, denotado-a por $e_\mu(k)$. Já o segundo problema, é solucionado introduzindo essa média à equação 3.4, uma vez que sua contribuição no cálculo do total de escritas é estritamente pontual:

$$\mathbb{W}(k) = \sum_{i=1}^{k} \frac{c_i \cdot f_{wr}(i)}{P_{\text{SAFER}}(e_\mu(i))} \tag{3.18}$$

A configuração para a técnica SAFER mais comum é a SAFER_{32} a qual mantém, assim como todas citadas até agora, o *overhead* até o limite de 12,5%. Na Tabela 3.5 é mostrado para essa configuração o comportamento da probabilidade de *bit-flip* ajustada e não ajustada. Os valores mínimos são obtidos no inicio da simulação, quando a quantidade de erros é zero, devido à suscetibilidade ao número de erros, a probabilidade de *bit-flip* aumenta até atingir uma valor máximo logo antes de falhar – isso vale para probabilidades abaixo de 50%. Esse valor, como também a probabilidade de *bit-flips* média ($\mu(P_{\text{SAFER}})$), depende do número de erros médio para uma linha de memória, que é diferente a cada simulação executada. Desse modo, o valor visualizado na tabela é a média das máximas probabilidades atingidas na execução de 1250 simulações com $N = n = 512$, $\Delta = k = 32$, $N_L = 64$, $N_P = 256$, $\mu = 10^8$ e $\sigma = 2,5 \cdot 10^7$.

Assim como nas técnicas DRM e SECDED, observa-se que quanto menor é a probabilidade p, mais suscetível a um aumento a probabilidade ajustada está, como é possível visualizar para $p = 10\%$. Todavia, diferentemente dessas técnicas, a influência da alta

taxa de *flips* dos *bits* de verificação não é constante e influencia os *bits* de dados (devido ao mecanismo de inversão), por isso aparece valores máximos de probabilidade muito maiores do que os resultados nas Tabelas 3.2 e 3.4. Dessa forma, os valores de p bem abaixo de 50% têm tendência a aumentarem com o aparecimento de falhas e ao passo que se aproximam de 50%, o impacto do esquema de inversão é gradualmente absorvido pelos *bits* adicionais – como os contadores de partição – que não são utilizados a todo momento. Consequentemente, nas Figuras 3.41 à 3.43 o ajuste representa uma melhora na duração da memória, contrário aos resultados das Figura 3.39, nas quais o ajuste mostra redução, e da Figura 3.40, na qual praticamente não diferem. Como a ECP, o maior taxa de erro apresentado na média do número de escritas foi 0,2%, mostrando que, a despeito de sua complexidade, é uma técnica bem estável e que variações no processo afetam pouco o resultado médio.

Na Tabela 3.5, note também que, acima dos valores de probabilidade nominal de 50%, a maior e a menor probabilidade de *bit-flip* são as mesmas. Isso ocorre por conta do aparecimento de falhas, uma vez que elas implicam numa taxa de *flip* de 50% para os grupos de *bits* que as contêm. Portanto, à medida que o número de falhas aumenta, há uma predominância da taxa de inversão de 50% para os *bits* das linhas. Este valor que é menor do que p nominal, nesses casos, acaba por reduzir as probabilidades ajustadas, como ocorrido nas demais técnicas.

As Figuras 3.44 à 3.48 resultam da mesma configuração de simulação supracitada. Igualmente às outras técnicas, as probabilidades ajustadas apresentam um melhor tempo de vida para probabilidades acima de 50%, porém, essas altas probabilidades de *bit-flip* não são desejadas, nem esperadas, pelos motivos já discorridos nas Seções 3.2.2 e 3.2.3.

Tabela 3.5: Comparação entre valores de probabilidade de *bit-flip* com e sem ajuste para SAFER_{32}. Dado $n = 512$ e $k = 32$. Na terceira, na quinta e na sétima colunas denota-se, ou a redução, ou o aumento porcentual da probabilidade ajustada em relação à p.

p	$\min(P_{\text{SAFER}_{32}})$	%	$\mu(P_{\text{SAFER}_{32}})$	%	$\max(P_{\text{SAFER}_{32}})$	%
10%	9,59%	4,10% ↓	12,62%	26,20% ↑	16,92%	69,20% ↑
20%	19,19%	4,05% ↓	21,46%	7,30% ↑	24,69%	23,45% ↑
30%	28,78%	4,07% ↓	30,30%	1,00% ↑	32,45%	8,17% ↑
40%	38,38%	4,05% ↓	39,13%	2,17% ↓	40,21%	0,53% ↑
50%	47,97%	4,06% ↓	47,97%	4,06% ↓	47,97%	4,06% ↓
60%	57,57%	4,05% ↓	56,81%	5,32% ↓	57,57%	4,05% ↓
70%	67,16%	4,06% ↓	65,65%	6,21% ↓	67,16%	4,06% ↓
80%	76,75%	4,06% ↓	74,49%	6,89% ↓	76,75%	4,06% ↓
90%	86,35%	4,06% ↓	83,32%	7,42% ↓	86,35%	4,06% ↓
100%	95,94%	4,06% ↓	92,16%	7,84% ↓	95,94%	4,06% ↓

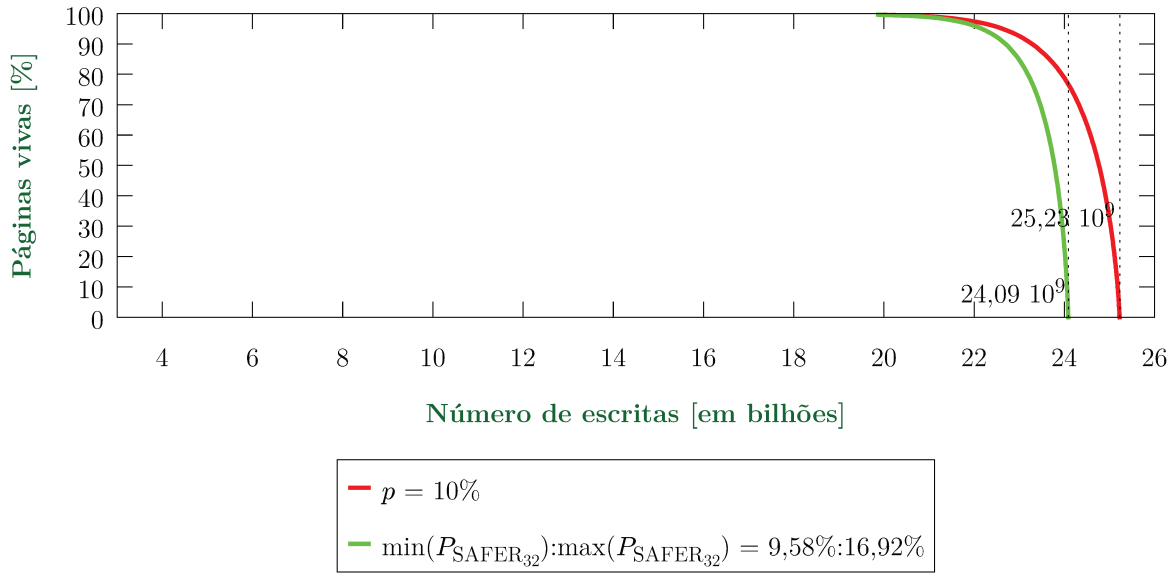


Figura 3.39: Resultados de simulações para uma memória com SAFER_{32}, sendo $p = 0,10$, $\min(P_{\text{SAFER}_{32}}) = 0,09594$ e $\max(P_{\text{SAFER}_{32}}) = 0,1692$.

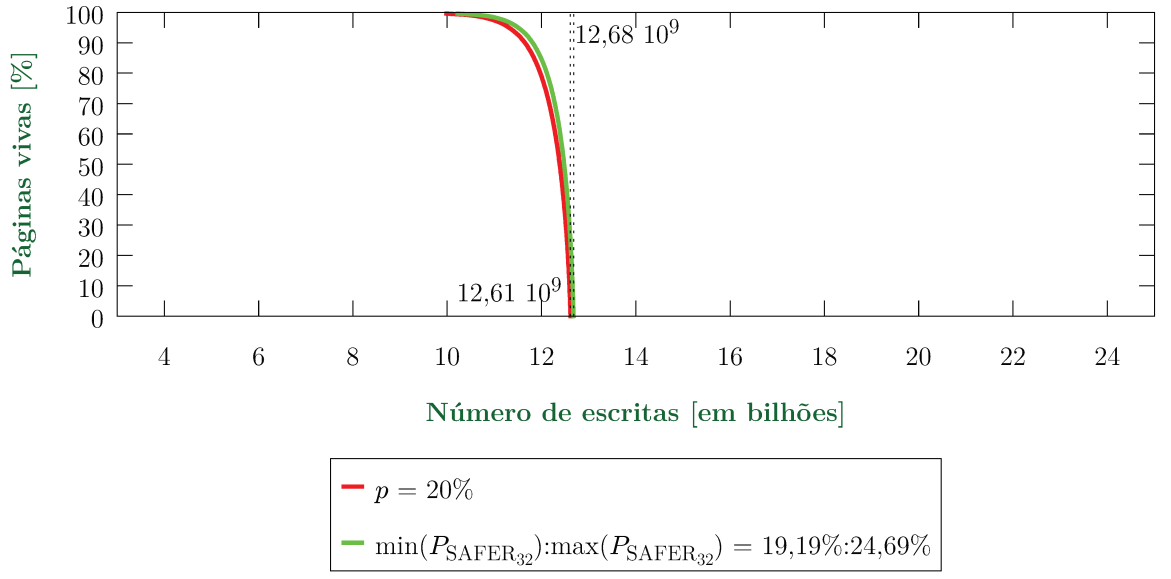


Figura 3.40: Resultados de simulações para uma memória com SAFER_{32}, sendo $p = 0,20$, $\min(P_{\text{SAFER}_{32}}) = 0,1919$ e $\max(P_{\text{SAFER}_{32}}) = 0,2469$.

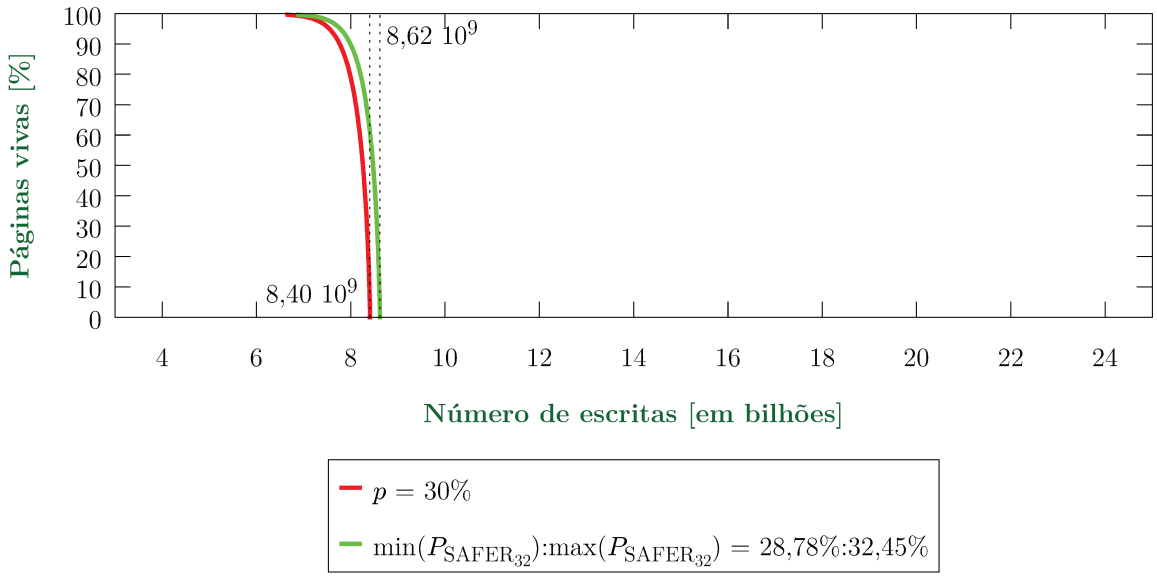


Figura 3.41: Resultados de simulações para uma memória com SAFER_{32}, sendo $p = 0,20$, $\min(P_{\text{SAFER}_{32}}) = 0,2878$ e $\max(P_{\text{SAFER}_{32}}) = 0,3245$.

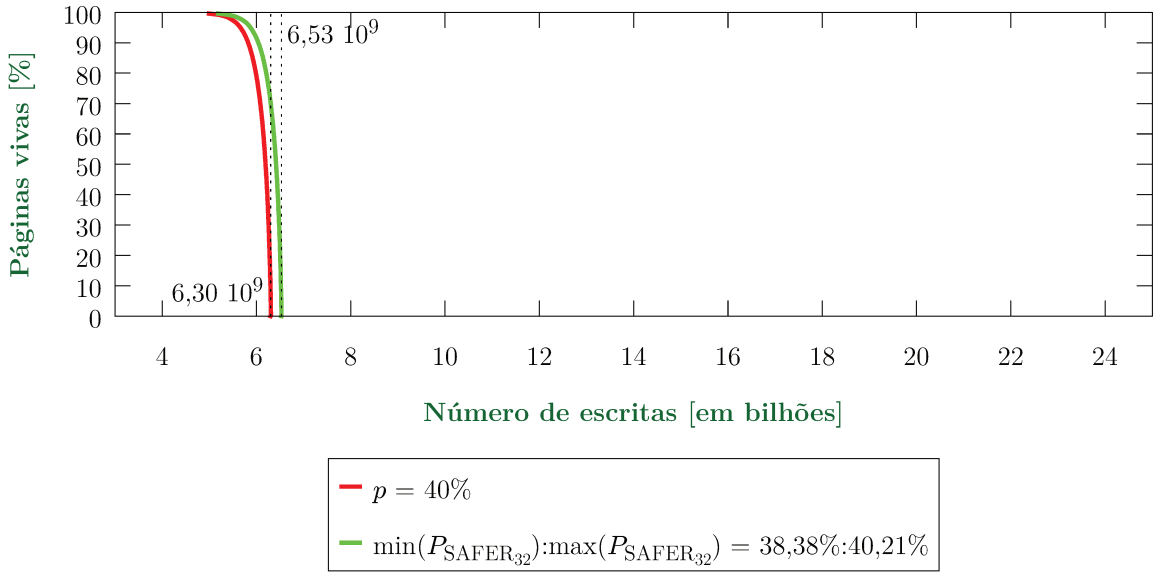


Figura 3.42: Resultados de simulações para uma memória com SAFER_{32}, sendo $p = 0,40$, $\min(P_{\text{SAFER}_{32}}) = 0,3838$ e $\max(P_{\text{SAFER}_{32}}) = 0,4021$.

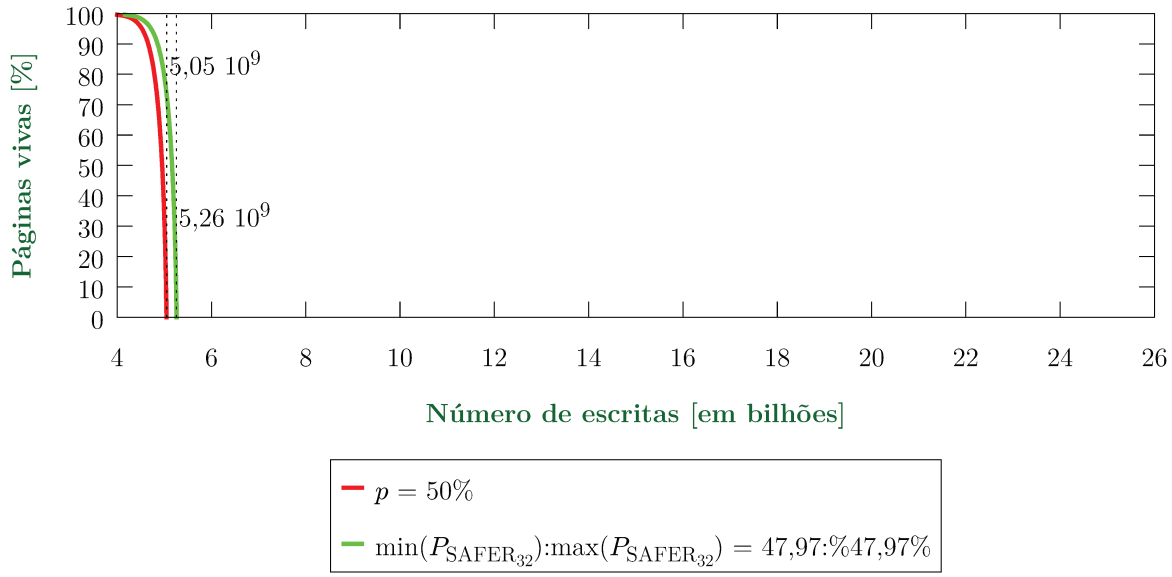


Figura 3.43: Resultados de simulações para uma memória com SAFER_{32}, sendo $p = 0,50$, $\min(P_{\text{SAFER}_{32}}) = 0,4797$ e $\max(P_{\text{SAFER}_{32}}) = 0,4797$.

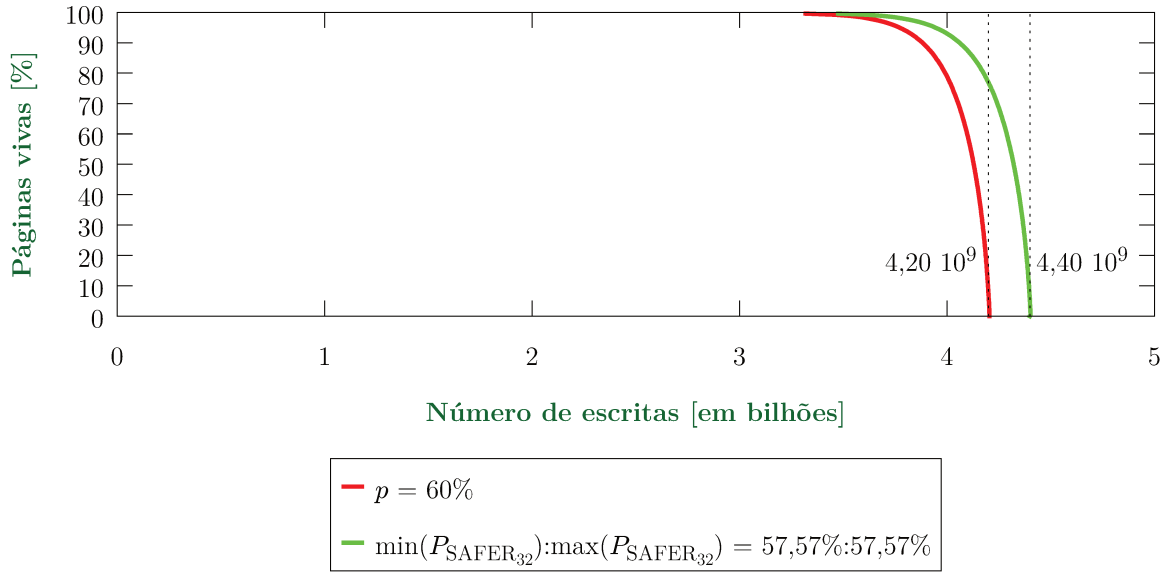


Figura 3.44: Resultados de simulações para uma memória com SAFER_{32}, sendo $p = 0,60$, $\min(P_{\text{SAFER}_{32}}) = 0,5757$ e $\max(P_{\text{SAFER}_{32}}) = 0,5757$.

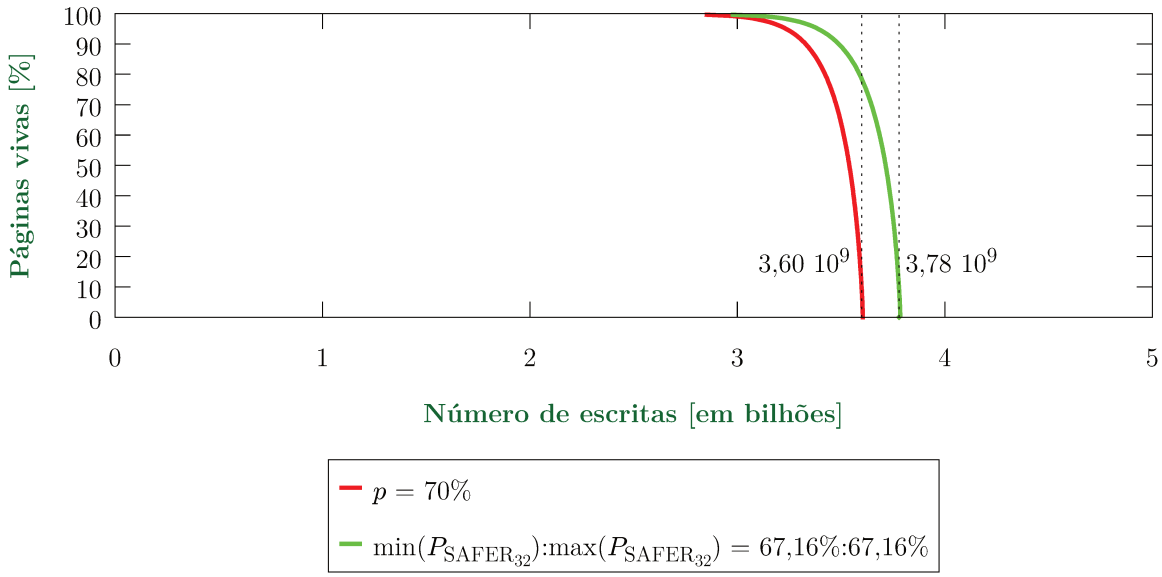


Figura 3.45: Resultados de simulações para uma memória com SAFER_{32}, sendo $p = 0,70$, $\min(P_{\text{SAFER}_{32}}) = 0,6716$ e $\max(P_{\text{SAFER}_{32}}) = 0,6716$.

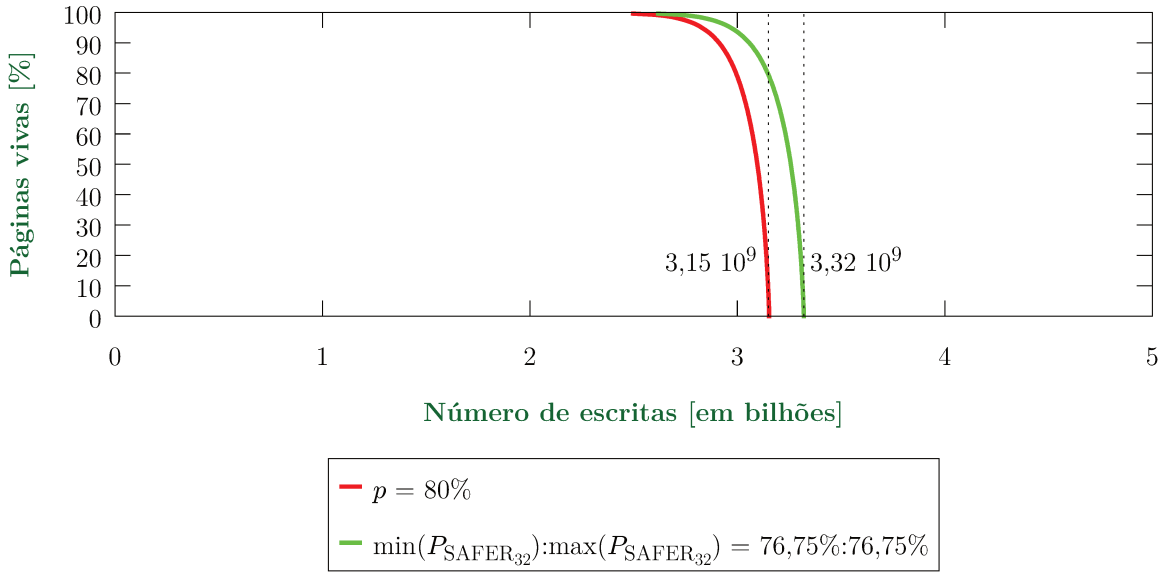


Figura 3.46: Resultados de simulações para uma memória com SAFER_{32}, sendo $p = 0,80$, $\min(P_{\text{SAFER}_{32}}) = 0,7675$ e $\max(P_{\text{SAFER}_{32}}) = 0,7675$.

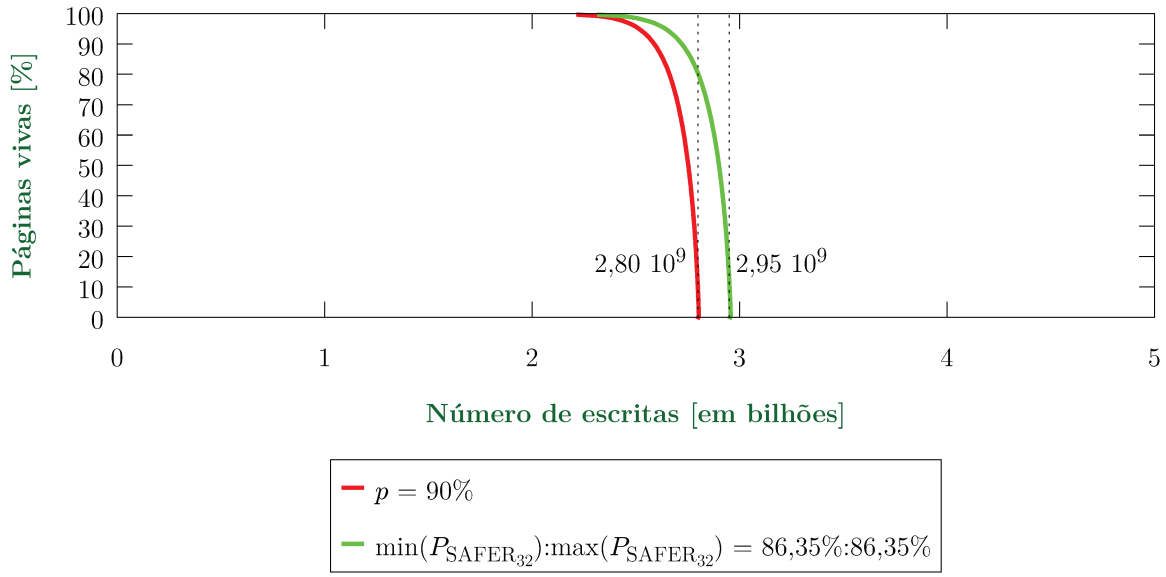


Figura 3.47: Resultados de simulações para uma memória com SAFER_{32}, sendo $p = 0,90$, $\min(P_{\text{SAFER}_{32}}) = 0,8635$ e $\max(P_{\text{SAFER}_{32}}) = 0,8635$.

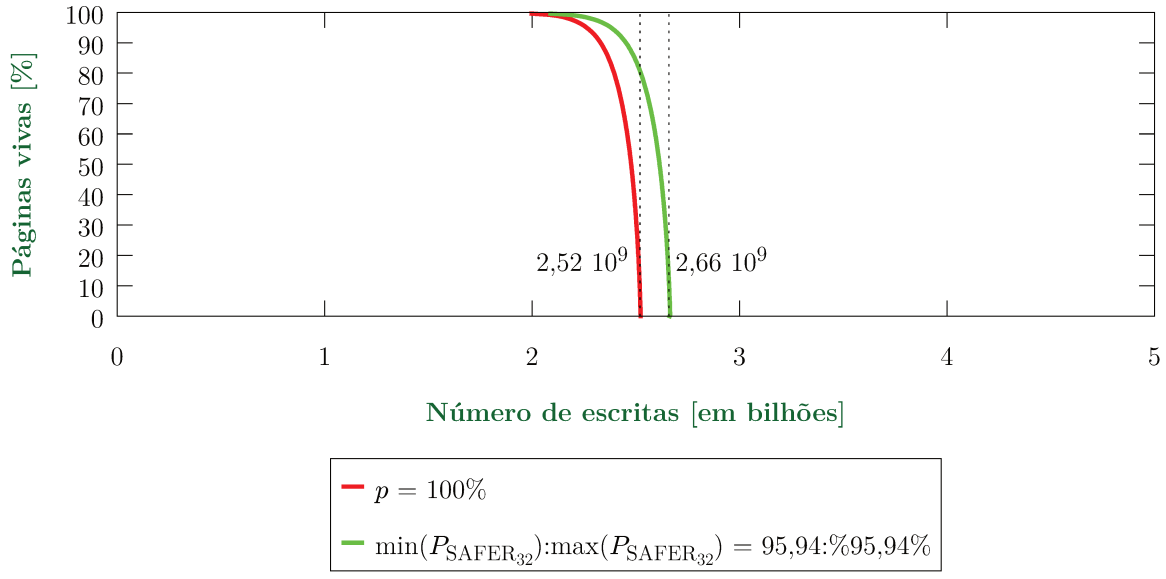


Figura 3.48: Resultados de simulações para uma memória com SAFER_{32}, sendo $p = 1,00$, $\min(P_{\text{SAFER}_{32}}) = 0,9594$ e $\max(P_{\text{SAFER}_{32}}) = 0,9594$.

3.2.5 FREE-p

FREE-p [60] é uma técnica de implementação ortogonal[10] às demais técnicas de correção de erros apresentadas neste capítulo, à exceção da DRM. Sua principal característica está em reaproveitar as linhas falhas das páginas como ponteiros para linhas livres de erros em outras páginas. Ao transformar uma linha com falhas em um ponteiro, a página que a possui não é descartada, tendo seu funcionamento mantido normalmente. As páginas livres de erros são gerenciadas pelo sistema operacional e liberadas sob demanda para essa função de reposição de linhas.

As linhas de memória com FREE-p podem usar qualquer forma de correção de erros, como ECP, paridade, SECDED, SAFER, etc, pois o reaproveitamento das linhas, só acontecerá depois que qualquer uma dessas técnicas não puder corrigir mais falhas na linha. Apesar disso, o código de correção de erros usado no trabalho é o BCH[11]. Os códigos BCH são uma generalização dos códigos de Hamming para correção de múltiplos erros [31]. Define-se o código BCH por

Tamanho do código:	$n = 2^m - 1$
Número de *bits* de informação:	k
Número de *bits* de verificação:	$n - k \leq m \cdot t$
Descrito por BCH(n,k)	
$m \in \mathbb{N} \wedge m \geq 3$	
t é o número de erros recuperáveis	$t \in \mathbb{N} \wedge t < 2^{m-1}$

Analogamente ao que foi discorrido na Subseção 3.2.3, o código BCH pode ser usado em um formato irregular (fora da definição). Por exemplo, o código usado na técnica FREE-p é o BCH(572,512), que não é encontrado na tabela de códigos em [31]. Novamente, é uma apropriação de notação e o código do qual se origina o BCH(572,512) é, na verdade, o código BCH(1023,963), onde $m = 10$ já que $1023 = 2^{10} - 1$ e $t = 6$. Logo, são 963-512=451 *bits* que devem preenchidos com a operação de *padding* para que seja possível decodificar a palavra-código.

Além dos 60 *bits* de verificação no BCH(572,512), a técnica FREE-p acrescenta 1 *bit* de paridade, de forma que o código usado é o 6EC-7ED: *Six Error Correction and Seven Error Detection*. Apesar de ser possível corrigir até seis erros em uma linha, apenas quatro são tolerados antes de se transferir o conteúdo da linha e transformá-la em um ponteiro, porquanto a técnica procura eliminar também os *soft-errors*, que embora inexistentes em PCM com SLC (veja o Capítulo 2), podem ocorrer em canais de comunicação, como barramento. Dessa forma, até dois possíveis *soft-errors* podem ocorrer durante a transferência que serão recuperados.

[10] Ou seja, sua utilização não impede o uso em conjunto de certas técnicas de correção de erros.

[11] Abreviação dos nomes dos autores: Bose, Chaudhuri and Hocquenghem.

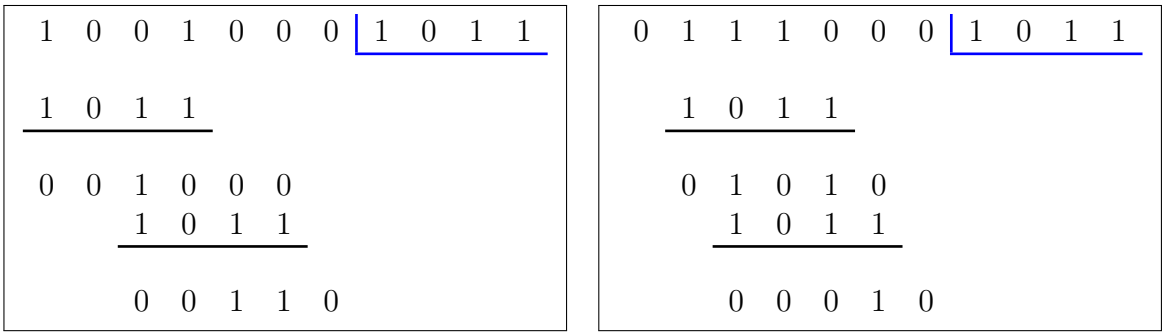


Figura 3.49: Cálculo de palavra de verificação para código BCH(7,4): (a) $n = 7$, $k = 3$, $d(x) = x^3 + 1 = 1001$, $g(x) = x^3 + x + 1 = 1011$. O código de verificação gerado foi $v(x) = x^2 + x = 110$, portanto, a palavra código é dada por $c(x) = x^6 + x^3 + x^2 + x = 1001110$. (b) $n = 7$, $k = 3$, $d(x) = x^2 + x + 1 = 0111$, $g(x) = x^3 + x + 1 = 1011$. O código de verificação gerado foi $v(x) = x = 010$, portanto, a palavra código é dada por $c(x) = x^5 + x^4 + x^3 + x = 0111010$.

As operações no código BCH são baseadas na matemática de corpos finitos. A despeito da beleza dos conceitos envolvidos, apenas detalhes indispensáveis para a determinação da probabilidade de *bit-flip* dos *bits* de verificação serão explicados. A palavra-código é definida como um polinômio, em $GF(2)$[12], $c(x) = d(x) \cdot x^{n-k} + v(x)$, onde $c(x)$ é um polinômio de grau $n-1$, $d(x)$ é o polinômio de grau $k-1$ representado a informação e $v(x)$ é um polinômio de grau $l-1$, onde $l = m \cdot t = n - k$, representando o código de verificação.

O processo de codificação é dado pela divisão do polinômio $d(x) \cdot x^l$ pelo polinômio[13] $g(x)$, denominado polinômio gerador [52,54] de grau l. O processo de divisão em $GF(2)$ é parecido com o processo de divisão binária feito usando subtrações e deslocamentos, todavia, em GF(2), a operação de substração é equivalente à adição, tornando possível realizar as subtrações através de soma-módulo-2, isto é, através da operação lógica ou-exclusivo. Na Figura 3.49 são ilustrados dois processos de obtenção da palavra de verificação para o código BCH(7,4).

Seja $G = \{g_0, g_1, \ldots, g_l\}$ um conjunto de $l+1$ *bits* para a representação binária de um polinômio gerador de grau l. Seja $D = \{d_0, d_1, \ldots, d_{k-1}\}$ uma palavra binária de k *bits* representando a informação e seja $V = \{v_0, v_1, \ldots, v_{l-1}\}$ o conjunto de l *bits* representando o resto da divisão de D por G. Tomando o código BCH(7,4) genérico da Figura 3.50 como exemplo, a execução da divisão de D por G se dá pela soma-módulo-2 entre uma

[12] $GF(2)$ significa *Galois Fields* (Campos de Galois) para dois elemento e é o menor corpo finito possível.
[13] Note que x^l é no fundo uma operação de deslocamento binária para que os *bits* representados pelo polinômio $v(x)$ sejam adicionados a palavra-código sem modificação dos dados.

	d_3	d_2	d_1	d_0	0	0	0	g_3	g_2	g_1	g_0
$\mathfrak{L}_1$:	g_3	g_2	g_1	g_0							
$\mathfrak{L}_2$:		g_3	g_2	g_1	g_0						
$\mathfrak{L}_3$:			g_3	g_2	g_1	g_0					
$\mathfrak{L}_4$:				g_3	g_2	g_1	g_0				
					v_2	v_1	v_0				

Figura 3.50: Cálculo de palavra de verificação genérica para BCH(7,4).

das combinações das linhas $\mathfrak{L}_1$, $\mathfrak{L}_2$, $\mathfrak{L}_3$ e $\mathfrak{L}_4$. Note que ausência de todas as linhas também é uma combinação.

Pensando em forma de conjuntos, todas as possíveis combinações da operação de divisão, mostradas na Figura 3.50, podem ser representadas pelo conjunto potência $\mathcal{P}(L)$ do conjunto $L = \{\mathfrak{L}_1, \mathfrak{L}_2, \mathfrak{L}_3, \mathfrak{L}_4\}$. Sabe-se que os *bits* de V resultarão de uma combinação dos *bits* de G, e essa combinação pode ser representada por um elemento do conjunto $\mathcal{P}(L)$ sob a aplicação $f_\oplus$ (veja a Definição B.3), isto é, um elemento de $\mathcal{L} = \bigcup_{P\in\mathcal{P}(L)} f_\oplus(P) = \{0, \mathfrak{L}_1, \mathfrak{L}_2, \mathfrak{L}_3, \mathfrak{L}_4, \mathfrak{L}_1 \oplus \mathfrak{L}_2, \mathfrak{L}_1 \oplus \mathfrak{L}_3, \ldots, \mathfrak{L}_1 \oplus \mathfrak{L}_2 \oplus \mathfrak{L}_3 \oplus \mathfrak{L}_4\}$.

Suponha que a divisão seja representada por $\mathfrak{L}_2 \oplus \mathfrak{L}_3$; isso implica que v_2 é resultado de $g_0 \oplus g_1$, v_1 de g_0 e v_0 de 0. Sabe-se que todos possíveis resultados de v_2 são $\{0, g_0, g_1, g_2, g_0 \oplus g_1, g_0 \oplus g_2, \ldots, g_0 \oplus g_1 \oplus g_2\}$. Porém, observa-se que $|\mathcal{L}| = 16$ e o número de possíveis resultados para v_2 é 8. Isso acontece porque a linha $\mathfrak{L}_1$ não contém *bit* algum que afete v_2; como cada combinação que modifica v_2 acontece uma vez combinada com o *bit* de $\mathfrak{L}_1$ (*i.e.*, $\emptyset$) e outra não, cada combinação aparece duas vezes: tem duas chances de acontecer em dezesseis. Similarmente, v_1 terá seu valor calculado a partir de uma das combinações $\{0, g_0, g_1, g_0 \oplus g_1\}$, o que implica que elas se repetem quatro vezes. Deduz-se, então, que para v_0 cada uma das possíveis combinações $\{0, g_0\}$ terá oito chances em dezesseis de ocorrer.

É importante dizer que, por construção, todo polinômio gerador no código BCH tem seus *bits* mais e menos significativos valendo 1; sendo que o *bit* mais significativo não influência os valores de v_2, v_1, ou v_0. Pela Proposição B.1 e pelo Corolário B.1.1, todo conjunto de *bits* com pelo menos um *bit* 1 terá metade das operações $f_\oplus$ nos elementos de seu conjunto potência resultando em 1 e metade resultando em 0. Em outra palavras, é possível categoricamente afirmar que, independentemente do polinômio gerador, qualquer *bit* de verificação terá metade de suas possíveis combinações resultando em 1 e a outra metade em 0.

Com isso, suponha que em um endereço de memória se encontram gravados uma palavra de dados D e seu código de verificação V, e que uma nova palavra de dados D' será gravada nesse endereço. A probabilidade de não se modificar os *bits* de verificação é

a probabilidade de ocorrer $d_3 = d'_3$, $d_2 = d'_2$, $d_1 = d'_1$ e $d_0 = d'_0$, ou seja, a probabilidade de não haver alteração nos *bits* de dados. Como cada *bit* de dados tem probabilidade de inversão p, a probabilidade do código de verificação não ser alterado é $(1-p)^4$.

Se, por ventura, algum *bit* de D' for diferente de D – o que tem probabilidade de $1-(1-p)^4$ de ocorrer – o valor de v_2, por exemplo, será uma entre estas possibilidades $\{0, g_0, g_1, g_2, g_0 \oplus g_1, \ldots, g_0 \oplus g_1 \oplus g_2\}$, contudo, uma deles já é o valor atual de v_2 e está exclusa da probabilidade $1-(1-p)^4$, portanto, existe 8/15 de chance de acontecer uma das combinações que alterará o valor de v_2 e 7/15 disso não acontecer. Lembrando que o número total de combinações possíveis nessa operação de divisão é 16, que para v'_2 cada operação se repete duas vezes, somando-se a isso, que a combinação pela qual v_2 resulta não está inclusa na probabilidade de ser escolhida, sobram 15 possíveis combinações. Desse modo, v'_2 tem probabilidade de *bit-flip* de $(1-(1-p)^4) \cdot \frac{8}{15}$.

De maneira geral, a substituição de uma palavra de dados $D = \{d_0, d_1, d_2, \ldots, d_{k-1}\}$ e seu código de verificação $V = \{v_0, v_1, \ldots, v_{l-1}\}$, escritos em um endereço de memória, por uma palavra de D' com seu código de verificação V', implica na modificação dos *bits* do código de verificação se houver possibilidade dos *bits* de dados sofrerem modificações, o que ocorre probabilidade $1-(1-p)^k$. Sendo que a probabilidade de ocorrer uma combinação de operação ou-exclusivo, dos *bits* do polinômio gerador, que resulte em um valor que modificaria algum *bit* de verificação é igual a probabilidade de escolher algum elemento, de 2^{k-1} elementos iguais, dentro de um conjunto com $2^k - 1$ elementos no total, então, a probabilidade de *bit-flip* para um *bit* de verificação é

$$P_{bit \text{ de verificação}} = (1-(1-p)^k) \cdot \frac{2^{k-1}}{2^k - 1} \tag{3.19}$$

Para uma linha de dados codificada usando um código BCH(n,k), considerando *wear-leveling* intra-linha, a probabilidade de *bit-flip* dá-se por

$$P_{\text{BCH}} = \frac{1}{n}\left(k \cdot p + (n-k) \cdot P_{bit \text{ de verificação}}\right) \tag{3.20}$$

Finalmente, para a técnica FREE-p existe 1 *bit* de paridade sobre todo o código BCH, portanto

$$P_{\text{FREE-p}} = \frac{1}{n+1}\left(P_{\text{PARIDADE}}(n, P_{\text{BCH}}) + n \cdot P_{\text{BCH}}\right) \tag{3.21}$$

Como na técnica FREE-p usa-se o código BCH(572,512), $k = 512$, a Equação 3.19 pode ser simplificada evitando-se computação desnecessária, logo

$$P_{bit \text{ de verificação}} = \left(\frac{1-(1-p)^k}{2}\right) \cdot \frac{2^k}{2^k - 1}\bigg|_{k=512} \approxeq 1 = \frac{1-(1-p)^k}{2} \tag{3.22}$$

Conquanto $(1-p)^k$ para $0 < p \leq 1$ tenda a 0 quando $k \to \infty$, para $k = 512$ alguns valores de p bem pequenos, mais ainda possíveis, fazem $(1-p)^{512}$ resultar em valores ainda significantes, impedindo desse termo ser simplificado. Portanto, o total de escritas para cada passo da simulação é dado por:

$$\mathbb{W}(k) = \frac{W(k)}{P_{\text{FREE-p}}} \tag{3.23}$$

Na Tabela 3.6 é mostrado os valores da probabilidade de *bit-flip* ajustada e não ajustada para a técnica FREE-p. A probabilidade $P_{bit \text{ de verificação}}$ está ocultada porque, para todos os valores p, ela resultou em 0,5. É interessante notar que todos os códigos que usam a operação ou-exclusivo para calcular seus *bits* de verificação, a despeito das diferentes formas de se fazer, resultam em probabilidade de *bit-flip* muito próxima de 50%, praticamente de forma independente da probabilidade p; o que de certa forma prejudica o desempenho dessas técnicas se p for menor do que 0,5.

Com 1250 simulações – logrando erro padrão máximo menor do que 0,3% para a média dos números de escritas – para uma PCM configurada com $\Delta = n - k = 60$, tendo os outros parâmetros com os valores de praxe usados neste capítulo, foram obtidos os resultados mostrados nos gráficos das Figuras 3.51 à 3.60, que se assemelham aos gráficos referentes as simulações das técnicas DRM e SECDED, onde é notória a degradação do tempo de vida para probabilidade de *bit-flip* pequenas (menores do que 50%).

Tabela 3.6: Comparação entre valores de probabilidade de *bit-flip* com e sem ajuste para FREE-p. Dado $n = 512$ e $n - k = 60$. A terceira coluna é a redução (ou aumento) porcentual da probabilidade ajustada.

p	$P_{\text{FREE-p}}$	%
10%	14,26%	42,60% ↑
20%	23,19%	15,95% ↑
30%	32,13%	7,10% ↑
40%	41,06%	2,65% ↑
50%	50,00%	0,00% –
60%	58,94%	1,77% ↓
70%	67,87%	3,04% ↓
80%	76,81%	4,00% ↓
90%	85,74%	4,73% ↓
100%	94,68%	5,32% ↓

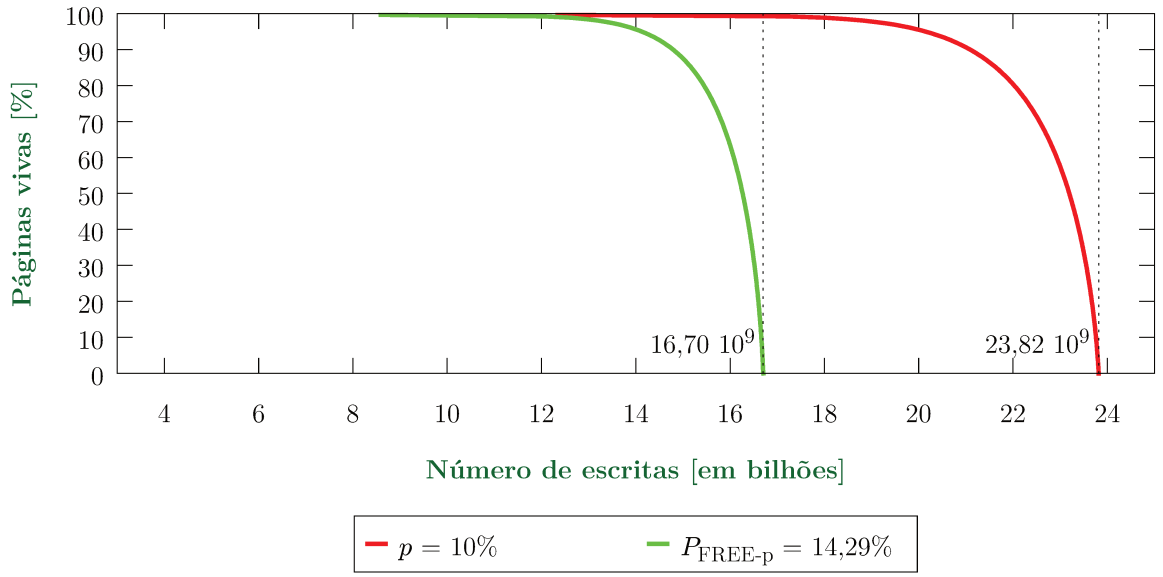


Figura 3.51: Resultados de simulações para uma memória com FREE-p, sendo $p = 0,10$ e $P_{\text{FREE-p}} = 0,0904$.

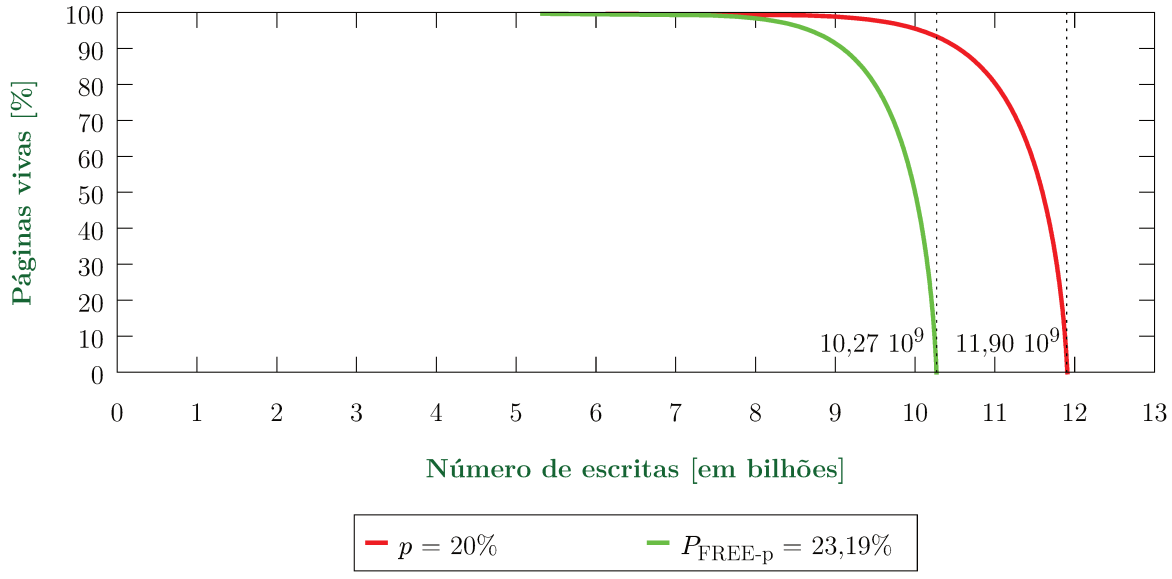


Figura 3.52: Resultados de simulações para uma memória com FREE-p, sendo $p = 0,20$ e $P_{\text{FREE-p}} = 0,1808$.

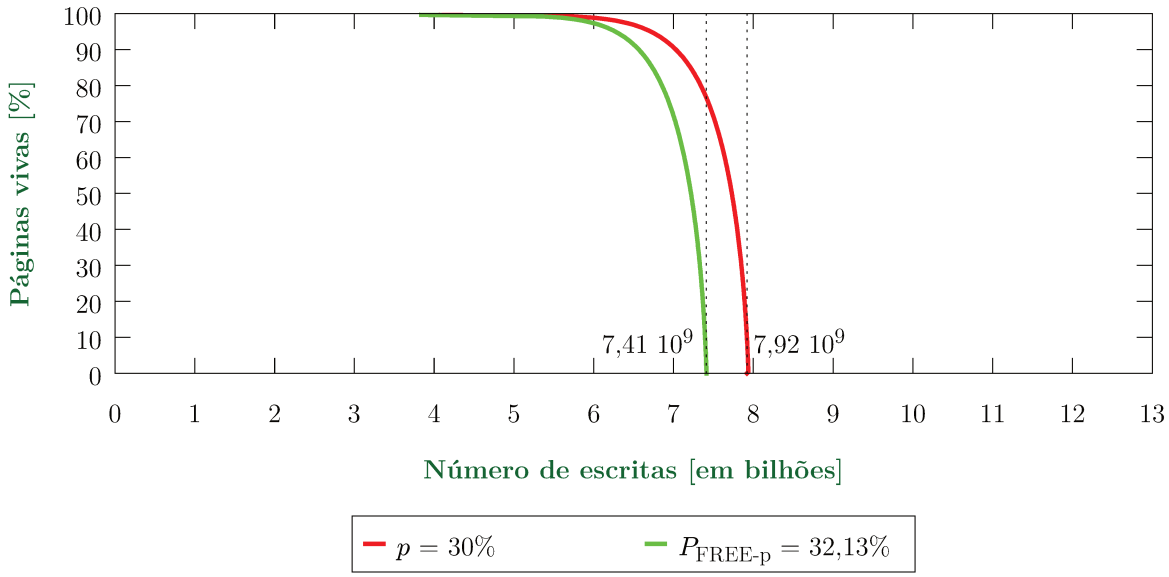


Figura 3.53: Resultados de simulações para uma memória com FREE-p, sendo $p = 0,30$ e $P_{\text{FREE-p}} = 0,2712$.

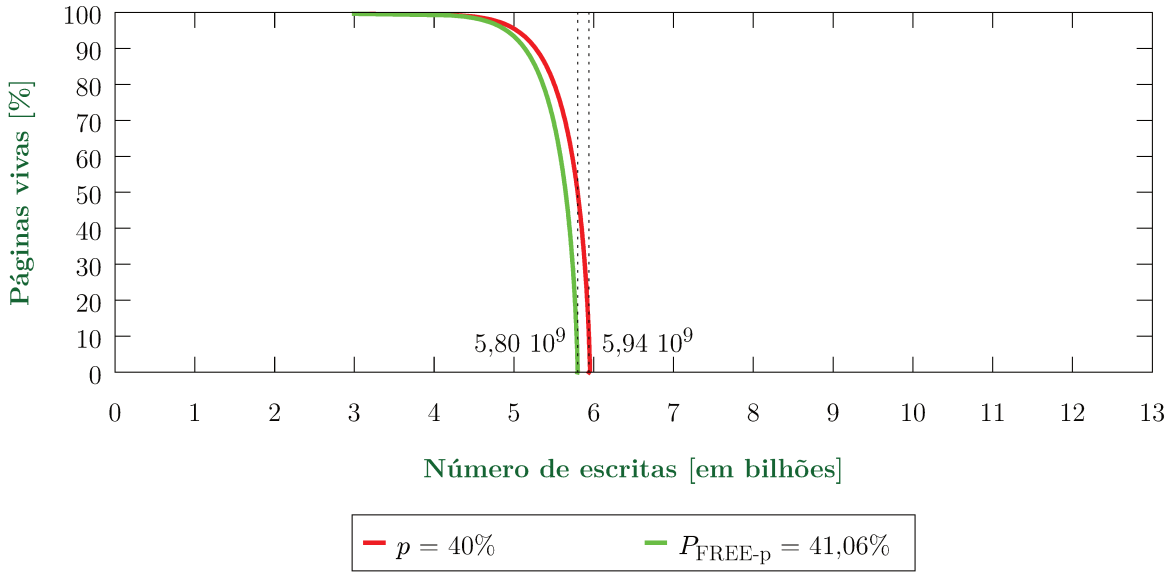


Figura 3.54: Resultados de simulações para uma memória com FREE-p, sendo $p = 0,40$ e $P_{\text{FREE-p}} = 0,3616$.

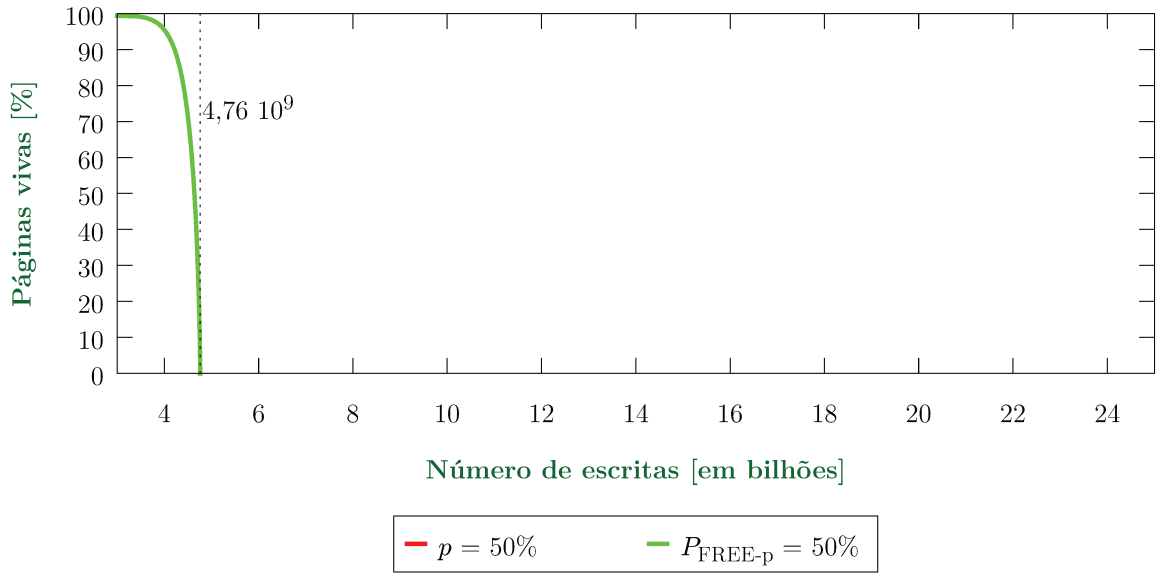


Figura 3.55: Resultados de simulações para uma memória com FREE-p, sendo $p = 0,50$ e $P_{\text{FREE-p}} = 0,4520$.

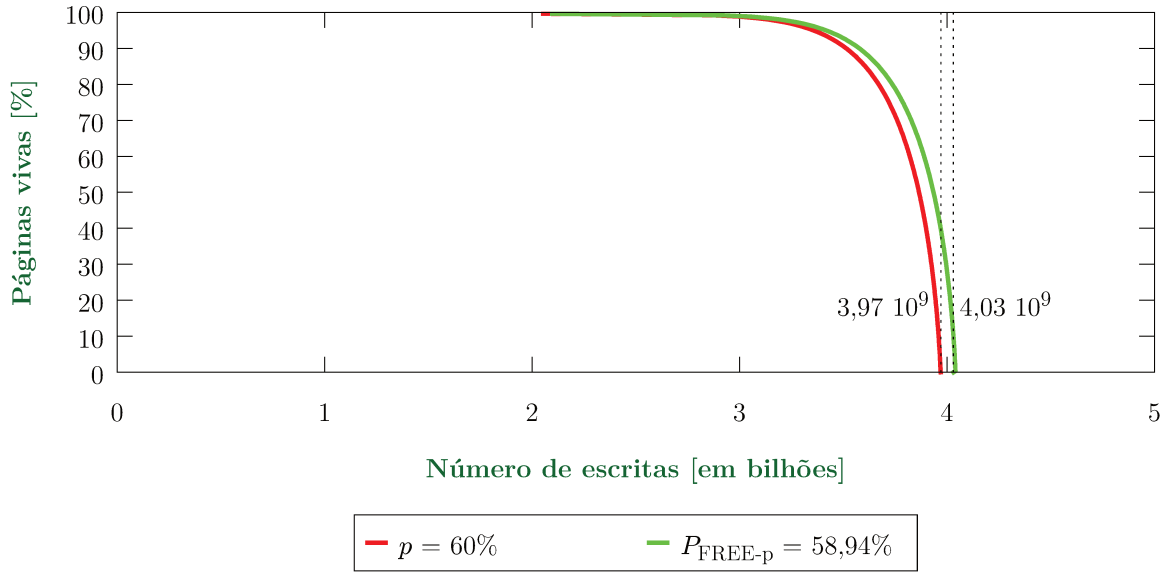


Figura 3.56: Resultados de simulações para uma memória com FREE-p, sendo $p = 0,60$ e $P_{\text{FREE-p}} = 0,5894$.

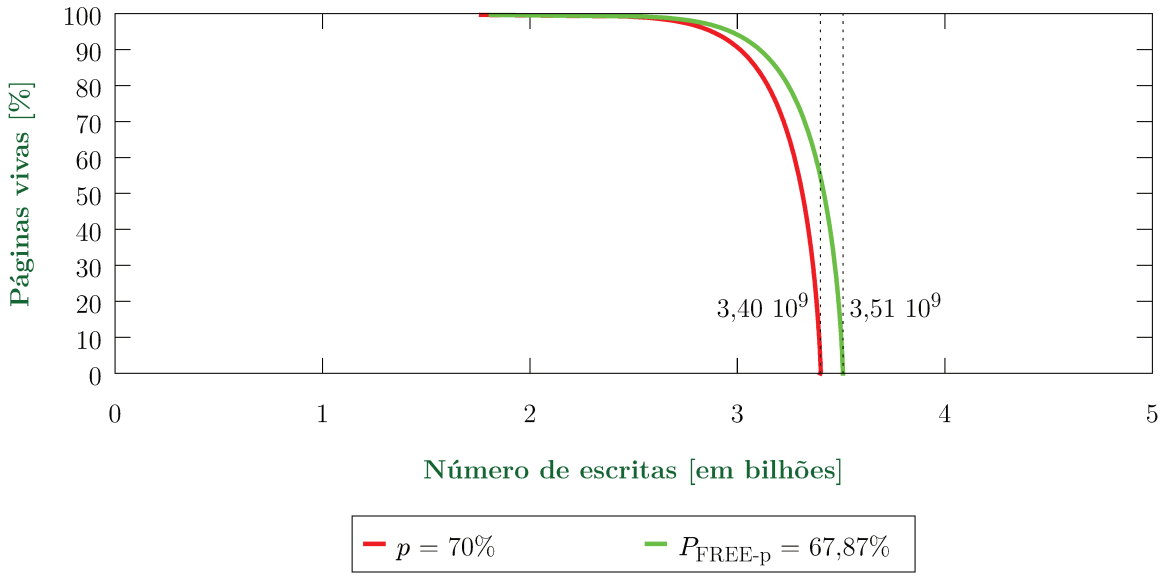


Figura 3.57: Resultados de simulações para uma memória com FREE-p, sendo $p = 0,70$ e $P_{\text{FREE-p}} = 0,6787$.

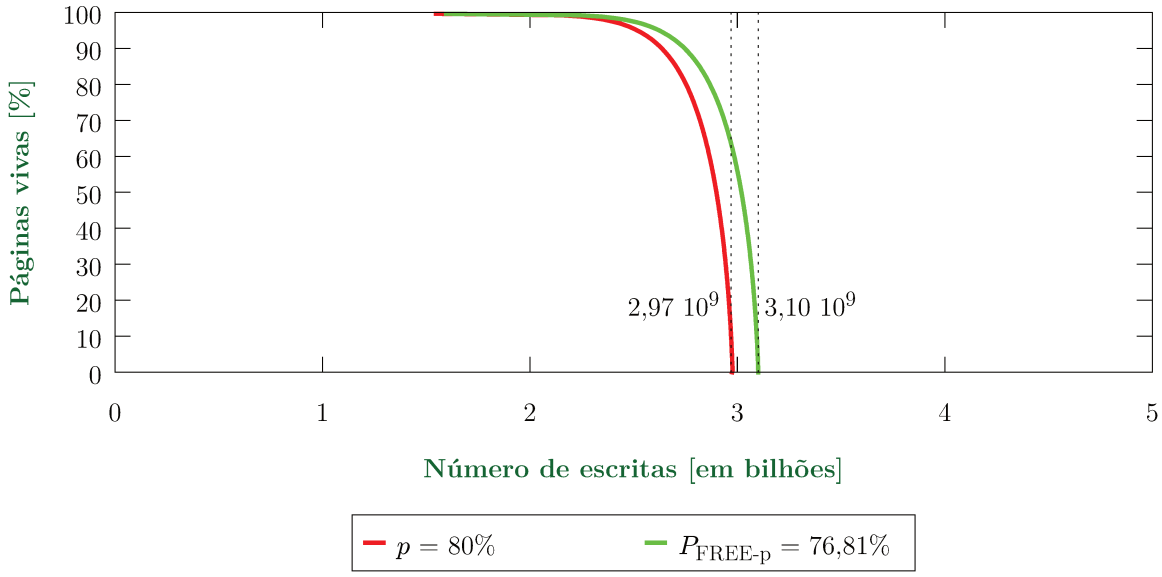


Figura 3.58: Resultados de simulações para uma memória com FREE-p, sendo $p = 0,80$ e $P_{\text{FREE-p}} = 0,7681$.

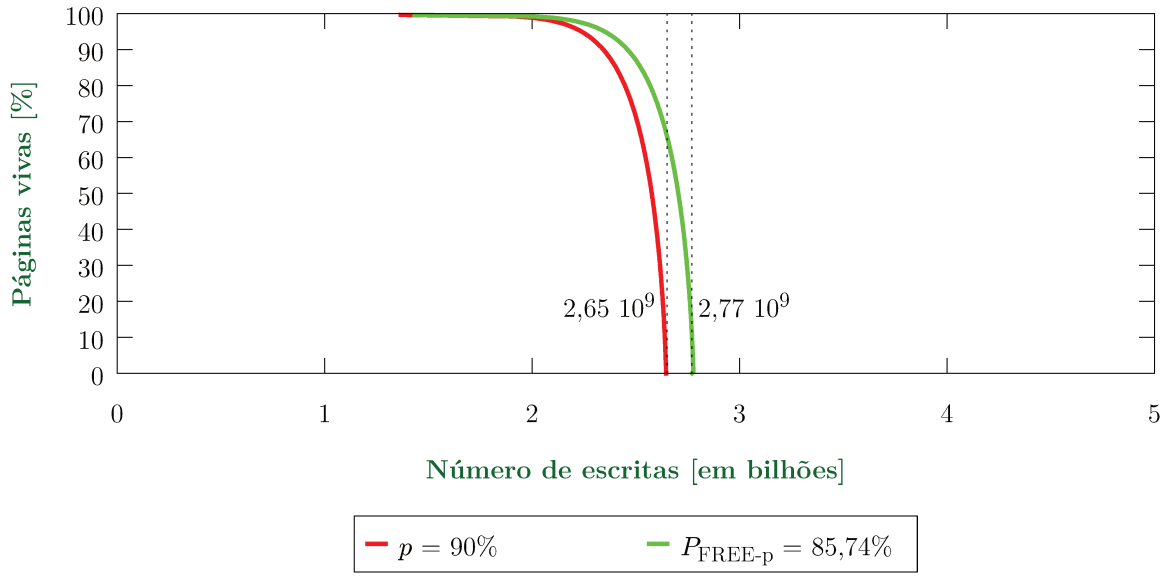


Figura 3.59: Resultados de simulações para uma memória com FREE-p, sendo $p = 0,90$ e $P_{\text{FREE-p}} = 0,8574$.

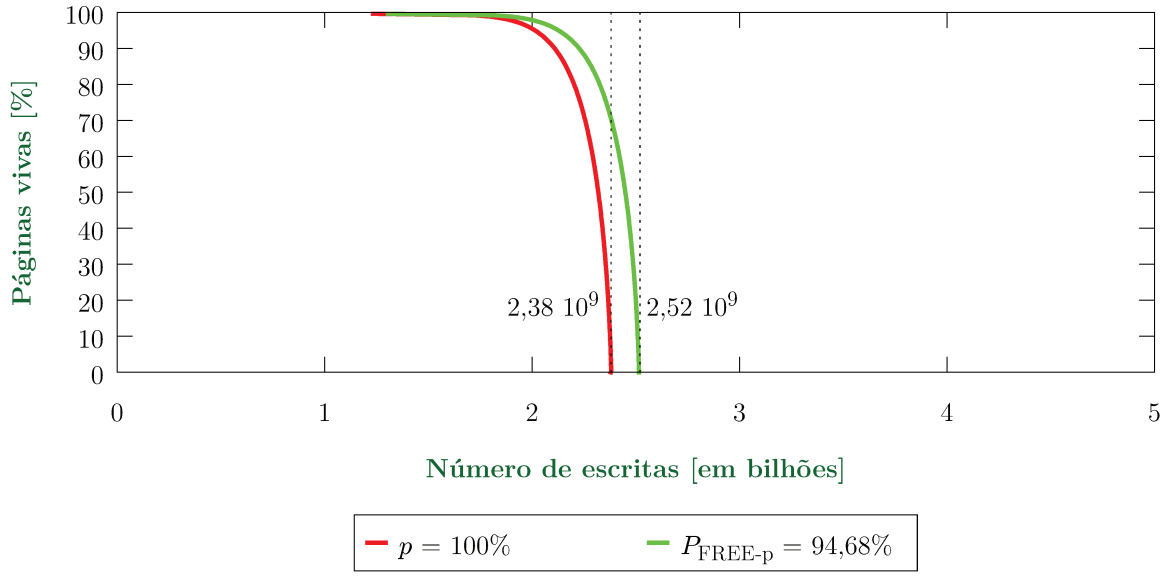


Figura 3.60: Resultados de simulações para uma memória com FREE-p, sendo $p = 1,00$ e $P_{\text{FREE-p}} = 0,9468$.

3.3 Modelos analíticos

Modelos analíticos são ferramentas tão poderosas quantos modelos de simulação, com a vantagem de serem precisos e rápidos quando a computação de suas equações não demandam muitos recursos computacionais. Modelos de simulação, como aqueles descritos na seção anterior, são algorítmicos, possuem um comportamento iterativo e a execução do passo k depende da simulação dos $k-1$ passos anteriores. Contrariamente, modelos analíticos permitem conhecer o que acontece no instante k sem conhecimento prévio. Normalmente, modelos analíticos tendem a ser de elaboração mais difícil, pela complexidade inerente de se equacionar o comportamento do sistema.

Em um trabalho de pesquisa é interessante avaliar os resultados dos trabalhos relacionados, sem haver necessidade de simulá-los, muito menos de construir um simulador para tanto. Esses esforços poderiam ser poupados com a utilização de modelos analíticos. Assim, nesta seção são propostos modelos analíticos probabilísticos – ou apenas modelos analíticos – para as técnicas estudadas na seção anterior.

Duas técnicas foram modeladas com sucesso: ECP e SECDED. A principal dificuldade em modelar técnicas como DRM e FREE-p é a modelagem do algoritmo de manipulação de páginas, visto que modelar a falha de uma linha para essas técnicas não é complicado. Por outro lado, na técnica SAFER, o número de falhas que causam o definhamento de uma linha é variado, tornando também a sua modelagem muito complexa.

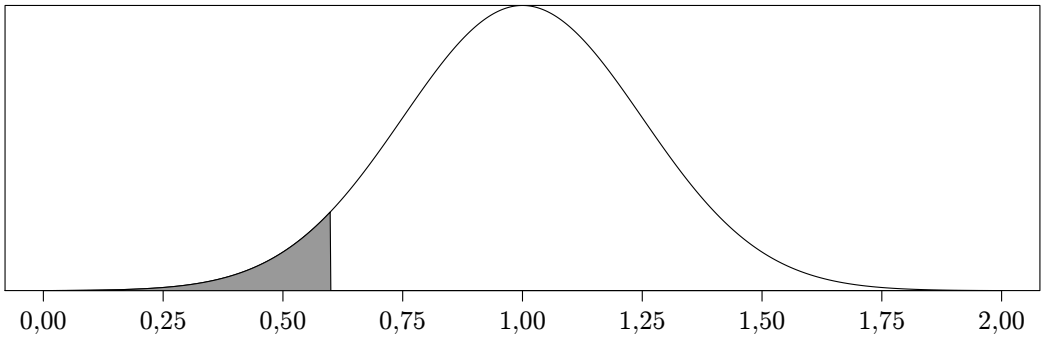
3.3.1 Modelos de falhas

Seja $\Phi(t)$ a função de distribuição acumulada de probabilidade da distribuição normal. Dados μ, tempo de vida médio de uma célula PCM, e σ desvio padrão do tempo de vida, a probabilidade $P(t)$ de uma célula ter um tempo de vida menor (ou igual) do que t é

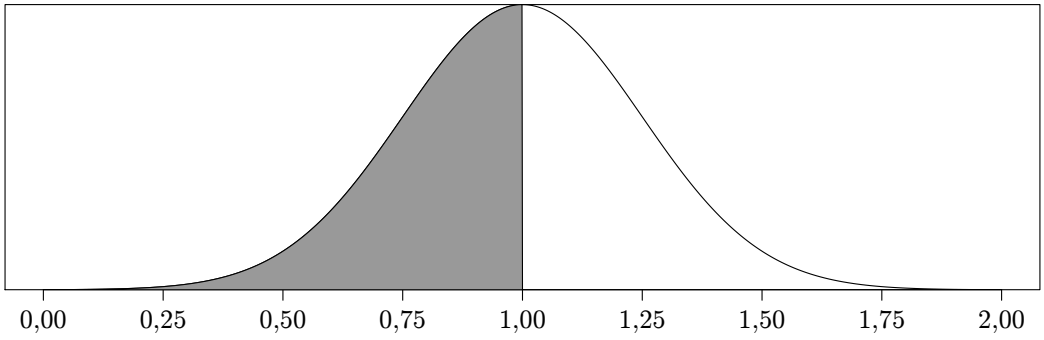
$$P(t) = \Phi\left(\frac{t-\mu}{\sigma}\right) \tag{3.24}$$

Outra forma de ver $P(t)$ seria como a probabilidade de uma célula pertencer ao grupo de células falhas na memória, dado t escritas. Na Figura 3.61 são mostradas três situações para $P(t)$, $t = 0,6 \cdot \mu$, $t = \mu$ e $t = 1,6 \cdot \mu$. A Figura 3.61(a) exibe a primeira situação, onde a probabilidade de uma célula ter o tempo de vida menor que $0,6 \cdot \mu$ é relativamente baixa. Já para na segunda situação – Figura 3.61(b) – é exibido a probabilidade de uma célula, ou estar no grupo com tempo de vida menor do que μ, ou maior do que μ, isto é, 50%, pois $P(\mu) = \Phi(0)$. Por fim, a última situação (Figura 3.61(c)) representa aquela na qual encontrar uma célula com tempo de vida maior do que $1,6 \cdot \mu$ é um evento raro.

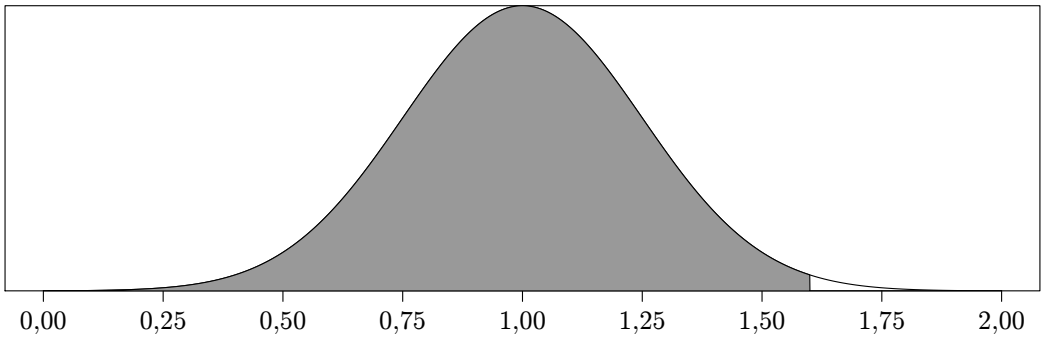
Para uma PCM usada como memória principal, costuma-se modelar: a falha de uma célula, representada pela probabilidade $P(t)$; de uma linha; de uma página; ou do sistema



(a)



(b)



(c)

Figura 3.61: Distribuição da probabilidade do tempo de vida de uma célula para três instantes diferentes. (a) Probabilidade de uma célula ter tempo de vida menor do que $t = 0,6 \cdot \mu$. (b) Probabilidade de uma célula ter tempo de vida menor que $t = \mu$. (c) Probabilidade de uma célula ter tempo de vida menor do que $t = 1,6 \cdot \mu$.

de memória. Essas falhas podem ser: dependentes, por exemplo, quando a probabilidade de uma linha falhar depende do número de células falhas que ela possui, logo, a probabilidade $\chi(t)$ de uma linha falhar com t escritas é uma função de $P(t)$; ou independentes, tal como acontece na técnica DRM[14] a qual a probabilidade $\Psi(t)$, de uma página falhar com t escritas, independe de $\chi(t)$, muito embora, nesse caso, dependa de $P(t)$.

3.3.2 Validação dos modelos analíticos

A comparação entre os modelos simulados e os modelos analíticos é feita construindo um algoritmo que, iterativamente, compute as equações do modelo analíticos para diferentes valores de t em ordem crescente. Os resultados para cada t são agrupados em um arquivo de *profile*, podendo ser traçados da mesma forma que é feita com os resultados do simulador, o que permite uma comparação visual entre os dois tipos de modelos. Vale dizer que todas suposições feitas na Subseção 3.1.2.1 também são válidas no modelo analítico.

O algoritmo iterativo de execução dos modelos tem parâmetros idênticos ao do simulador, com um adicional; ou seja, para sua execução são definidos N, Δ, N_L, N_P, μ, σ e S_T, este último é o número de iterações do algoritmo de um modelo analítico. É definido, arbitrariamente, que o valor máximo de t é $\mu+2\cdot\sigma$, uma vez que, experimentalmente, não foram observadas alterações nos resultados considerando t maiores. Para certa iteração do algoritmo, o número total de escritas é dado por

$$t(k) = k \cdot \left(\frac{\mu + 2 \cdot \sigma}{S_T}\right) \cdot \frac{1}{p} \cdot \sum_{i=1}^{k} f_{wr}(i) \tag{3.25}$$

Observando a Equação 3.25 percebe-se que ela é praticamente igual à Equação 3.4, distinguindo-se daquela no fato de que o incremento da diferença do tempo de vida é constante ($c_i = k \cdot (\frac{\mu+2\cdot\sigma}{S_T})$) e a probabilidade de *bit-flip* está inclusa. Por uma questão de simplicidade de notação, continuar-se-á utilizando apenas t para se referir ao número de escritas, pela associação direta ao tempo de vida. O arquivo de *profile* gerado no final da execução do algoritmo de um modelo é constituído de duas colunas formadas por $t(k)$ e $[1 - \Psi(t(k))] \cdot 100\%$ – note que $1 \leq k \leq S_T$ –, sendo que em cada linha há um resultado dessas equações, para um certo k.

Um problema de ordem prática foi que os modelos analíticos se tornaram indiscerníveis dos seus modelos de simulação, apenas quando o número de iterações dos seus algoritmos foi relativamente alto, isto é, $k \geq 10000$. Isso é um problema de precisão ocorrido quando duas iterações muito espaçadas – um intervalo representando muitas escritas – não contemplam valores intermediários de $f_{wr}(k)$, fazendo os resultados de um modelo analítico

[14] Lembrando que nesta técnica, para uma página de memória falhar, basta a falha de um *bit*, portanto a probabilidade de uma linha falhar é a igual à probabilidade de um *bit* falhar.

divergirem dos obtidos por simulação. Uma forma de corrigir isso é eliminar das equações o fator de f_{wr}, contabilizando apenas o número de escritas sem os efeitos de degradação. Embora isso diminua a verossimilidade, permite uma comparação e validação dos modelos mais apurada.

O ambiente de execução desenvolvido para avaliação dos modelos analíticos está disponível em [22].

3.3.3 Exemplo de modelagem: modelo analítico da ECP

Foram construídos dois modelos analíticos para a técnica ECP. Um consiste em modelar a probabilidade $\Psi(t)$ através da probabilidade $\chi(t)$ que, por sua vez, é uma função de $P(t)$. O outro define $\Psi(t)$ apenas em função de $P(t)$ usando o problema (ou paradoxo) do aniversário[15].

3.3.3.1 Modelagem de falha por linhas de memória

Nesta modelagem é necessário explicitar $\chi(t)$ a partir das condições que levam uma linha com ECP falhar. Considerado uma memória com ECP_e, uma linha falhará se mais do que e erros acontecerem. Logo, a situação na qual a linha não falhará é aquela onde ela apresenta, ou 0 erros, ou 1 erro, ou 2 erros, ..., ou e erros. Dado que uma linha tem n *bits*, existe uma forma de haver 0 erros, $\binom{n}{1}$ formas de haver 1 erro, $\binom{n}{2}$ formas de haver 2 erros, e assim em diante.

A probabilidade de haver 0 erros é $(1-P(t))^n$, de haver 1 erro é $P(t)\cdot(1-P(t))^{n-1}$, de haver 2 erros é $P(t)^2\cdot(1-P(t))^{n-2}$, etc, portanto, a probabilidade de uma linha possuir até e erros é

$$\sum_{i=0}^{e} B(i;n,P(t))$$

Complementar a essa probabilidade, é a probabilidade de haver mais do que e erros, ou seja, a probabilidade de uma linha falhar com ECP_e:

$$\chi(t) = 1 - \sum_{i=0}^{e} B(i;n,P(t)) \tag{3.26}$$

Como no modelo de simulação da técnica ECP a falha de uma linha leva a falha de sua página, a probabilidade de uma página não falhar é a probabilidade de que nenhuma de suas linhas falhe, isto é, $(1-\chi(t))^{N_L}$, onde N_L é o número de linhas de uma página. Portanto, a probabilidade de uma página falhar é

$$\Psi(t) = 1 - (1-\chi(t))^{N_L} \tag{3.27}$$

[15]Do inglês *birthday problem* (ou *paradox*).

Para averiguar este modelo analítico, ele foi comparado a um modelo de simulação de uma memória ECP_6. Seguindo o padrão deste capítulo, 1250 simulações foram executadas para a obtenção da curva de tempo de vida, utilizando os parâmetros $N = n = 512$, $\Delta = e = 6$, $N_L = 64$, $N_P = 256$, $\mu = 10^8$, $\sigma = 2,5 \cdot 10^7$, com a probabilidade de *bit-flip* nominal de 50% e $p = P_{\text{ECP}_6}$. Na obtenção da curva do modelo analítico, os mesmos valores de parâmetros foram usados no algoritmo, sendo $S_T = 10000$.

No gráfico da Figura 3.62, note que quase não se distinguem as curvas geradas pelo simulador e obtidas através do modelo analítico. Como graficamente a diferença é imperceptível, tomou-se o tempo de vida final (0% de páginas vivas) como ponto comum para as duas curvas visando calcular a relação entre o número total de escritas das duas curvas. Dessa forma, obteve-se que o ponto em análise tem valor 0,13% maior para o modelo analítico, o que é uma boa precisão, levando em conta que o último valor possui todo acúmulo das diferenças entre as duas curvas e é menor do que o maior erro padrão obtido nas simulações para o ponto em questão.

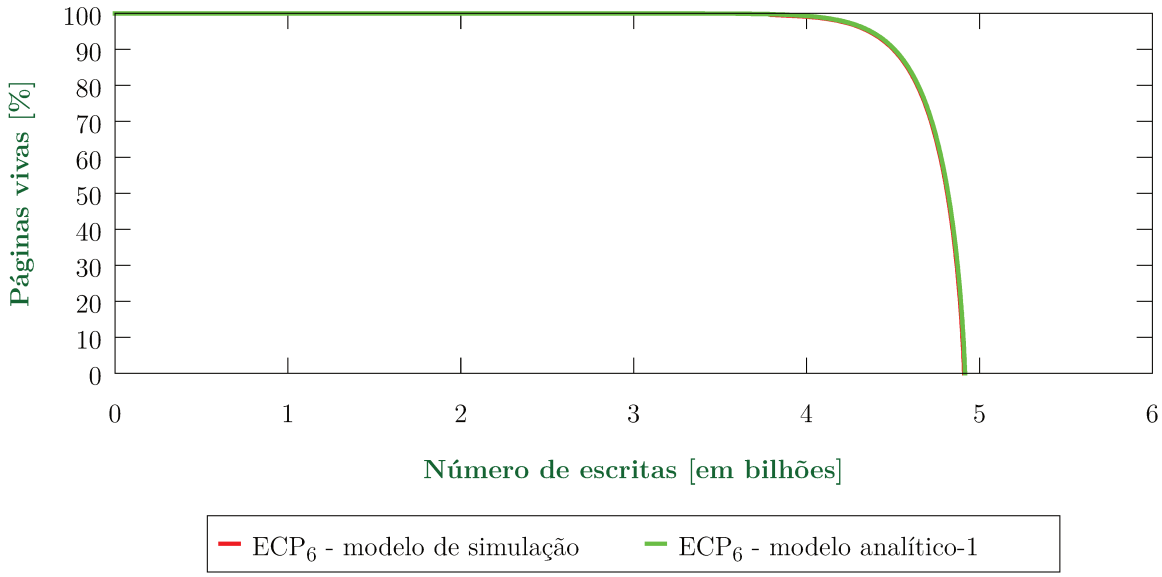


Figura 3.62: Gráfico resultante do modelo analítico da técnica ECP_6 considerando falhas por linhas de memória.

3.3.3.2 Modelagem de falha por páginas de memória

Esta modelagem parte da seguinte pergunta: dado η erros em uma página de memória com ECP_e, qual é a probabilidade de que pelo menos $e+1$ desses erros tenham ocorridos em uma mesma linha dessa página? Transcrevendo a pergunta de outra forma: dado um grupo de n pessoas, qual a probabilidade de que pelo menos r delas façam aniversário no

mesmo dia do ano? Este problema conhecido problema do aniversário generalizado[16] e foi modelado por E. H. McKinney [32].

Tradicionalmente, o problema do aniversário tenta responder qual é o número n de pessoas, sendo que pelo menos r façam aniversário no mesmo dia, tal que a probabilidade desse evento seja 50%. A equação modelada por McKinney é

$$1 - \sum \text{Prob}(n; n_1, n_2, \ldots, n_{r-1}) \tag{3.28}$$

onde

$$\text{Prob}(n; n_1, n_2, \ldots, n_{r-1}) = \frac{n!}{\prod_{j=1}^{r-1}(n_j!)(j!)^{n_j}} \cdot \frac{\text{Perm}\left(M, \sum_{i=1}^{r-1} n_i\right)}{M^n} \tag{3.29}$$

Na equação 3.29, M denota o número de dias do ano, n_i é o número de grupos de i pessoas que fazem aniversário no mesmo dia. Por exemplo, considerando $M = 365$, suponha que se deseja saber qual a probabilidade de haver mais do que 3 pessoas que façam aniversário no mesmo dia em uma festa com $n = 30$ pessoas. As possíveis distribuições de pessoas, supondo que *não há* mais do que três pessoas nascidas no mesmo dia são: $n_1 = 30$, $n_2 = 0$; $n_1 = 28, n_2 = 1$; $n_1 = 26, n_2 = 2$;...;$n_1 = 2$, $n_2 = 14$; $n_1 = 0$, $n_2 = 15$. Com essas distribuições discriminadas, aplica-se a Equação 3.29 para cada uma delas, *e.g.*,

$$\text{Prob}(30; 2, 14) = \frac{30!}{(2!)(1!)^2 \cdot (14!)(2!)^{14}} \cdot \frac{\text{Perm}(365, 2 + 14)}{365^{30}} = 9,82 \cdot 10^{-19}$$

Após a obtenção dos resultados para cada distribuição, a probabilidade de haver mais do que três é calcula pela Equação 3.28. No exemplo trabalhado, a probabilidade é 2,85%. Considere agora que o número de dias do ano é o número de linhas de uma página de memória, e que duas pessoas fazerem aniversário no mesmo dia signifique que dois erros pertencem à uma mesma linha de memória. A partir dessa analogia, tomando $M = N_L$, $n = \eta$, $r = e + 1$, tem-se o modelo de falhas por página para uma memória com ECP_e.

Os parâmetros n_1, n_2, ..., n_e, podem ser entendidos nessa analogia como o número de linhas que possuem apenas um erro, n_1, o número de linhas que possuem erros duplos, n_2, assim em diante, até se ler n_e, como o número de linhas que possuem tuplas de e erros. Logo, $\sum_{i=1}^{e} i \cdot n_i = \eta$.

A probabilidade de uma página de memória com ECP_e falhar é a probabilidade de se encontrar 0 erros em uma página e de que $e + 1$ deles estejam na mesma linha, ou

[16] O problema do aniversário original é enunciado como qual a probabilidade de que pelo menos duas pessoas, dentre n escolhidas ao acaso, façam aniversário no mesmo dia.

a probabilidade de se encontrar 1 erro em uma página e de que $e+1$ deles estejam na mesma linha, e assim consecutivamente, até se calcular a probabilidade de se encontrar $e \cdot N_L$ erros e de que $e+1$ deles estejam na mesma linha. Note que, pelo princípio da casa dos pombos, se houver mais do que $e \cdot N_L$ erros haverá, inexoravelmente, $e+1$ erros nalguma linha.

Portanto,

$$\Psi(t) = \sum_{i=0}^{e \cdot N_L} B(i; N_L \cdot N, P(t)) \cdot \left(1 - \sum \text{Prob}(i; n_1, n_2, \ldots, n_e)\right) \tag{3.30}$$

O gráfico da Figura 3.63 foi logrado a partir de 1250 execuções do modelo de simulação e da execução do algoritmo deste modelo analítico para uma memória com ECP_6. Sendo os parâmetros de execução idênticos àqueles utilizados na Seção 3.3.3.1.

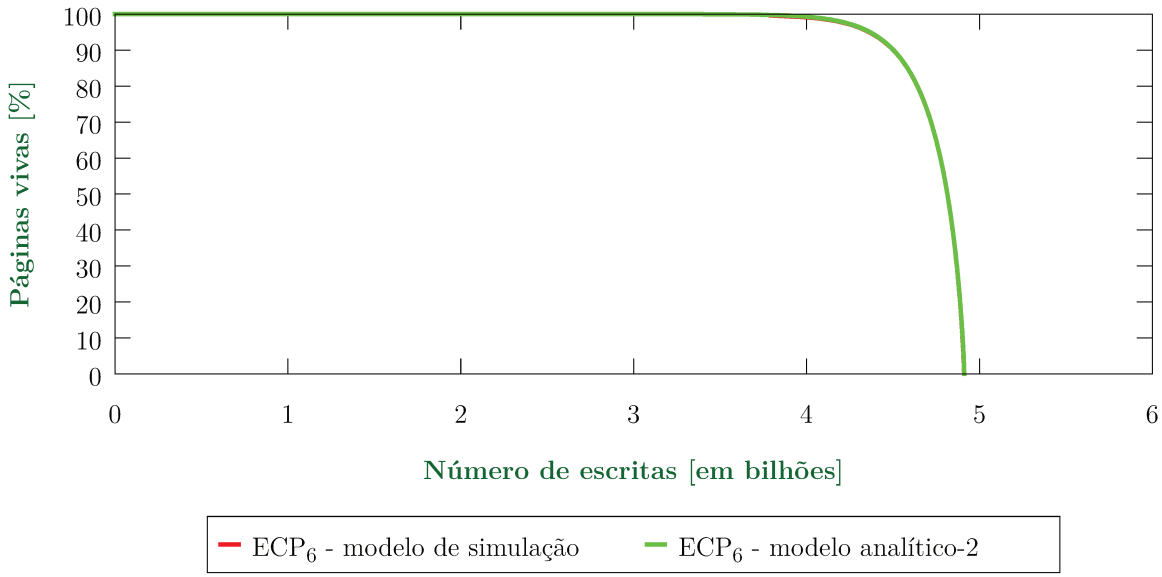


Figura 3.63: Gráfico resultante do modelo analítico da técnica ECP_6 considerando falhas por páginas de memória.

Tomando o ponto no qual as curvas possuem 0% de páginas vivas, a diferença entre o resultado do modelo de simulação e do modelo analítico é 0,03% maior para este último. Um resultado muito melhor do que o modelo analítico anterior. Todavia, esse modelo é bem mais lento, enquanto no anterior cada interação computa um somatório de tamanho e (Equação 3.26), neste computa-se um somatório de $e \cdot N_L$. Ademais, é necessário a pré-computação das probabilidades dadas pela Equação 3.28 e utilizadas na Equação 3.30; sendo que obter as probabilidades de todas possíveis combinações de grupos de erros podem demandar muito tempo.

3.3.4 Exemplo de modelagem: modelo analítico da SECDED

Este modelo é construído de maneira análoga à primeira abordagem de modelagem da técnica ECP. Contudo, como a SECDED é aplicada por blocos dentro de uma linha, faz-se necessário, *a priori*, definir a probabilidade de um bloco falhar para, desse modo, obter a probabilidade de uma linha falhar. Para SECDED(n,k) a probabilidade de um bloco falhar é a probabilidade de haver mais do que um erro, ou seja, $1-[B(0;n,P(t))+B(1;n,P(t))]$.

Sabendo que o número de *bits* de uma linha (N) é múltiplo de k, a probabilidade de uma linha falhar em uma memória com SECDED dá-se pela probabilidade complementária à não incidência de mais do que 1 erro em todos os blocos SECDED da linha, isto é,

$$\chi(t) = 1 - \Big(B(0;n,P(t)) + B(1;n,P(t))\Big)^{\frac{N}{k}} \tag{3.31}$$

Finalmente, a probabilidade de uma página de memória com SECDED falhar é dado por

$$\Psi(t) = 1 - (1-\chi(t))^{N_L} \tag{3.32}$$

A Figura 3.64 contém os resultados de 1250 simulações e da execução do algoritmo iterativo deste modelo analítico. Considerando $N = 512$, $\Delta = 64$, $N_L = 64$, $N_P = 256$, $\mu = 10^8$, $\sigma = 2,5 \cdot 10^7$ e $S_T = 10000$ (para o algoritmo do modelo analítico). Lembrando

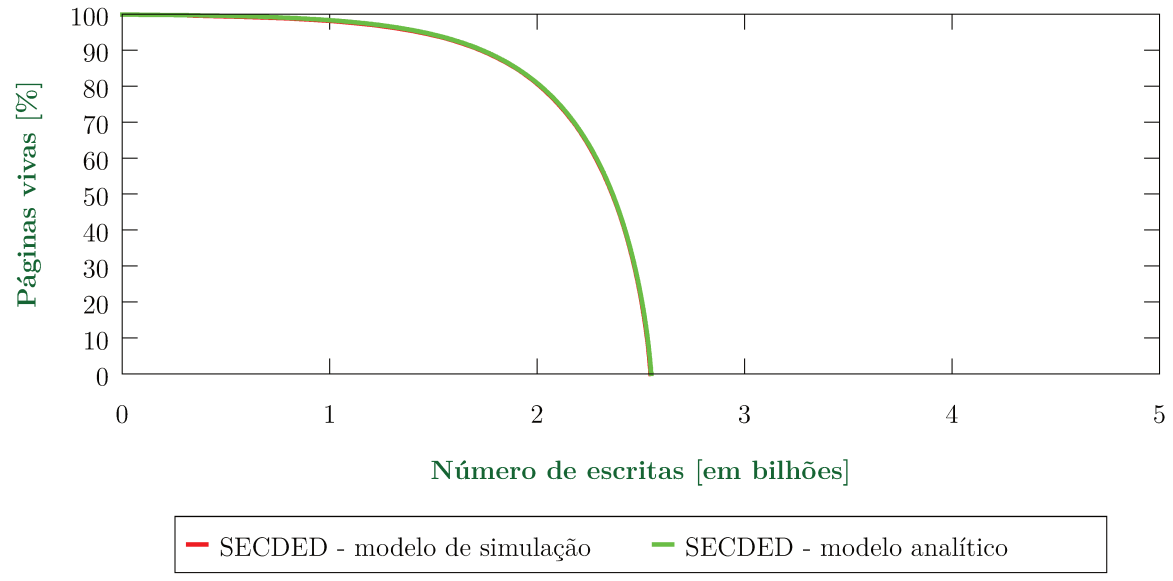


Figura 3.64: Gráfico resultante do modelo analítico da técnica SECDED considerando falhas por linhas de memória.

que uma linha de memória com b blocos SECDED implica em $n = (N + \Delta)/b$ e $k = N/b$. Analisando o tempo de vida total descrito no gráfico, tem-se que o modelo analítico termina (alcança 0% de páginas vivas) com um número de escritas 0,16% maior, ou seja, uma boa precisão, levando em conta o maior erro padrão obtido no mesmo ponto no conjunto de simulações desta técnica.

3.3.5 Modelos analíticos deslogrados

Nesta subseção a intenção é comentar as abordagens sem sucesso para elaboração dos modelos analíticos das técnicas SAFER e DRM. A grande dificuldade na modelagem da técnica SAFER é a inconstância do número de erros que leva uma linha falhar. No exemplo da Figura 3.38(b), o terceiro erro que acontece na linha é suportado de qualquer maneira, um quarto erro pode provocar a falha da linha, dependendo se ocorrerá em um grupo que já possui um erro ou não. Caso esse quarto erro caia no único grupo que não possui erro algum, a linha só falhará no quinto erro. Para a configuração da SAFER usada nas simulações deste trabalhos, ou seja, $k = 32$ e $n = 512$, a linha pode falhar entre qualquer quantidade de erros a partir do sétimo até o trigésimo segundo.

Na SAFER, tanto o esquema em *hardware*, quanto o modelo de simulação, lidam facilmente com a questão dos endereços de *bits* da linha, que é a essência do funcionamento da técnica. Porém, no modelo analítico isso parece não ser possível, visto que não existe essa noção de localidade. A abordagem para modelá-la foi através de cadeias de Markov, já que há duas opções para o acontecimento de uma falha: acontecer em um grupo que já possui outra falha com certa probabilidade, ou acontecer em um grupo que não possui falhas com a probabilidade complementar. Dessa forma, na árvore de probabilidades, a probabilidade de uma linha falhar, depende da probabilidade de uma sequência de eventos, nos quais erros colidiriam em um mesmo grupo. Infelizmente, não houve sucesso nessa abordagem embora fosse aquela que apresentou uma melhor aproximação do modelo de simulação.

As técnicas FREE-p e DRM, diferentemente da SAFER, possuem modelagem mais simples da probabilidade de uma linha falhar, equivalente às descritas nas Subseções 3.3.3 e 3.3.4, entretanto, as páginas sofrem alterações de função, como linhas armazenando ponteiros ao invés de dados e páginas emparelhadas que representam uma só. Além disso, há, no sistema de memória, gerenciamento de alocação páginas, filas, *pools*, etc. Essas características acabam transformando a modelagem um desafio.

Conquanto não tenha havido tentativa de modelar a FREE-p, pela semelhança com a DRM e por ser mais elaborada do que esta, acredita-se que também haveria insucesso na sua modelagem. Uma das grandes dificuldades na modelagem da DRM é explicitar probabilisticamente o emparelhamento de duas páginas quando ambas não possuem um

erro no mesmo *byte*. Mesmo a probabilidade divulgada no trabalho original [23] não é bem explicada, tendo constantes na equação que não remetem a valor algum da descrição da técnica. Contudo, não se pode afirmar que é a imprecisão da probabilidade de emparelhamento que torna o modelo incorreto.

Enfim, espera-se que essas descrições, bem como o código dos algoritmos de execução dos modelo analíticos disponíveis em [22], forneçam informações suficientes para ajudar na elaboração de trabalhos futuros.

3.4 Síntese

Para cada uma das técnicas de correção de erros, modeladas e analisadas, é possível observar que aquelas possuintes de códigos de correção de erros, tradicionalmente utilizados em teoria da informação, têm seus desempenhos deteriorados, principalmente quando se analisa probabilidades de *bit-flip* menores do que 50%. Nas simulações, mesmo quando a probabilidade de *bit-flip* considerada é 50%, o comportamento difere daquele apresentado na literatura por conta de não se considerar que as taxas de *flips* de dados e da metainformação podem ser diferentes.

Tabela 3.7: Sumário analítico das técnicas de correção de erros. Na linha Erros corrigíveis está se levando em conta uma linha de 512 *bits*, quando não informado o número de *bits*.

	ECP_6	DRM	SECDED	$SAFER_{32}$	FREE−p
Erros corrigíveis	até 6 *bits*	0 *bits* (apenas detecta 1 erros cada *byte*)	1 em cada 71 *bits* (detecta até 2 em 72 *bits*)	de 7 até 32 *bits*	até 4 *bits*
Código de correção	Sem código (usa mecanismo próprio)	Paridade	Hamming + Paridade	Sem código (usa mecanismo próprio)	BCH
Probabilidade ajustada	para todos os valores de p, o desgaste é amenizado	quanto menor o valor p, maior é o desgaste provocado	quanto menor o valor p, maior é o desgaste provocado	o ajuste não provoca desgaste significativo	quanto menor o valor p, maior é o desgaste provocado

À medida que a probabilidade de *bit-flip* aumenta para além dos 50%, o ajuste realizado pela modelagem da probabilidade de *bit-flip* mostra uma atenuação do desgaste para todas as técnicas. O problema é que não se deseja uma taxa de *bit-flip* acima de 50%, pois isso acelera o desgaste das células de memória. Além disso, como será analisado no próximo capítulo, aparentemente a taxa de *bit-flip* de programas reais está bem abaixo dos 50%. Na Tabela 3.7 sumariza-se esses resultados acima descritos.

Na Seção 3.3, discorreu-se brevemente sobre a importância e utilidade do uso de modelos analíticos, além de se mostrar três modelagens construídas com sucesso. No próximo capítulo, analisar-se-á a técnica PAYG, construída sobre a análise do modelo analítico da técnica ECP, demonstrando, dessa forma, uma outra forma de utilização de modelos analíticos.

Capítulo 4

Análise de resultados

No Capítulo 3 analisou-se o comportamento da durabilidade de um sistema de memória se utilizando de várias probabilidades de *bit-flip*, entretanto, na literatura o valor é sempre fixado em 50%. Dessa forma, neste capítulo, descreve-se o que foi feito para a obtenção de uma taxa experimental de *bit-flip*, obtida por meio da execução de programas reais. A problematização do uso do valor teórico de 50%, quando se leva em conta a modelagem feita no capítulo anterior, é então discutida na próxima seção.

Em sequência, neste capítulo, utiliza-se novamente a modelagem da probabilidade de *bit-flip* para se analisar consumo energético. Enquanto na literatura nenhuma das técnicas de correção de erros estudadas possuía uma análise de consumo de energia, com a separação comportamental dos *bits* de dados e de metainformação, tornou-se viável estudar o consumo de energia de cada técnica ao se realizar uma escrita. Desse modo, é possível comparar quantitativamente as técnicas não só em função da durabilidade que proporcionam, mas também sobre o consumo de energia. Por fim, na Seção 4.2, analisamos a técnica PAYG, e os problemas que foram encontrados durante sua tentativa de implementação.

4.1 *Bit-flip* experimental

Conforme explicado no Capítulo 3, a taxa experimental de *bit-flips* ρ é obtida pela execução do conjunto de programas através de uma ferramenta de instrumentação (*PIN Tool*) que simula uma memória *cache* de processador. Na Tabela 4.1 tem-se a configuração da *cache* usada na instrumentação da execução do SPEC CPU 2006 v1.2. O valor médio de *bit-flip* de cada conjunto *benchmark* e entrada estão na Tabela 4.2. Infelizmente, não foi possível obter um erro estatístico para os valores estimados de *bit-flip* de cada *benchmark*, pois como as simulações levavam dias para finalizar, seria necessário um ano de simulações para alcançar uma quantidade de valores significativa.

Tabela 4.1: Configuração da memória cache simulada com a instrumentação da execução do SPEC CPU.

Paramêtro	Valor	Nível
Linhas	128	L1
Conjuntos	8	L1
Tamanho de linha	64 *bytes*	L1
Tamanho (em dados)	8 KB	L1
Linhas	4096	L2
Conjuntos	8	L2
Tamanho de linha	64 *bytes*	L2
Tamanho (em dados)	256 KB	L2
Política de substituição	*LRU*	L1, L2
Política de escrita	*writeback*	L1, L2

Os valores esperados de *bit-flip* para cada *benchmark* estão discriminados na terceira coluna da Tabela 4.2, sendo que na última linha está τ (taxa média de *bit-flip*) de todo o SPEC. O tamanho de linha de memória considerado é igual ao tamanho de uma linha da *cache* simulada, 512 *bits*. Dessa forma, o cálculo de ρ é dado por $\tau/N = 77,97/512 = 15,23\%$ (lembrando que τ é obtido pela equação 3.1), valor que corrobora com o que foi divulgado em [25] e apenas um pouco acima do que foi divulgado em [20], 13%. Nesses trabalhos, porém, nem a metodologia nem os *benchmarks* são iguais, na verdade, a metodologia, ou não é clara, ou é omitida. Além disso, embora os *benchmarks* utilizados façam parte do SPEC2006, apenas alguns são avaliados (diferentemente, deste trabalho que avalia todos), sendo que em [25] também é usado adicionalmente *benchmarks* do STREAM.

O trabalho de Hay *et. al* [20] tem uma certa peculiaridade sobre a obtenção da quantidade de *bit-flips*. Sua memória se utiliza do *Flip-N-Write* [11] para limitar a quantidade de *flips* a 50%. Lembrando que a ideia do mecanismo é bem simples: por exemplo, se a palavra a ser escrita em determinado endereço gerará 60% de inversões de *bits*, então, ela é invertida antes de ser escrita, fazendo com que apenas 40% dos *bits* daquele endereço sejam invertidos.

Apesar da memória *cache* de instrumentação não ter sido implementada com essa opção, ainda é possível obter esse resultado pelo arquivo de *profile* gerado pela simulação. O arquivo possui duas colunas sendo que a primeira possui a quantidade de *flips* (de 0 à N) e a segunda a quantidade de escritas. O que se faz, através de um *script*, é somar às linhas com $i - N/2$ *flips*, a quantidade de escritas da linha com i *flips*, quando $i > N/2$. O resultado de $\tau/N = 75.90/512 = 14,82\%$, dessa forma, optou-se por considerar $\rho = 15\%$,

visto que os valores obtidos usando ou não o esquema de inversão são praticamente iguais.

Tabela 4.2: Quantidade de *flips* por entrada de *benchmark* do SPEC2006. Na terceira coluna encontra-se os resultados de τ para cada entrada, bem como para cada *benchmark*. Na última linha denota-se o τ da execução de todo *suite*

Benchmark	Entrada	Média de *bit-flip*
400.perlbench	*checkspam*	38,67
	diffmail	149,62
	splitmail	185.33
	Subtotal:	144,29
401.bzip2	*chicken.jpg*	35,47
	input.source	121,75
	liberty.jpg	34,60
	text.html	115,93
	Subtotal:	69,73
403.gcc	*166*	81,49
	200	37,27
	cp-decl	63,58
	c-typeck	29,76
	expr2	30,87
	expr	24,29
	g23	21,03
	s04	36,34
	scilab	39,25
	Subtotal:	32,29
410.bwaves	*bwaves*	181,81
416.gamess	*cytosine.2*	218,75
	h2ocu2+.gradient	209,34
	triazolium	123,08
	subtotal:	123,98
429.mcf	*inp*	10,66
433.milc	*su3imp*	188,66
434.zeusmp	*zmp_inp*	81,15
435.gromacs	*gromacs*	152,08
436.cactusADM	*benchADM*	125,50
437.leslie3d	*leslie3d*	125,50
444.namd	*namd*	174,51

Tabela 4.2: Quantidade de *flips* por entrada de *benchmark* do SPEC2006. Na terceira coluna encontra-se os resultados de τ para cada entrada, bem como para cada *benchmark*. Na última linha denota-se o τ da execução de todo *suite* (continuação).

Benchmark	Entrada	Média de *bit-flip*
445.gobmk	*13x13*	3,68
	nngs	26,28
	score2	34,44
	trevorc	19,02
	trevord	28,54
	subtotal:	15,69
447.dealII	*DummyData*	82,00
450.soplex	*pds-50*	24,44
	ref	150,46
	subtotal:	67,28
453.povray	*SPEC-benchmark-ref*	63,12
454.calculix	*hyperviscoplastic*	209,66
456.hmmer	*nph3*	118,48
	retro	119,66
	subtotal:	118,50
458.sjeng	*ref*	35,09
459.GemsFDTD	*sphere*	154,26
462.libquantum	*control*	3,43
464.h264ref	*foreman_ref_encoder_baseline*	63,46
	foreman_ref_encoder_main	73,66
	sss_encoder_main	82,37
	subtotal:	118,50
465.tonto	*stdin*	213,72
470.lbm	*100_100_130_ldc*	46,33
471.omnetpp	*omnetpp*	19,70
473.astar	*astar_rivers*	22,10
	BigLakes2048	14,96
	subtotal:	16,88
481.wrf	*namelist*	134,33
482.sphinx3	*args.an4*	21,11
483.xalancbmk	*t5.xml*	66,20
483.xalancbmk	*t5.xml*	66,20
998.specrand	*control*	184,09

Tabela 4.2: Quantidade de *flips* por entrada de *benchmark* do SPEC2006. Na terceira coluna encontra-se os resultados de τ para cada entrada, bem como para cada *benchmark*. Na última linha denota-se o τ da execução de todo *suite* (continuação).

Benchmark	Entrada	Média de *bit-flip*
999.specrand	*control*	184,67
	Total:	**77,97**

4.1.1 Comparação entre as técnicas

Para uma análise crível da transformação que o ajuste da probabilidade de *bit-flip* gera sobre as técnicas estudas, foram feitas 1250 simulações com $N = n = 512$, $N_L = 64$, $N_P = 256$, $\mu = 10^8$ e $\sigma = 2,5 \cdot 10^7$, usando os valores de ρ= 13% e 15%, e de $p = 50\%$. A intenção é comparar se os valores de *bit-flip* obtidos experimentalmente produzem situações de desempenho diferentes daquelas apresentadas na literatura e se, mesmo para probabilidade $p = 50\%$, o ajuste das probabilidades de *bit-flip* fornece resultados diferentes daqueles já apresentados. As comparações entre as probabilidades nominais e ajustadas estão apresentadas na Tabela 4.3.

Quando não há ajuste, como nos gráficos das Figuras 4.1, 4.3 e 4.5, os resultados quantitativos permanecem como divulgados na literatura, isto é, o tempo de vida de uma memória com DRM é menor do que uma com SECDED que, por sua vez, é menor do que uma com ECP (veja [43]). Tanto a SAFER, quanto a FREE-p prolongam a vida de uma

Tabela 4.3: Comparação entre valores de probabilidade de *bit-flip* com e sem ajuste para as simulações com os valores experimentais de $\rho = \{13\%, 15\%\}$ e o valor usado na literatura $p = 50\%$. Dado $n = 512$. A terceira, quinta e sétima coluna representam o aumento ou a redução porcentual do valor da probabilidade ajusta em relação aos valores experimentais e à $p = 50\%$.

	$\rho = 13\%$	%	$\rho = 15\%$	%	$p = 50\%$	%
P_{ECP_6}	11,75%	9,60% ↓	13,56%	9,60% ↓	45,20%	9,60% ↓
P_{DRM}	16,61%	27,78% ↑	18,57%	23,79% ↑	50,00%	-
P_{SECDED}	17,03%	30,97% ↑	18,83%	25,54% ↑	50,00%	-
$\min(P_{\text{SAFER}_{32}})$	12,47%	4,06% ↓	14,39%	4,06% ↓	47,97%	4,06% ↓
$\mu(P_{\text{SAFER}_{32}})$	15,27%	17,46% ↑	17,04%	13,58% ↑	47,97%	4,06% ↓
$\max(P_{\text{SAFER}_{32}})$	19,25%	48,08% ↑	20,80%	38,67% ↑	47,97%	4,06% ↓
$P_{\text{FREE-p}}$	16,94%	30,30% ↑	18,73%	24,84% ↑	50,00%	-

memória mais do que a ECP (veja [46, 60]), contudo, não há comparação entre elas na literatura.

Com a probabilidade de *bit-flip* ajustada, o desempenho da ECP supera quase todas as outras técnicas, exceto a SAFER, sendo que para $\rho = 13\%$ e $\rho = 15\%$ ambas praticamente têm durabilidade igual, a diferença apenas aparece com $p = 50\%$, isso acontece pela capacidade maior de correção de erros que a SAFER possui. Claramente, a utilização de código corretores de erros com probabilidade de *bit-flip* de 50% degrada significativamente a durabilidade da memória, basta observar como o desempenho da FREE-p difere nos gráficos onde a probabilidade está ajustada e onde não está.

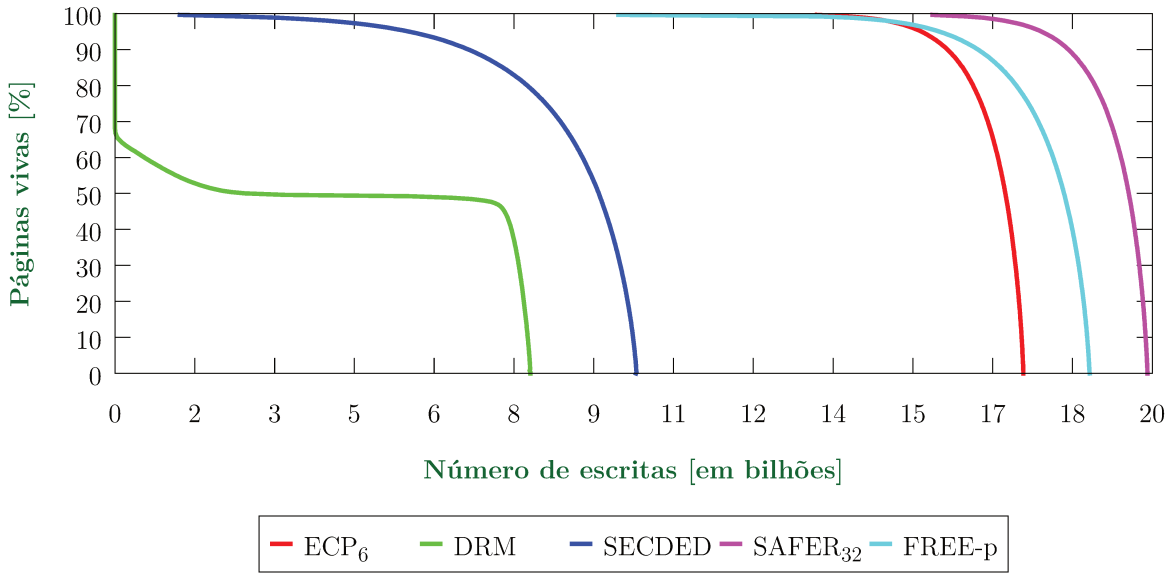


Figura 4.1: Comparativo entre as técnicas para probabilidade de *bit-flip* $\rho = 0,13$ sem ajuste.

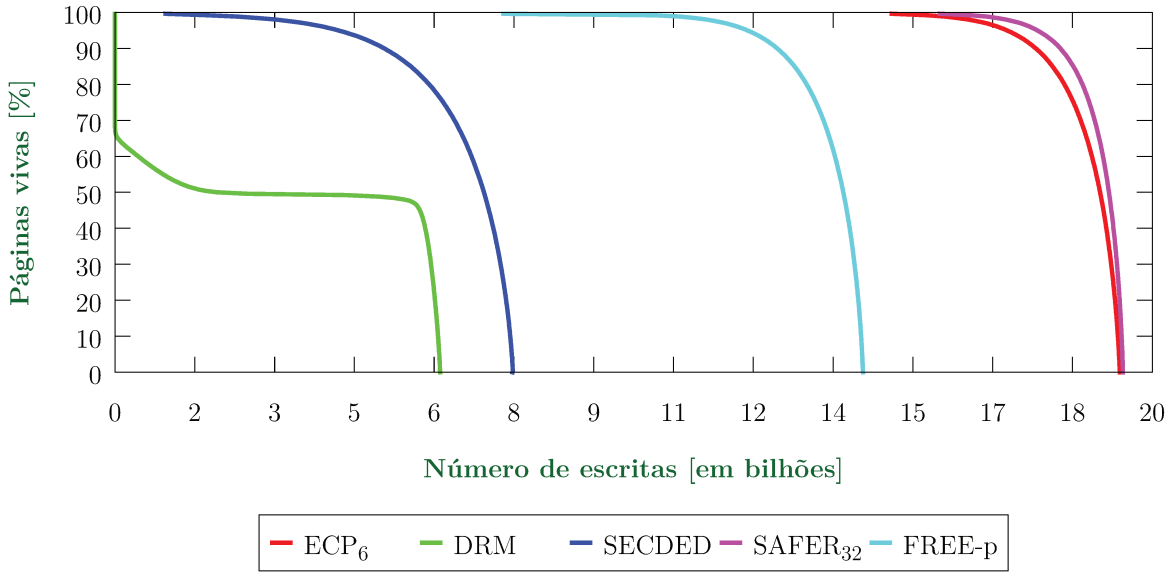


Figura 4.2: Comparativo entre as técnicas para probabilidade de *bit-flip* $\rho = 0,13$ com ajuste.

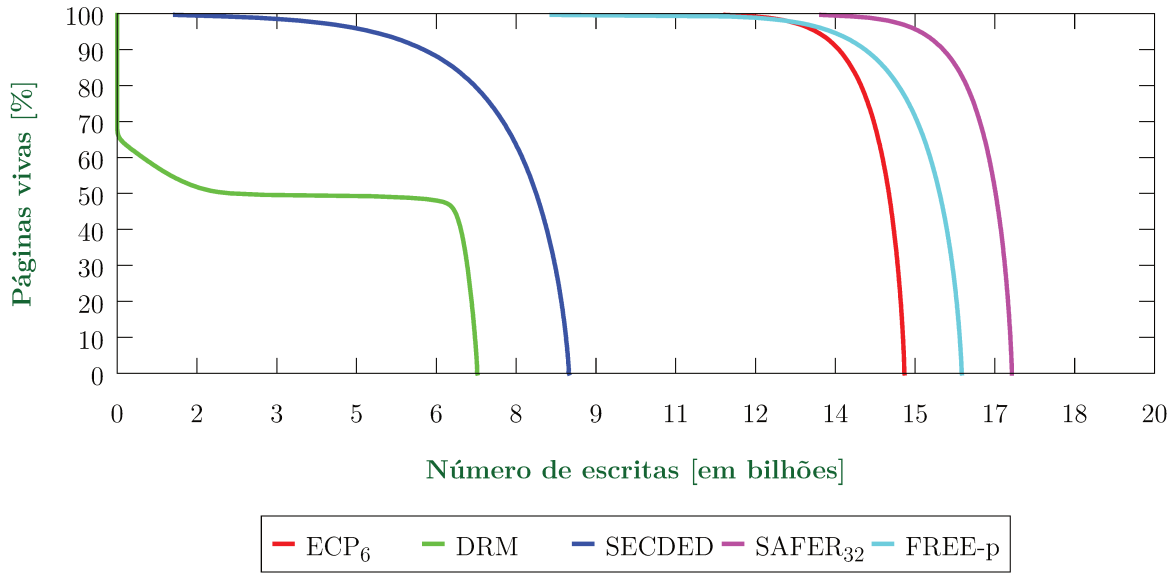


Figura 4.3: Comparativo entre as técnicas para probabilidade de *bit-flip* $\rho = 0,15$ sem ajuste.

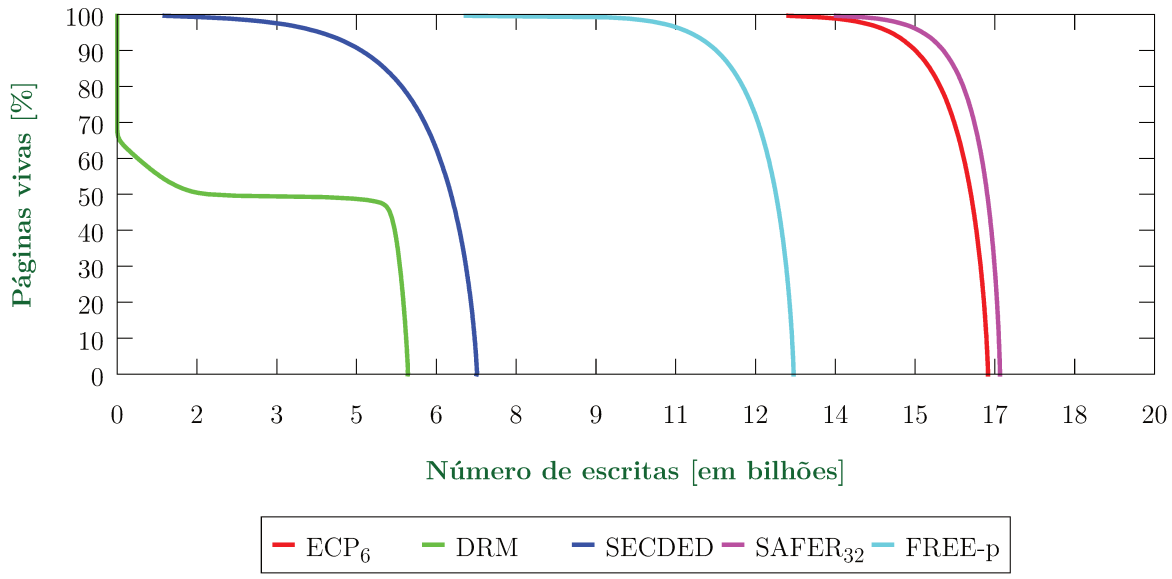


Figura 4.4: Comparativo entre as técnicas para probabilidade de *bit-flip* $\rho = 0,15$ com ajuste.

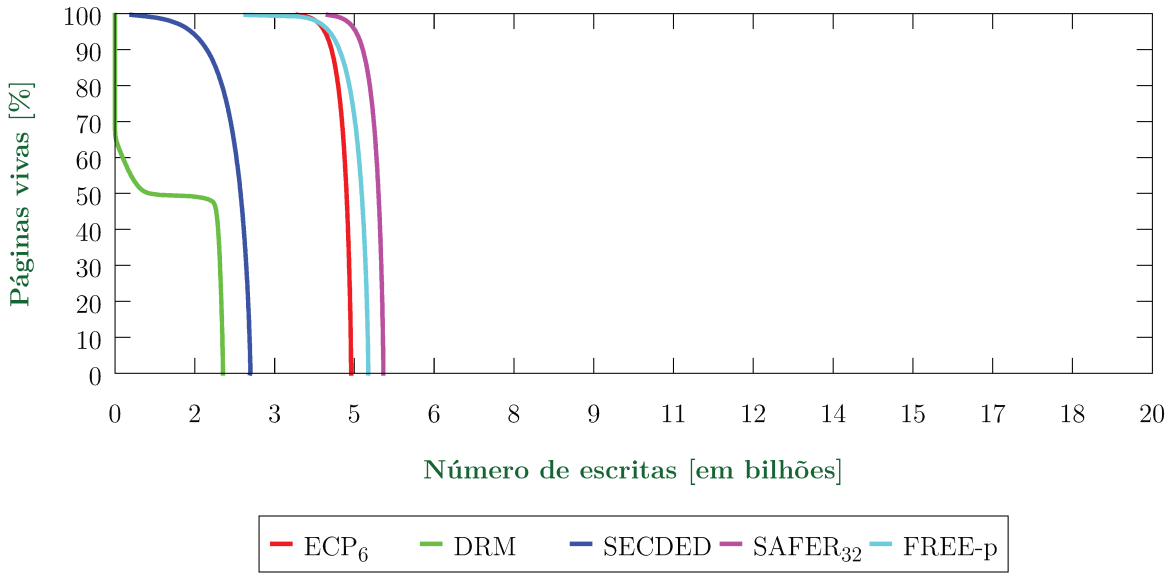


Figura 4.5: Comparativo entre as técnicas para probabilidade de *bit-flip* $p = 0,50$ sem ajuste.

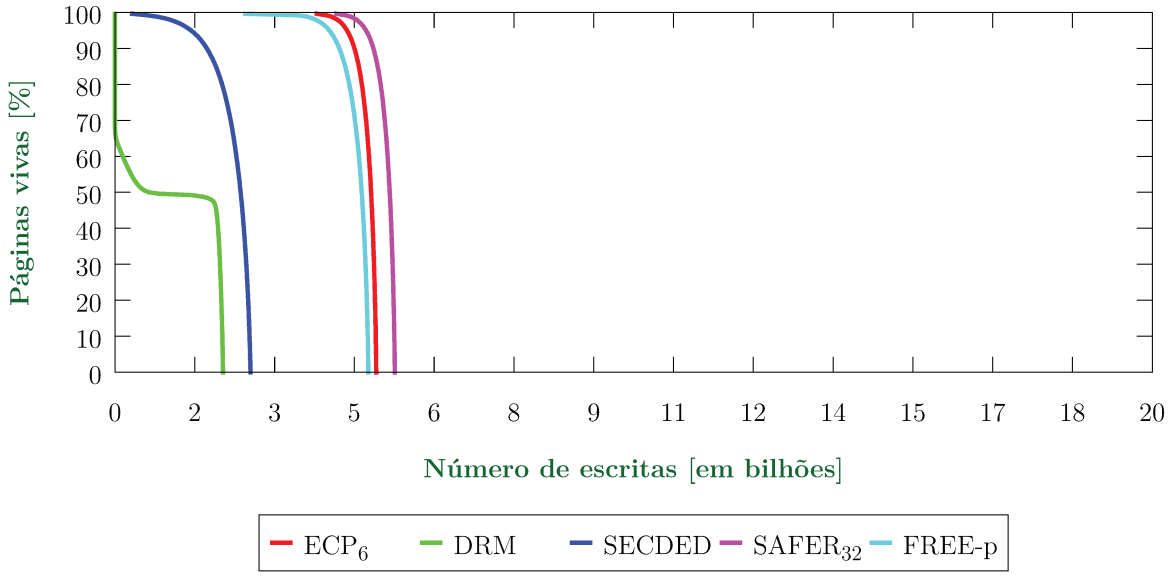


Figura 4.6: Comparativo entre as técnicas para probabilidade de *bit-flip* $p = 0,50$ com ajuste.

4.1.2 Consumo de energia

No Capítulo 3, para cada uma das técnicas ECP, DRM, SECDED, SAFER e FREE-p, foram mostradas como o ajuste da probabilidade de *bit-flip* pode aumentar a taxa de *flips* de uma linha e, consequentemente, diminuir sua duração.

Além disso, o aumento na taxa de modificações dos *bits* de um endereço de memória, também, implica num gasto maior de energia, independentemente do tipo de operação de escrita de *bit*: *set* ou *reset*. Portanto, analisar a probabilidade de *bit-flip* é uma forma indireta de analisar o consumo de energia necessário para escrever uma palavra ou linha de memória.

O consumo de energia de uma escrita para o modelo de simulação sem ajuste da probabilidade de *bit-flip*, pode ser formulado como $N \cdot p \cdot (E_{set} \cdot 0,5 + E_{reset} \cdot 0,5)$, para qualquer técnica, uma vez que as energias das operações *set* e *reset* para um *bit* são distintas e a probabilidade de uma ou outra ocorrer é a probabilidade de acontecer a transição $0 \rightarrow 1$ ou $1 \rightarrow 0$, as quais ambas são de 50%. Agora, considerando a probabilidade de *bit-flip* ajustada, é possível comparar a eficiência energética por escrita de cada técnica. Definindo a energia gasta na escrita de uma linha de memória para uma técnica X por

$$\xi_X = N_X \cdot P_X \cdot \left(\frac{E_{set} + E_{reset}}{2} \right) \tag{4.1}$$

Sendo $X \in \{\text{ECP}_6, \text{DRM}, \text{SECDED(72,64)}, \text{SAFER}_{32}, \text{FREE-p}\}$. No Capítulo 3, os tamanhos totais de linha não foram expressos isoladamente, pois, a seguir serão apresentados:

$$N_{\text{ECP}_e} = n + e \cdot (\log_2(n) + 1) + 1 \tag{4.2}$$

$$N_{\text{DRM}} = N/n \cdot (n + e) \tag{4.3}$$

$$N_{\text{SECDED}(n,k)} = n \cdot N/k \tag{4.4}$$

$$N_{\text{SAFER}_k} = n + k + \lceil \log_2 k \rceil \cdot \lceil \log_2 \lceil \log_2 n \rceil \rceil + \lceil \log_2 (\lceil \log_2 k \rceil + 1) \rceil \tag{4.5}$$

$$N_{\text{FREE-p}} = N_{\text{BCH}(n,k)} + 1 = n + 1 \tag{4.6}$$

A configuração dos valores de n, k e e para cada técnica, bem como o valor os valores do tamanho da linha, estão expressas na Tabela 4.4, considerando $N = 512$. As energias E_{reset} e E_{set} foram obtidas utilizando a configuração descrita no trabalho [30] e calculada no simulador NVSIM, divulgado em [19]. Isso foi feito, primeiro, para uma familiarização com a ferramenta NVSIM e, segundo, para permitir a utilização de configurações mais modernas do que aquelas normalmente citadas na literatura [29, 58], contendo parâmetros de processo de fabricação, tamanho de células, tensão de alimentação, etc, mais atuais. A despeito dos valores de energia obtidos diferirem daqueles normalmente usados, para a análise a ser feita, isso pouco importa, visto que os valores de energia são meras

constantes, assim, $E_{reset} = 301,25\text{pJ}$ e $E_{set} = 481,25\text{pJ}$. Na Tabela 4.5 são visualizadas as energias para escrita de uma linha de memória relacionada com as probabilidades de *bit-flip* apresentadas na Tabela 4.3.

Tabela 4.4: Tamanho de linha de memória para cada técnica, isto é, valores para N_X, com $X \in \{\text{ECP}_6, \text{DRM}, \text{SECDED(72,64)}, \text{SAFER}_{32}, \text{FREE-p}\}$.

ECP_6	DRM	SECDED(72,64)	SAFER_{32}	FREE-p
$n = 512 \quad e = 6$	$n = 8 \quad e = 1$	$n = 72 \quad k = 64$	$n = 512 \quad k = 32$	$n = 572$
$N_{\text{ECP}_6} = 573$	$N_{\text{DRM}} = 576$	$N_{\text{SECDED}} = 576$	$N_{\text{SAFER}_{32}} = 567$	$N_{\text{FREE-p}} = 573$

Tabela 4.5: Energia consumida na escrita de uma linha de memória para cada técnica. A probabilidade de *bit-flip* utilizada no cálculo da energia é o valor ajustado para as probabilidades $\rho = 13\%$, $\rho = 15\%$ e $p = 50\%$.

P_X	ξ_X (nJ)		
	$\rho = 13\%$	$\rho = 15\%$	$p = 50\%$
P_{ECP_6}	26,34	30,33	101,33
P_{DRM}	37,43	41,85	112,68
P_{SECDED}	38,38	42,44	112,68
$\min(P_{\text{SAFER}_{32}})$	27,66	31,92	106,42
$\mu(P_{\text{SAFER}_{32}})$	33,87	37,80	106,42
$\max(P_{\text{SAFER}_{32}})$	42,70	46,14	106,42
$P_{\text{FREE-p}}$	37,98	41,99	112,09

Para estabelecer uma métrica de comparativa entre a duração de uma memória com uma técnica X e seu desempenho de consumo de energia, define-se

$$\Lambda_X = \frac{\mathbb{W}_X(\omega)}{\xi_X} \tag{4.7}$$

Lembrando que ω é o último passo da simulação antes da memória falhar completamente, logo, $\mathbb{W}(\omega)$ é quando a curva dos gráficos encontra a abscissa. Analisando Λ, observa-se que quanto maior a durabilidade propiciada a uma memória, maior ele será. Entretanto, se o consumo de energia por escrita for alto, Λ diminuirá; portanto, Λ pode mensurar o quão bom é a técnica nesses dois aspectos fundamentais. Na Tabela 4.6 mostra-se os valores de Λ logrados para cada técnica em relação ao Λ_{ECP_6}, para cada valor de probabilidade, considerando apenas as probabilidades de *bit-flip* ajustadas e avaliadas neste capítulo.

Tabela 4.6: $\Lambda_X/\Lambda_{\text{ECP}_6}$ para cada técnica e para cada probabilidade de *bit-flip* ajustada, tendo como base os valores $\rho = 13\%$, $\rho = 15\%$ e $p = 50\%$.

P_X	$\Lambda_X/\Lambda_{\text{ECP}_6}$		
	$\rho = 13\%$	$\rho = 15\%$	$p = 50\%$
P_{ECP_6}	1,00	1,00	1,00
P_{DRM}	0,23	0,24	0,37
P_{SECDED}	0,27	0,30	0,47
$\min(P_{\text{SAFER}_{32}})$	0,96	0,96	1,02
$\mu(P_{\text{SAFER}_{32}})$	0,78	0,81	1,02
$\max(P_{\text{SAFER}_{32}})$	0,62	0,67	1,02
$P_{\text{FREE-p}}$	0,52	0,56	0,88

Analisando a Tabela 4.6, é possível notar que tanto a ECP_6 quanto a SAFER_{32} em uma PCM tem maiores benefícios nas questões de durabilidade e consumo de energia. Apesar de na tabela a técnica SAFER_{32} mostrar um desempenho melhor apenas em uma situação mais favorável (quando a probabilidade de *bit-flip* é o valor teórico de 50%) em relação à técnica ECP_6. Essas diferenças são pequenas e em uma análise mais profunda, considerando todo o sistema de memória, a situação pode se inverter; uma vez que dependerá de outros fatores como, por exemplo, o consumo de energia dos circuitos periféricos. Ademais, pela pouca diferença de Λ para os valores estatísticos de probabilidade de *bit-flip* obtidos, não é possível afirmar, cabalmente, que uma técnica é melhor do que outra.

4.2 Avaliação da técnica PAYG

Durante o desenvolvimento deste trabalho, Moinuddin K. Qureshi divulgou a técnica PAYG[1] [37], na qual reestrutura a organização das entradas ECP de uma memória, reduzindo-as em quantidade, mas melhorando a durabilidade em relação à organização original. Analisando o modelo analítico de falhas da técnica ECP_6 para uma memória de 1 GB, ele percebeu que menos do que 5% das linhas de memória utilizavam mais do que uma entrada ECP. Portanto, boa parte do *overhead* de *bits*, introduzidos pela técnica, não era utilizado.

Diferentemente da análise feita na Subseção 3.3.3, na qual o critério de parada do algoritmo de execução do modelo é estabelecido como quando a probabilidade de falha de um *bit* ($P(t)$) alcança o valor $\mu + 2 \cdot \sigma$, Qureshi adota como critério de parada para o modelo analítico o momento no qual a probabilidade de um sistema de memória falhar $\Omega(t)$, dado pela Equação 4.8, atinge 50%. Na verdade, a primeira falha que não puder

[1]Sigla para *Pay-As-You-Go*.

ser corrigida é o que provocará a falha do sistema, porém, probabilisticamente, isso não ocorre antes de $\Omega(t) = 50\%$.

$$\Omega(t) = 1 - (1 - \Psi(t))^{N_P} \tag{4.8}$$

Com base nos resultados desse modelo analítico, Qureshi propôs que toda linha da memória tivesse uma única entrada ECP e que um banco de entradas ECP, organizado como uma memória *cache*, fosse utilizado sob demanda à medida que mais do que dois erros acontecessem nas linhas de memória. Supondo, em seus modelos de simulação, que a primeira linha a falhar, sem poder ser recuperada por qualquer entrada ECP, colapsa a memória, obteve resultados semelhantes à técnica ECP_6, com um *overhead* de *bits* muito menor.

Na tentativa de construir o modelo de simulação da técnica PAYG de modo incremental, como feito no artigo, esbarrou-se em resultados aquém daqueles mostrados, logo nos estágios iniciais das implementações. Em busca de razões, fez-se uma análise criteriosa do modelo analítico da ECP, cedido pelo próprio Qureshi. Observou-se, então, que o número de linhas de memória que usam determinado número de entradas ECP varia com o tamanho da memória. Dessa forma, as simulações realizadas nunca alcançariam os resultados obtidos no artigo, pois não se havia simulado uma memória de tamanho igual àquele utilizado. Na verdade, não havia infraestrutura suficiente para simular uma memória de 1 GB.

Como mostrado na Figura 4.7, nota-se que quanto maior o tamanho da memória, maior é o número de linhas que, ou utilizaram somente uma entrada ECP, ou não utilizaram entrada alguma. Isto é, memórias de alta capacidade possui muitas linhas e a probabilidade de haver algumas dessas linhas com várias células com pouco tempo de vida é alta. Embora o respaldo da técnica PAYG seja essa situação, sua aplicação acaba ficando limitada à *chips* de alta capacidade e, além disso, o critério de que a primeira linha a falhar colapse a memória é muito rígido para memória com milhões de linhas ou mais. Dessa forma, faz-se necessário um mecanismo que permita o funcionamento da memória, a despeito dessas falhas, até um limite mais crítico como, por exemplo, mais do que metade da memória esteja comprometida.

4.3 Síntese

A execução instrumentada do SPEC CPU2006 permitiu a obtenção da taxa experimental de *bit-flip*, cujo valor é corroborado com as taxas de *bit-flip* divulgadas em alguns trabalhos da literatura; o que evidencia que a taxa de 50% de *bit-flip* pode não ser verossímil em programas reais. Situação que é problemática quando se leva em conta a modelagem realizada no Capítulo 3, visto que os resultados divulgados na literatura não se sustentam.

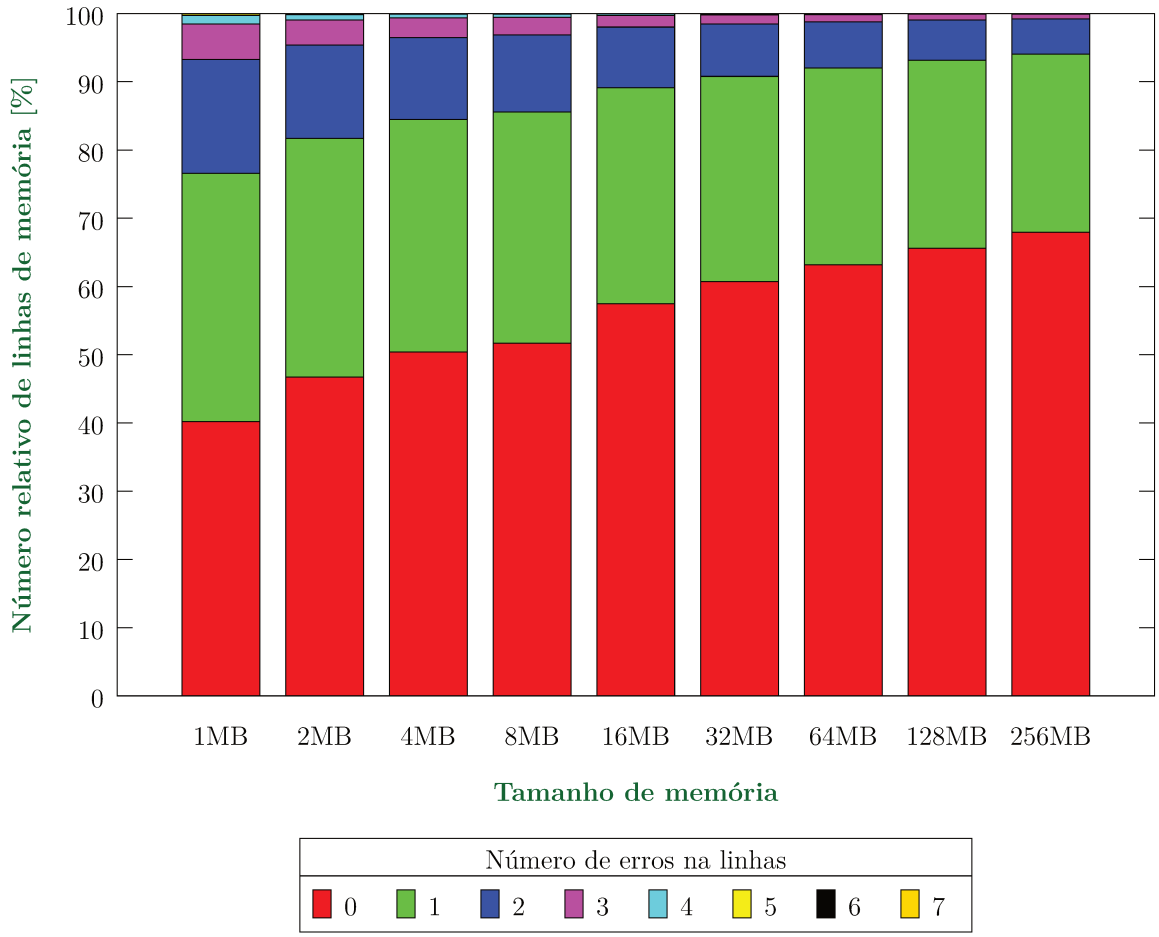


Figura 4.7: Avaliação dos resultados do modelo analítico para a técnica ECP_6 por meio de 50 execuções de simulação, utilizando o critério de parada do qual a primeira linha que não puder ser corrigida termina a simulação. Simulação configurada com $N = n = 512$, $\Delta = e = 6$, $N_L = 1$, $\mu = 2^{25}$, $\sigma = 0,2 \cdot \mu$. N_P pode ser calculado pelo tamanho da memória em *bytes* divido pelo tamanho da linha em *bytes*.

Através de Λ foi possível relacionar durabilidade e consumo de energia. Embora seja evidente que quanto maior for a durabilidade, maior será o número de escritas e, consequentemente, maior o gasto energético, entretanto, o fator Λ impede que uma técnica que tenha alta durabilidade e escritas que consumam muita energia se destaque perante uma que tenha uma durabilidade mediana e um baixo consumo de energia. Ou seja, é possível se fazer uma análise de compromissos (*trade offs*).

Finalmente, foi discutido uma precaução em relação à técnica PAYG ao ser mostrado que memórias menores que 1 GB terão desempenho inferior ao originalmente obtido e discutido no artigo.

Capítulo 5

Conclusões

5.1 Contribuições e avaliação do trabalho

Neste trabalho foi avaliado como a probabilidade de *bit-flip*, dos principais códigos corretores de erros e das principais técnicas de correção de erros, ECP, paridade, SECDED, SAFER e BCH, relaciona-se com o degaste de uma memória não-volátil. Com base nos modelos matemáticos de probabilidade *bit-flip* desenvolvidos, e se utilizando de modelos de simulação das técnicas de recuperação de erros, os resultados corroboraram com a hipótese de que códigos de correção de erros comportam-se dicotomicamente: permitindo que a memória se mantenha funcionando após erros permanentes, mas também acelerando o aparecimento deles. Notadamente, a técnica SAFER comporta-se de maneira similar aos códigos de correção de erros, à medida que erros surgem. Logo, a técnica ECP é a única que não se encaixou no papel duplo.

Ficou evidente que, para uma avaliação mais precisa de técnicas de correção de erros, faz-se necessário a discriminação entre dados e metainformação, principalmente da sua frequência de escritas, papel que neste trabalho foi desempenho pela probabilidade de *bit-flip*, por meio da ponderação das probabilidades de *bit-flip* dos dados e da metainformação das técnicas de correção de erros.

Com a execução do SPEC2006, obteve-se o valor estatístico da probabilidade de *bit-flip* de 15% para toda a coleção de programas que pertencem ao *suite*. Esse resultado experimental mostrou que as técnicas avaliadas podem ter uma durabilidade maior em aplicações reais, diferindo dos resultados divulgados na literatura que usam a probabilidade de *bit-flip* teórica de 50%. Mais do que isso, aplicando essa taxa aos modelos matemáticos de ajuste da probabilidade de *bit-flip*, chegou-se a resultados que levam as técnicas ECP e SAFER a se diferenciarem das demais.

Ao se estabelecer uma relação métrica entre durabilidade máxima (número máximo de escritas) e consumo energético para escrever uma linha de memória, as técnicas ECP e

SAFER se destacaram novamente sobre as demais. Sendo que, para as duas probabilidades de *bit-flip* experimentais (13% e 15%) utilizadas na comparação, a ECP foi quem teve a melhor relação entre durabilidade e consumo energético; contudo, as diferenças dos resultados da ECP e da SAFER foram pequenos. Considerando que os piores cenários de probabilidade de *bit-flip* (isto é, probabilidades maiores do que 50%) não se realizam experimentalmente para o conjunto de aplicações utilizadas, é possível tomar a SAFER ou a ECP como a mais eficiente técnica de correção de erros.

Além disso, vale ressaltar que foi o ajuste da probabilidade de *bit-flip* neste trabalho que permitiu avaliar o consumo de energia de uma escrita de memória para cada técnica de correção de erros. Nos trabalhos da literatura, como todas utilizam-se da probabilidade de *bit-flip* de 50% e o *overhead* de metainformação é praticamente idêntico para todas (12,5%), não há meios de distinguir quais técnicas têm escritas mais custosas (energeticamente). Portanto, a utilização dos modelos de probabilidade com a taxa de *bit-flip* obtida experimentalmente possibilitou essa análise de consumo de energia que faltou nos trabalhos estudados quando foram divulgados na literatura.

Para completar este trabalho, três modelos analíticos probabilísticos foram propostos, dois para a técnica ECP e um para o código SECDED. Provavelmente, duas das mais utilizadas formas de correção de erros na literatura relacionada. Para outras técnicas falhou-se na modelagem. Por fim, foi analisada a técnica PAYG cujos mecanismos e organização foram designados a partir da análise do modelo analítico da ECP. Sobre ela, observou-se uma dependência do tamanho da memória para apresentar bom resultados, além de um critério de durabilidade pouco realista para memórias muito grandes: a memória só durará enquanto linha alguma possuir mais erros do que a quantidade que pode ser corrigida em uma linha. Em milhões de linhas de memória, a probabilidade de uma linha assim aparecer é bem alta.

5.2 Trabalhos futuros

Existem diversos caminhos para além deste trabalho:

- **Modelagem da probabilidade de *bit-flip* para células de memória *multi-bits*.** A despeito de algumas técnicas não poderem ser avaliadas com a utilização de memórias MLC, é importante que esta avaliação seja feita, principalmente, porque memórias MLC são desejadas comercialmente e, claramente, precisam deste suporte a falhas fornecido pelas técnicas de correção de erros. Um detalhe importante, é que a modelagem para MLC não pode ser baseada na distribuição binomial, muito provavelmente, na distribuição multinomial.

- **Ampliação da população de programas testados.** Utilizar mais *benchmarks* e

mais diversificados, para aprimorar o valor estatístico da probabilidade de *bit-flip*.

- **Avaliação estatística dos resultados dos modelos de simulação.** Com um tratamento estatístico correto, avaliar erros obtidos na simulação de cada técnica, correlações entre variáveis, etc.

- **Ampliação dos modelos analíticos.** Os modelos podem acelerar e muito comparações e avaliações de trabalhos futuros que usem as técnicas aqui avaliadas. Além disso, os modelos abrem perspectivas de análise que, porventura, uma simulação não proporciona, permitindo que técnicas como a PAYG sejam desenvolvidas.

- **Implementação completa da técnica PAYG.** Terminar a implementação parcial que foi interrompida, por conta dos resultados inicias parecerem errados. A partir do modelo de simulação completo é possível se pensar em melhorias e adaptações que tornem a técnica eficiente, para além da primeira linha falha e para memórias de menor capacidade.

- **Próximas técnicas de correção de erros.** Embasando-se na proposta deste trabalho, espera-se que próximas técnicas de correção de erros para PCM, avaliem qual é o desgaste por elas proporcionado e a eficiência em termos energéticos.

Referências Bibliográficas

[1] S.J. Ahn, Y.J. Song, C.W. Jeong, J.M. Shin, Y. Fai, Y.N. Hwang, S.H. Lee, K.C. Ryoo, S.Y. Lee, J.H. Park, H. Horii, Y.H. Ha, J.H. Yi, B.J. Kuh, G.H. Koh, G.T. Jeong, H.S. Jeong, Kinam Kim, and B.I. Ryu. Highly manufacturable high density phase change memory of 64mb and beyond. In *Electron Devices Meeting, 2004. IEDM Technical Digest. IEEE International*, pages 907–910, dec. 2004.

[2] Gary Anthes. Memristors: pass or fail? *Commun. ACM*, 54(3):22–24, March 2011.

[3] G. Atwood and R. Bez. Current status of chalcogenide phase change memory. In *Device Research Conference Digest, 2005. DRC '05. 63rd*, volume 1, pages 29–33, june 2005.

[4] M. Awasthi, M. Shevgoor, K. Sudan, B. Rajendran, R. Balasubramonian, and V. Srinivasan. Efficient scrub mechanisms for error-prone emerging memories. In *High Performance Computer Architecture (HPCA), 2012 IEEE 18th International Symposium on*, pages 1–12, feb. 2012.

[5] Sylvie Barak. Intel, micron on sub 20-nm and insatiable thirst for memory. `http://www.eetimes.com/electronics-news/4371276/Intel--Micron-on-sub-20-nm-and-insatiable-thirst-for-memory`, 2012. Web Site. Accessed: 17-Mar-2013.

[6] F. Bedeschi, R. Bez, C. Boffino, E. Bonizzoni, E.C. Buda, G. Casagrande, L. Costa, M. Ferraro, R. Gastaldi, O. Khouri, F. Ottogalli, F. Pellizzer, A. Pirovano, C. Resta, G. Torelli, and M. Tosi. 4-mb mosfet-selected mu;trench phase-change memory experimental chip. *Solid-State Circuits, IEEE Journal of*, 40(7):1557–1565, july 2005.

[7] David Bondurant, Brand Engel, and Jon Slaughter. MRAM - The Future of Non-Volatile Memory? *Portable Design*, pages 16–21, 2008.

[8] Li-Pin Chang. On efficient wear leveling for large-scale flash-memory storage systems. In *Proceedings of the 2007 ACM symposium on Applied computing*, SAC '07, pages 1126–1130, New York, NY, USA, 2007. ACM.

[9] Chi-Hao Chen, Pi-Cheng Hsiu, Tei-Wei Kuo, Chia-Lin Yang, and Cheng-Yuan Michael Wang. Age-based pcm wear leveling with nearly zero search cost. In *Proceedings of the 49th Annual Design Automation Conference*, DAC '12, pages 453–458, New York, NY, USA, 2012. ACM.

[10] Jie Chen, Ron C. Chiang, H. Howie Huang, and Guru Venkataramani. Energy-aware writes to non-volatile main memory. *SIGOPS Oper. Syst. Rev.*, 45(3):48–52, January 2012.

[11] Sangyeun Cho and Hyunjin Lee. Flip-n-write: a simple deterministic technique to improve pram write performance, energy and endurance. In *Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO 42, pages 347–357, New York, NY, USA, 2009. ACM.

[12] Peter Clarke. Elpida claims it is first with 25-nm dram. `http://www.eetimes.com/electronics-news/4215633/Elpida-claims-it-is-first-with-25-nm-DRAM`, 2011. Web Site. Accessed: 17-Mar-2013.

[13] Peter Clarke. Asml says fast EUV machines coming by 2016. `http://www.eetimes.com/electronics-news/4390667/ASML-says-fast-EUV-machines-coming-by-2016`, 2012. Web Site. Accessed: 19-Mar-2013.

[14] Intel Corporation. Pin - A Dynamic Binary Instrumentation Tool. `http://www.pintool.org/`, 2012. Web Site. Accessed: 28-Jun-2012.

[15] Standard Performance Evaluation Corporation. SPEC CPU2006:Read Me First. `http://www.spec.org/cpu2006/docs/readme1st.html`, 2011. Web Site. Accessed: 22-Apr-2012.

[16] Saptarshi Das and Joerg Appenzeller. Fetram. an organic ferroelectric material based novel random access memory cell. *Nano Letters*, 11(9):4003–4007, 2011.

[17] J.L. Devore. *Probability & Statistics for Engineering and the Sciences*. Brooks/Cole, 6th edition, 2004.

[18] Gaurav Dhiman, Raid Ayoub, and Tajana Rosing. Pdram: a hybrid pram and dram main memory system. In *Proceedings of the 46th Annual Design Automation Conference*, DAC '09, pages 664–469, New York, NY, USA, 2009. ACM.

[19] Xiangyu Dong, Cong Xu, Yuan Xie, and N.P. Jouppi. Nvsim: A circuit-level performance, energy, and area model for emerging nonvolatile memory. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 31(7):994 –1007, july 2012.

[20] Andrew Hay, Karin Strauss, Timothy Sherwood, Gabriel H. Loh, and Doug Burger. Preventing pcm banks from seizing too much power. In *Proceedings of the 44th Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO-44 '11, pages 186–195, New York, NY, USA, 2011. ACM.

[21] Yenpo Ho, G.M. Huang, and Peng Li. Nonvolatile memristor memory: Device characteristics and design implications. In *Computer-Aided Design - Digest of Technical Papers, 2009. ICCAD 2009. IEEE/ACM International Conference on*, pages 485–490, 2009.

[22] Caio Hoffman. Infrastructure and analytical models. `http://masters.caiohoffman.org`, 2013. Web Site. Accessed: 05-Mar-2013.

[23] Engin Ipek, Jeremy Condit, Edmund B. Nightingale, Doug Burger, and Thomas Moscibroda. Dynamically replicated memory: building reliable systems from nanoscale resistive memories. In *Proceedings of the fifteenth edition of ASPLOS on Architectural support for programming languages and operating systems*, ASPLOS '10, pages 3–14, New York, NY, USA, 2010. ACM.

[24] Bruce Jacob, Spencer Ng, and David Wang. *Memory Systems: Cache, DRAM, Disk*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2007.

[25] Lei Jiang, Bo Zhao, Youtao Zhang, Jun Yang, and B.R. Childers. Improving write operations in mlc phase change memory. In *High Performance Computer Architecture (HPCA), 2012 IEEE 18th International Symposium on*, pages 1–10, feb. 2012.

[26] et al J.M. Slaughter, N.D. Rizzo. St-mram gets practical. `http://www.eetimes.com/design/memory-design/4411560/ST-MRAM-gets-practical-`, 2013. Web Site. Accessed: 13-Abr-2013.

[27] M.J. Kang, T. J. Park, Y. W. Kwon, D.H. Ahn, Y.S. Kang, H. Jeong, S.J. Ahn, Y.J. Song, B.C. Kim, S.W. Nam, H. K Kang, G.T. Jeong, and C.H. Chung. Pram cell technology and characterization in 20nm node size. In *Electron Devices Meeting (IEDM), 2011 IEEE International*, pages 3.1.1–3.1.4, 2011.

[28] Sangbeom Kang, WooYeong Cho, Beak-Hyung Cho, Kwang-Jin Lee, Chang-Soo Lee, Hyung-Rock Oh, Byung-Gil Choi, Qi Wang, Hye-Jin Kim, Mu-Hui Park, Yu-Hwan Ro, Suyeon Kim, Du-Eung Kim, Kang-Sik Cho, Choong-Duk Ha, Youngran Kim, Ki-Sung Kim, Choong-Ryeol Hwang, Choong-Keun Kwak, Hyun-Geun Byun, and Yun Sueng Shin. A 0.1-um 1.8-v 256-mb phase-change random access memory (pram) with 66-mhz synchronous burst-read operation. In *Solid-State Circuits Conference,*

2006. ISSCC 2006. Digest of Technical Papers. IEEE International, pages 487 –496, feb. 2006.

[29] Benjamin C. Lee, Engin Ipek, Onur Mutlu, and Doug Burger. Architecting phase change memory as a scalable dram alternative. *SIGARCH Comput. Archit. News*, 37:2–13, June 2009.

[30] S.H. Lee, H.C. Park, M.S. Kim, H.W. Kim, M.R. Choi, H.G. Lee, J.W. Seo, S.C. Kim, S.G. Kim, S.B. Hong, S.Y. Lee, J.U. Lee, Y.S. Kim, K.S. Kim, J.I. Kim, M.Y. Lee, H.S. Shin, S.J. Chae, J.H. Song, H.S. Yoon, J.M. Oh, S.K. Min, H.M. Lee, K.R. Hong, J.T. Cheong, S.N. Park, J.C. Ku, H.S. Shin, Y.S. Sohn, S.K. Park, T.S. Kim, Y.K. Kim, K.W. Park, C.S. Han, H.W. Kim, W. Kim, H.J. Kim, K.S. Choi, J.H. Lee, and S.J. Hong. Highly productive pcram technology platform and full chip operation: Based on 4f2 (84nm pitch) cell scheme for 1 gb and beyond. In *Electron Devices Meeting (IEDM), 2011 IEEE International*, pages 3.3.1 –3.3.4, dec 2011.

[31] Shu Lin and Daniel J. Costello. *Error Control Coding, Second Edition*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 2004.

[32] E. H. Mckinney. Generalized birthday problem. *The American Mathematical Monthly*, 73(4):pp. 385–387, 1966.

[33] Vidyabhushan Mohan, Taniya Siddiqua, Sudhanva Gurumurthi, and Mircea R. Stan. How i learned to stop worrying and love flash endurance. In *Proceedings of the 2nd USENIX conference on Hot topics in storage and file systems*, HotStorage'10, pages 3–3, Berkeley, CA, USA, 2010. USENIX Association.

[34] Volnei Pedroni. *Digital Electronics and Design with VHDL*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2008.

[35] M.K. Qureshi, M.M. Franceschini, and L.A. Lastras-Montano. Improving read performance of phase change memories via write cancellation and write pausing. In *High Performance Computer Architecture (HPCA), 2010 IEEE 16th International Symposium on*, pages 1–11, Jan.

[36] M.K. Qureshi, A. Seznec, L.A. Lastras, and M.M. Franceschini. Practical and secure pcm systems by online detection of malicious write streams. In *High Performance Computer Architecture (HPCA), 2011 IEEE 17th International Symposium on*, pages 478–489, Feb.

[37] Moinuddin K. Qureshi. Pay-as-you-go: low-overhead hard-error correction for phase change memories. In *Proceedings of the 44th Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO-44 '11, pages 318–328. ACM, 2011.

[38] Moinuddin K. Qureshi, Michele M. Franceschini, Ashish Jagmohan, and Luis A. Lastras. Preset: improving performance of phase change memories by exploiting asymmetry in write times. *SIGARCH Comput. Archit. News*, 40(3):380–391, jun 2012.

[39] Moinuddin K. Qureshi, Sudhanva Gurumurthi, and Bipin Rajendran. *Phase Change Memory: From Devices to Systems*. Synthesis Lectures on Computer Architecture. Morgan & Claypool Publishers, 2011.

[40] Moinuddin K. Qureshi, John Karidis, Michele Franceschini, Vijayalakshmi Srinivasan, Luis Lastras, and Bulent Abali. Enhancing lifetime and security of pcm-based main memory with start-gap wear leveling. In *Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO 42, pages 14–23, New York, NY, USA, 2009. ACM.

[41] Moinuddin K. Qureshi, Vijayalakshmi Srinivasan, and Jude A. Rivers. Scalable high performance main memory system using phase-change memory technology. *SIGARCH Comput. Archit. News*, 37(3):24–33, June 2009.

[42] S. Raoux, G. W. Burr, M. J. Breitwisch, C. T. Rettner, Y.-C. Chen, R. M. Shelby, M. Salinga, D. Krebs, S.-H. Chen, H.-L. Lung, and C. H. Lam. Phase-change random access memory: a scalable technology. *IBM J. Res. Dev.*, 52:465–479, July 2008.

[43] Stuart Schechter, Gabriel H. Loh, Karin Straus, and Doug Burger. Use ecp, not ecc, for hard failures in resistive memories. *SIGARCH Comput. Archit. News*, 38:141–152, June 2010.

[44] Adel S. Sedra and Kenneth C. Smith. *Microelectronic Circuits Revised Edition*. Oxford University Press, Inc., New York, NY, USA, 5th edition, 2007.

[45] Nak Hee Seong, Dong Hyuk Woo, and Hsien-Hsin S. Lee. Security refresh: prevent malicious wear-out and increase durability for phase-change memory with dynamically randomized address mapping. *SIGARCH Comput. Archit. News*, 38(3):383–394, June 2010.

[46] Nak Hee Seong, Dong Hyuk Woo, Vijayalakshmi Srinivasan, Jude A. Rivers, and Hsien-Hsin S. Lee. Safer: Stuck-at-fault error recovery for memories. In *Proceedings of the 2010 43rd Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO '43, pages 115–124, Washington, DC, USA, 2010. IEEE Computer Society.

[47] Andre Seznec. A phase change memory as a secure main memory. *IEEE Comput. Archit. Lett.*, 9(1):5–8, January 2010.

[48] International Technology Roadmap for Semiconductor. Emerging research devices - 2011 Edition. http://www.itrs.net/Links/2011ITRS/2011Chapters/2011ERD.pdf, 2011. Web Site. Accessed: 13-Abr-2013.

[49] International Technology Roadmap for Semiconductor. Factory integration - 2011 Edition. http://www.itrs.net/Links/2011ITRS/2011Chapters/2011Factory.pdf, 2011. Web Site. Accessed: 17-Mar-2013.

[50] International Technology Roadmap for Semiconductor. Lithography - 2011 Edition. http://www.itrs.net/Links/2011ITRS/2011Chapters/2011Lithography.pdf, 2011. Web Site. Accessed: 17-Mar-2013.

[51] International Technology Roadmap for Semiconductor. Process integration, devices, and structures - 2011 Edition. http://www.itrs.net/Links/2011ITRS/2011Chapters/2011PIDS.pdf, 2011. Web Site. Accessed: 17-Mar-2013.

[52] Hank Wallace. Error Detection and Correction Using the BCH Code. http://www.aqdi.com/bch.pdf, 2001. Web Site. Accessed: 11-Feb-2013.

[53] Neil Weste and David Harris. *CMOS VLSI Design: A Circuits and Systems Perspective*. Addison-Wesley Publishing Company, USA, 3rd edition, 2005.

[54] Wikipedia. BCH code. http://en.wikipedia.org/wiki/BCH_code, 2013. Web Site. Accessed: 11-Feb-2013.

[55] Wikipedia. Ferroelectricity. http://en.wikipedia.org/wiki/Ferroelectric_material, 2013. Web Site. Accessed: 11-Abr-2013.

[56] Wikipedia. Hamming code. http://en.wikipedia.org/wiki/Hamming_code, 2013. Web Site. Accessed: 27-Jan-2013.

[57] Wikipedia. Spin-transfer torque. http://en.wikipedia.org/wiki/Spin-transfer_torque, 2013. Web Site. Accessed: 13-Abr-2013.

[58] Wei Xu, Jibang Liu, and Tong Zhang. Data manipulation techniques to reduce phase change memory write energy. In *Proceedings of the 14th ACM/IEEE international symposium on Low power electronics and design*, ISLPED '09, pages 237–242, New York, NY, USA, 2009. ACM.

[59] J. Joshua Yang, Dmitri B. Strukov, and Duncan R. Stewart. Memristive devices for computing. *Nature Nanotechnology*, 8(1):13–24, 2012.

[60] Doe Hyun Yoon, N. Muralimanohar, Jichuan Chang, P. Ranganathan, N.P. Jouppi, and M. Erez. Free-p: Protecting non-volatile memory against both hard and soft errors. In *High Performance Computer Architecture (HPCA), 2011 IEEE 17th International Symposium on*, pages 466–477, feb. 2011.

[61] Ping Zhou, Bo Zhao, Jun Yang, and Youtao Zhang. A durable and energy efficient main memory using phase change memory technology. *SIGARCH Comput. Archit. News*, 37(3):14–23, June 2009.

Apêndice A

Exemplos de Códigos de Hamming

Tabela A.1: Cobertura do código de Hamming(31,26), adaptado de [56]. Na primeira linha estão discriminadas as posições na palavra-código, começando do menos significativo para o mais significativo. A segunda linha mostra a disposição dos *bits* na palavra-código, d indica um *bit* de dados, h_j indicia o *bit* de paridade que cobre todos os *bits* cujas posições (em binário) possuem o $([\log_2 j] + 1)$-ésimo *bit* igual à 1.

Posição do *bit* na palavra		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
bits na palavra-código		h_1	h_2	d_1	h_4	d_2	d_3	d_4	h_8	d_5	d_6	d_7	d_8	d_9	d_{10}	d_{11}	h_{16}	d_{12}	d_{13}	d_{14}	d_{15}	d_{16}	d_{17}	d_{18}	d_{19}	d_{20}	d_{21}	d_{22}	d_{23}	d_{24}	d_{25}	d_{26}
Cobertura dos *bits* de paridade	h_1	•		•		•		•		•		•		•		•		•		•		•		•		•		•		•	
	h_2		•	•			•	•			•	•			•	•			•	•			•	•			•	•			•	•
	h_4				•	•	•	•					•	•	•	•					•	•	•	•					•	•	•	•
	h_8								•	•	•	•	•	•	•	•									•	•	•	•	•	•	•	•
	h_{16}																•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•

Tabela A.2: Cobertura do código de Hamming(71,64). Na primeira linha estão discriminadas as posições na palavra-código, começando do menos significativo para o mais significativo. A segunda linha mostra a disposição dos *bits* na palavra-código, d indica um *bit* de dados, h_j indicia o *bit* de paridade que cobre todos os *bits* cujas posições (em binário) possuem o $([\log_2 j] + 1)$-ésimo *bit* igual à 1.

Posição do *bit* na palavra		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
bits na palavra-código		h_1	h_2	d_1	h_4	d_2	d_3	d_4	h_8	d_5	d_6	d_7	d_8	d_9	d_{10}	d_{11}	h_{16}	d_{12}	d_{13}	d_{14}	d_{15}	d_{16}	d_{17}	d_{18}	d_{19}	d_{20}	d_{21}	d_{22}	d_{23}	d_{24}	d_{25}	d_{26}
Cobertura dos *bits* de paridade	h_1	•		•		•		•		•		•		•		•		•		•		•		•		•		•		•	
	h_2		•	•			•	•			•	•			•	•			•	•			•	•			•	•			•	•
	h_4				•	•	•	•					•	•	•	•					•	•	•	•					•	•	•	•
	h_8								•	•	•	•	•	•	•	•									•	•	•	•	•	•	•	•
	h_{16}																•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•
	h_{32}																															
	h_{64}																															

32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71
h_{32}	d_{27}	d_{28}	d_{29}	d_{30}	d_{31}	d_{32}	d_{33}	d_{34}	d_{35}	d_{36}	d_{37}	d_{38}	d_{39}	d_{40}	d_{41}	d_{42}	d_{43}	d_{44}	d_{45}	d_{46}	d_{47}	d_{48}	d_{49}	d_{50}	d_{51}	d_{52}	d_{53}	d_{54}	d_{55}	d_{56}	d_{57}	h_{64}	d_{58}	d_{59}	d_{60}	d_{61}	d_{62}	d_{63}	d_{64}
	•		•		•		•		•		•		•		•		•		•		•		•		•		•		•		•		•		•		•	
		•	•			•	•			•	•			•	•			•	•			•	•			•	•			•	•			•	•			•	•
				•	•	•	•					•	•	•	•					•	•	•	•					•	•	•	•					•	•	•	•
								•	•	•	•	•	•	•	•							•	•	•	•	•	•	•	•	•	•								
																•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•								
•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•								
																																•	•	•	•	•	•	•	•

Apêndice B

Demonstrações

Primeiramente, apresenta-se algumas propriedades e definições.

Propriedade B.1

$$\binom{n}{k} = \binom{n-1}{k} + \binom{n-1}{k-1}$$

Relação de Stifel.

Propriedade B.2

$$\sum_{k=0}^{n} \binom{n}{k} = 2^n$$

Demonstração. Do binômio de Newton tem-se

$$(x+y)^n = \sum_{k=0}^{n} \binom{n}{k} x^{n-k} y^k.$$

Assim,

$$(1+1)^n = \sum_{k=0}^{n} \binom{n}{k} 1^{n-k} 1^k = \sum_{k=0}^{n} \binom{n}{k} = 2^n$$

□

Definição B.1 *$a \oplus b$ é definido como operação ou-exclusivo entre a e b.*

Definição B.2 *$\bigoplus_{a \in A} a$ é definido como a operação ou-exclusivo entre todos os elementos do conjunto A. Isto é, $\bigoplus_{a \in A} a = \bigoplus_{i=1}^{|A|} a_i = a_1 \oplus a_2 \oplus \cdots \oplus a_{|A|},\ a_i \in A$.*

Definição B.3 *Seja B um conjunto n de bits b_i, $0 \leq i < n$ e $b_i \in \{0, 1\}$. Seja $\mathcal{P}(B)$ o conjunto potência de B e seja $P \in \mathcal{P}(B)$. Define-se função $\oplus$-Potência*

$$f_{\oplus}(P) = \begin{cases} 0, & \textit{se } P = \{\} \\ \bigoplus_{a \in P} a, & \textit{caso contrário.} \end{cases}$$

Quer-se demonstrar o seguinte enunciado:

Proposição B.1 *Seja B um conjunto n de bits b_i, $0 \leq i < n$ e $b_i = \{0, 1\}$, com $k \geq 1$ bits 1. Seja $\mathcal{P}(B)$ o conjunto potência de B e seja $f_{\oplus}$ a função $\oplus$-Potência. Então, a soma dos valores resultantes da aplicação de $f_{\oplus}$ a todos elementos do conjunto $\mathcal{P}(B)$ é metade da cardinalidade do conjunto potência. Isto é,*

$$\sum_{P \in \mathcal{P}(B)} f_{\oplus}(P) = \frac{|\mathcal{P}(B)|}{2} = 2^{n-1}$$

Demonstração. Seja $\pi_1(B)$ o subconjunto de B que contém todos os k *bits* b_i de B que representam valor 1 em B. Seja π_0 o subconjunto de B que contém os $n - k$ *bits* B que representam valor 0.

Seja $P_i \in \mathcal{P}(B),\ 0 \leq i < |\mathcal{P}(B)|$. Sem perda de generalidade, assume-se $P_0 = \{\}$, $P_1 = \{b_0\}$, $P_2 = \{b_1\}$, $\ldots$, $P_n = \{b_{n-1}\}$, $P_{n+1} = \{b_0, b_1\}$, $\ldots$, $P_{2^n-1} = \{b_0, b_1, b_2, \ldots, b_{n-1}\}$. Temos que $f_{\oplus}(P_0) = 0$, $f_{\oplus}(P_1) = b_0$, $\ldots$, $f_{\oplus}(P_{n+1}) = b_0 \oplus b_1$, $\ldots$, $f_{\oplus}(P_{2^n-1}) = b_0 \oplus b_1 \oplus \cdots \oplus b_{n-1}$.

Entre $f_{\oplus}(P_1)$, $f_{\oplus}(P_2)$, ..., $f_{\oplus}(P_n)$, sabe-se que k (ou $\binom{k}{1}$) dessas operações resultam em 1. Entre $f_{\oplus}(P_{n+1})$ e $f_{\oplus}(P_{n+\binom{n}{2}})$, isto é, entre os valores resultantes aplicando $f_{\oplus}$ nos subconjuntos de $\mathcal{P}(B)$ que são combinações dois a dois dos elementos de B, os valores que resultam em 1 proveem dos subconjuntos formados por um único elemento de π_1 e um de π_0, pois é necessário um número ímpar de uns para que $f_{\oplus}$ seja 1. Assim, o número de resultados 1 é igual $\binom{k}{1} \cdot \binom{n-k}{1}$: de quantas formas pode-se selecionar um elemento de π_1 e um elemento de π_0.

Outrossim, o número de resultados 1 logrados da aplicação de $f_{\oplus}$ nos subconjuntos de $\mathcal{P}(B)$ formados por três elementos de B, é o mesmo da soma do número de subconjuntos que possuem um único elemento de π_1 e dois elementos de π_0 ou todos os três elementos contidos em π_1. Ou seja, $\binom{k}{1} \cdot \binom{n-k}{2} + \binom{k}{3}$.

Suponha que $k \equiv 1 (\text{mod } 2)$. A contagem dos resultados da aplicação de $f_{\oplus}$ nos elementos de $\mathcal{P}(B)$ progride conforme abaixo,

$$\begin{aligned} &\binom{k}{1} && + \\ &\binom{k}{1} \cdot \binom{n-k}{1} && + \end{aligned}$$

$$
\begin{array}{ll}
\binom{k}{1}\cdot\binom{n-k}{2}+\binom{k}{3} & + \\
\binom{k}{1}\cdot\binom{n-k}{3}+\binom{k}{3}\cdot\binom{n-k}{1} & + \\
\vdots & \\
\binom{k}{1}\cdot\binom{n-k}{n-k}+\binom{k}{3}\cdot\binom{n-k}{n-k-2}+\binom{k}{5}\cdot\binom{n-k}{n-k-4}+\cdots & + \\
\binom{k}{3}\cdot\binom{n-k}{n-k-1}+\binom{k}{5}\cdot\binom{n-k}{n-k-3}+\binom{k}{7}\cdot\binom{n-k}{n-k-5}+\cdots & + \\
\binom{k}{3}\cdot\binom{n-k}{n-k}+\binom{k}{5}\cdot\binom{n-k}{n-k-2}+\binom{k}{7}\cdot\binom{n-k}{n-k-4}+\cdots & + \\
\vdots & \\
\binom{k}{k-2}\cdot\binom{n-k}{n-k}+\binom{k}{k}\cdot\binom{n-k}{n-k-2} & + \\
\binom{k}{k}\cdot\binom{n-k}{n-k-1} & + \\
\binom{k}{k}\cdot\binom{n-k}{n-k} & +
\end{array}
$$

Percebe-se, pois, que é possível agrupar as somatória da seguinte maneira

$$
\begin{aligned}
&\binom{k}{1}\cdot\left[1+\binom{n-k}{1}+\binom{n-k}{2}+\cdots+\binom{n-k}{n-k}\right]+\binom{k}{3}\cdot\left[1+\binom{n-k}{1}+\binom{n-k}{2}+\right.\\
&\left.\cdots+\binom{n-k}{n-k}\right]+\cdots+\binom{k}{k}\cdot\left[1+\binom{n-k}{1}+\binom{n-k}{2}+\cdots+\binom{n-k}{n-k}\right],
\end{aligned}
$$

de outro modo,

$$
\begin{aligned}
&\binom{k}{1}\cdot\left[\binom{n-k}{0}+\binom{n-k}{1}+\binom{n-k}{2}+\cdots+\binom{n-k}{n-k}\right]+\binom{k}{3}\cdot\left[\binom{n-k}{0}+\binom{n-k}{1}+\right.\\
&\left.\binom{n-k}{2}+\cdots+\binom{n-k}{n-k}\right]+\cdots+\binom{k}{k}\cdot\left[\binom{n-k}{0}+\binom{n-k}{1}+\binom{n-k}{2}+\cdots+\binom{n-k}{n-k}\right].
\end{aligned}
$$

Pela Propriedade B.2,

$$
\binom{n-k}{0}+\binom{n-k}{1}+\binom{n-k}{2}+\cdots+\binom{n-k}{n-k}=\sum_{i=0}^{n-k}\binom{n-k}{i}=2^{n-k},
$$

logo, a soma pode ser descrita por

$$2^{n-k} \cdot \left[\binom{k}{1} + \binom{k}{3} + \cdots + \binom{k}{k} \right]. \tag{B.1}$$

Agora, utilizando a Propriedade B.1, substitui-se $\binom{k}{1}$ por $\binom{k-1}{0}+\binom{k-1}{1}$, $\binom{k}{3}$ por $\binom{k-1}{2}+\binom{k-1}{3}$, e assim consecutivamente até $\binom{k}{k-2}$ por $\binom{k-1}{k-3}+\binom{k-1}{k-2}$ e $\binom{k}{k}$ por $\binom{k-1}{k-1}$. Note que a última substituição mantém a igualdade, pois ambas valem 1. Finalmente, obtém-se

$$2^{n-k} \cdot \left[\binom{k-1}{0} + \binom{k-1}{1} + \binom{k-1}{2} + \binom{k-1}{3} + \cdots + \binom{k-1}{k-3} + \binom{k-1}{k-2} + \binom{k-1}{k-1} \right],$$

novamente, pela Propriedade B.2 tem-se que

$$\binom{k-1}{0} + \binom{k-1}{1} + \binom{k-1}{2} + \cdots + \binom{k-1}{k-1} = \sum_{i=0}^{k-1} \binom{k-1}{i} = 2^{k-1}.$$

Portanto, a Equação B.1 pode ser escrita como

$$2^{n-k} \cdot 2^{k-1} = 2^{n-1}.$$

Agora, supondo que $k \equiv 0 (\text{mod } 2)$. Isso implica que a aplicação de $f_{\oplus}$ no elemento de $\mathcal{P}(B)$ no qual se encontra todos os k *bits* de π_1, resultará 0. Dessa forma, a Equação B.1 para k par passa a não possuir o elemento $\binom{k}{k}$.

Assim,

$$2^{n-k} \cdot \left[\binom{k}{1} + \binom{k}{3} + \cdots + \binom{k}{k-1} \right]. \tag{B.2}$$

Novamente, utilizando a Propriedade B.1, substitui-se $\binom{k}{1}$ por $\binom{k-1}{0} + \binom{k-1}{1}$, $\binom{k}{3}$ por $\binom{k-1}{2} + \binom{k-1}{3}$, e assim em diante, até $\binom{k}{k-3}$ por $\binom{k-1}{k-4} + \binom{k-1}{k-3}$ e $\binom{k}{k-1}$ por $\binom{k-1}{k-2}$ e $\binom{k-1}{k-1}$. Logo,

$$2^{n-k} \cdot \left[\binom{k-1}{0} + \binom{k-1}{1} + \binom{k-1}{2} + \binom{k-1}{3} + \cdots + \binom{k-1}{k-3} + \binom{k-1}{k-2} + \binom{k-1}{k-1} \right].$$

Observe que o número de combinações somadas não se alteram com a paridade de k. Com k ímpar, acaba-se não aplicando a relação de Stifel no último elemento, pois $\binom{k}{k}$ teria de ser transformado em $\binom{k-1}{k-1} + \binom{k-1}{k}$ o que é impossível.

Finalmente, pela Propriedade B.1 a Equação B.2 pode ser escrita como

$$2^{n-k} \cdot 2^{k-1} = 2^{n-1}.$$

□

Corolário B.1.1 *Metade dos resultados $f_{\oplus}(P_i)$, $P_i \in \mathcal{P}(B),\ 0 \leq i < |\mathcal{P}(B)|$, são 1 e, consequentemente, metade são 0.*

Demonstração. Essa prova será feita por contradição, supondo que a quantidade de resultados 1, ou é maior, ou é menor, do que a quantidade de resultados 0. *A priori* suponha que seja maior. Sabe-se que em $\mathcal{P}(B)$ existem 2^n elementos e que a soma da aplicação de $f_{\oplus}$ em todos esses elementos resulta em 2^{n-1}. Se existem mais resultados 1 do que 0 na aplicação de $f_{\oplus}$ nos elementos P_i, então, a soma de uns tem de ser maior 2^{n-1}, mas isso não é possível. Igualmente, se houver um número menor de resultados 1, a soma deveria ter um valor menor do que 2^{n-1} o que, novamente, é impossível. Portanto, o número resultados 1 é igual ao número de resultados 0. □

Glossário de Símbolos

n

Denota o tamanho de um conjunto de *bits*. Podem ser uma palavra, uma linha, etc. 30

p

Probabilidade de *bit-flip*. 30

ρ

Valor estatístico da probabilidade de *bit-flip*. 30

τ

Taxa média de *bit-flips* de uma sequência de escritas. 32

μ

Tempo de vida médio das células de uma memória. 33

σ

Desvio padrão, ou coeficiente de variação, do tempo de vida das células de uma memória. 33

C_M

Capacidade total da memória em *bits*. 35

N

Tamanho de uma linha de memória. 35

N_L

Número de linhas de memória numa página. 35

N_P

Número de páginas numa memória. 35

F_P

Número de páginas falhadas numa memória. 36

ω

É passo da simulação seguinte ao qual a memória falha irrecuperavelmente. 36

Δ

Número de *bits* das metainformações das técnicas de correção de erros. 41

$\oplus$

Operação binária ou-exclusivo. 50

$f_{\oplus}$

Função para calcular o valor da operação ou-exclusivo em um conjunto de *bits*. 79

$P(t)$

Probabilidade de encontrar um *bit* falho com tempo de vida menor do t. 87

$\chi(t)$

Probabilidade de uma linha falhar dado um tempo de vida (ou número de escritas) t. 89

$\Psi(t)$

Probabilidade de uma página falhar dado um tempo de vida (ou número de escritas) t. 89

S_T

Número de passos do algoritmo de algum modelo analítico. 89

$\Omega(t)$

Probabilidade de um sistema de memória colapsar dado um tempo de vida (ou número de escritas) t. 110