Impact of memory approximation on energy efficiency

Isaías B. Felzmann*, João Fabrício Filho*†, Rodolfo Azevedo* and Lucas F. Wanner*

*Institute of Computing
University of Campinas
{isaias.felzmann, rodolfo, lucas}@ic.unicamp.br

†Federal University of Technology - Paraná
Campus Campo Mourão
joaof@utfpr.edu.br

***Abstract*—Approximate memories can lower energy consumption at expense of incurring errors in some of the read/write operations. While these errors may be tolerated in some cases, in general, parts of the application must be re-executed to achieve usable results when a large number of errors occur. Frequent re-executions may, in turn, attenuate or negate energy benefits obtained from using approximate memories. In this work, we show the energy impact of memory approximations in applications considering different quality requirements. Five out of nine selected applications showed a positive energy-quality tradeoff. For these applications, our results show up to 30% energy savings at a 10^{-8} error rate, when a 20% degradation in quality is allowed.**

***Index Terms*—Approximate Computing, Energy Efficiency, Memory approximation**

I. Introdução

Elementos de memória podem representar até 40% do consumo de energia em sistemas computacionais [1], [2]. O uso de técnicas de ajuste dinâmico de tensão (*Dynamic Voltage-Frequency Scaling - DVFS*) permite que memórias operem em regiões de maior eficiência energética [3]. Porém, a diminuição da tensão de alimentação resulta em menores margens de ruído estático e dinâmico e, consequentemente, em maior probabilidade de erros de leitura e escrita [4].

Tendo em vista que diversas aplicações apresentam alguma resiliência a erros, o paradigma de Computação Aproximada propõe a exploração mais agressiva das margens de tolerância em diversos níveis de projeto, incluindo o ajuste de tensão além da margem aceitável de ruído [5], [6]. Isso permite uma significativa redução na dissipação de potência de um circuito integrado, que é um fator limitante para o desenvolvimento de *hardware* [7], [8]. Por outro lado, os erros podem ter impacto maior do que o previsto na qualidade do resultado de qualquer aplicação executada, o que requer algum mecanismo de recuperação e amortiza os ganhos em eficiência energética [9], [10].

Neste trabalho, utilizamos simulação de arquiteturas computacionais baseada em ArchC [11], [12] para expor diferentes tipos de aplicações a memórias aproximadas. Assim buscamos o ponto de equilíbrio entre o ganho com economia de energia e a perda para recuperação de resultados. Em nosso sistema conceitual, a aplicação possui controle direto sobre parâmetros que definem a probabilidade de erros na leitura e escrita de dados. Além disso, o sistema implementa um mecanismo de controle de qualidade baseado em reexecução de aplicações cujas métricas de qualidade não atingem níveis aceitáveis.

Nossos experimentos demonstram a execução de 9 aplicações que representam atividades comuns em sistemas computacionais. Dessas aplicações, mostramos 5 pontos de inflexão no consumo de energia, considerando as reexecuções para readequação de qualidade. Nossos resultados mostram o perfil energético para cada aplicação e demonstram o equilíbrio entre qualidade e energia, obtendo até 30% de economia de energia para aplicações que exijam um mínimo de 80% de precisão nos resultados.

II. Fundamentos e trabalhos relacionados

A Computação Aproximada consiste em um padrão de computação com menor custo energético, comprometendo a precisão dos resultados obtidos [13]. Uma técnica conhecida para exploração de Computação Aproximada é o ajuste de tensão de alimentação além dos limites considerados seguros para manutenção de um resultado preciso, geralmente causando falhas temporais ou de chaveamento dos circuitos [5]. Em elementos de memória, a redução da tensão de alimentação provoca a redução das margens de ruído estático e dinâmico nas células de memória, resultando em uma maior probabilidade de erros de leitura e escrita ou na perda do dado armazenado [6], [14].

O modelo estatístico proposto por Wang e Calhoun [2011] associa, com base em valores iniciais experimentais, a tensão de alimentação com a probabilidade de ocorrência de erros, conforme a Figura 1(a). Considerando que energia possui relação quadrática com a tensão de alimentação, é possível prever o consumo energético após ajuste de tensão, relativo ao consumo de uma memória que garanta uma taxa de erro menor do que 10^{-12} (Fig. 1(b)). A análise em nível de circuito, porém, dificulta o estudo do impacto desses erros em memória no nível de aplicação. A alternativa é a modelagem dos erros em níveis de abstração mais altos, na forma de modificações

This work was executed with support from the National Council for Scientific and Technological Development - Brazil (CNPq) grant #404261/2016-7; Coordination for the Improvement of Higher Education Personnel - Brazil (CAPES) - Finance Code 001; and São Paulo Research Foundation (FAPESP) grant #2017/08015-8.

978-1-7281-3772-8/18/$31.00 ©2018 IEEE
DOI 10.1109/WSCAD.2018.00018

nas palavras de dados provenientes dos elementos de memória, e a utilização desses modelos em ambientes simulados.

EnerJ [15] propõe um controle explícito para aproximar tipos de dados em linguagem de programação. Esse trabalho mostrou que pequenas aproximações de dados podem inferir em um significativo potencial de economia de energia. Sua sucessão, ACCEPT [16], é um *framework* para programação aproximada guiado por anotações. As relaxações por aproximação desse trabalho demonstraram benefícios para os programas no desempenho e na utilização energética.

No trabalho em [17], os autores exploraram modificações em códigos de programas, visando otimizações para aplicações de *data centers*. Os autores relacionaram a qualidade das saídas de cada código gerado com a economia de energia resultante, mostrando que há menor consumo energético em aplicações com maior aceitabilidade de erro. DrSEUs [18] se volta para memórias, explorando possíveis falhas na cache de processadores e a classificação dos possíveis erros decorrentes dessas falhas, levando em consideração a resiliência da aplicação e a implicação em reexecuções, porém sem considerar as implicações em termos de energia. Rumba [19] revisita a ideia de reexecução com um mecanismo de verificação de partes da aplicação para erros mais altos, ao custo de um monitoramento em tempo de execução.

Neste trabalho, expandimos a cobertura de erros em memória dentro de diferentes tipos de aplicação, sem isolamento de estruturas. Para cada aplicação, apresentamos os limiares de ganho energético com base na qualidade das saídas e probabilidade necessidade de uma reexecução. Com base nos dados de Wang e Calhoun [2011] e considerando a possibilidade de reexecução, a Figura 1(c) demonstra a tendência esperada no consumo médio de energia de acordo com a taxa de erros de memória. No ponto inicial *A*, a taxa de erros é tão baixa que se compara a uma execução sem erros, com custo energético equivalente. A medida que a taxa aumenta, o consumo energético encaminha-se para o ponto de equilíbrio *B*, de maior economia, caracterizado pela necessidade de poucas reexecuções. A medida que mais reexecuções são necessárias, a energia consumida por execuções precisas adicionais passa a ser dominante, até o ponto máximo *C*, em que muitas execuções aproximadas completam, mas resultam em qualidade abaixo do requerido. Por fim, a ocorrência de falhas de execução começa a causar a conclusão prematura de algumas execuções, reduzindo o trabalho em execuções aproximadas e fazendo com que novamente o custo energético se aproxime ao custo de execuções precisas, no ponto *D*.

III. Condução dos experimentos

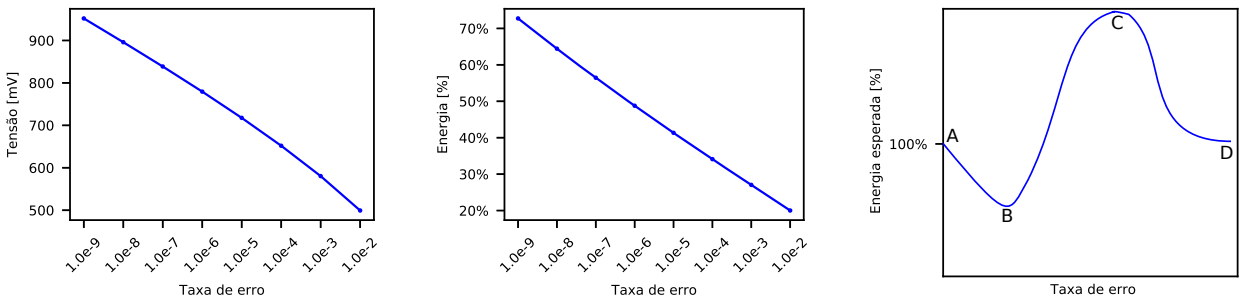
O modelo de arquitetura que consideramos neste trabalho é baseado em um processador de baixo consumo voltado para dispositivos embarcados. Nesse modelo conceitual, existe a implementação de um conjunto de "estados aproximados" capazes de influenciar diretamente a tensão de alimentação do banco de registradores e da memória de dados, deixando o sistema suscetível a erros de leitura e escrita [4]. O estado aproximado atual é definido por meio de um registrador mapeado em memória disponível para a aplicação.

Essa arquitetura foi representada em linguagem ArchC com base em um modelo do processador MIPS [11]. O modelo original foi estendido para a inclusão de aproximações utilizando a linguagem de descrição ADeLe, voltada à modelagem de Computação Aproximada [12]. Todas as operações de leitura e escrita no banco de registradores e na memória de dados foram substituídas por modelos em software de três tipos de erros:

- *BitFlip*: um dos bits do vetor de dados é sorteado e seu valor é invertido;
- *StuckAt(0)*: um dos bits do vetor de dados é sorteado e seu valor definido como 0;
- *StuckAt(1)*: análogo a *StuckAt(0)*, com o valor do bit definido como 1.

Devido a tais erros serem probabilísticos, qualquer operação de leitura ou escrita tem a mesma probabilidade de ser afetada. Foram selecionadas 9 diferentes aplicações para representar um conjunto de elementos comuns em dispositivos embarcados. Em cada aplicação, o código do *kernel* de execução foi isolado e apenas nele os erros são aplicados. Desse modo, as operações de entrada e saída, características de simulação, são executadas de forma precisa. As aplicações selecionadas, suas respectivas classificações e métricas de qualidade são:

- **Aplicações típicas:** Tipicamente, trabalhos relacionados a Computação Aproximada utilizam algoritmos de processamento multimídia para demonstração de resultados [20]. Assim, selecionamos o algoritmo de compressão de imagens JPEG disponível na suíte AxBench [21] e a computação de Transformada Rápida de Fourier (FFT) do MiBench [22]. As imagens JPEG computadas pela versão sujeita a erros foram comparadas com resultados precisos utilizando a métrica de Similaridade Estrutural [23], [24]. Para FFT foi computado o número de amostras fora de uma margem tolerância da ordem 10^{-9} no sinal após reconstrução com a Transformada Inversa.
- **Aplicações *CPU-bound*:** As aplicações Mandelbrot, N-Body e SpectralNorm foram selecionadas dentre as aplicações em [25]. Essas aplicações têm em comum um maior uso do tempo de CPU e um menor uso de memória, portanto potencialmente demonstrando maior resiliência para aproximações em memória. Os *bitmaps* gerados pelas versões sujeitas a erros na aplicação Mandelbrot foram comparados com as saídas geradas pela versão precisa utilizando a métrica de Similaridade Estrutural [23], [24]. As qualidades das saídas das execuções sujeitas a erros de N-Body e SpectralNorm foram calculadas pelo complemento do erro relativo médio.
- **Aplicações *Memory-Bound*:** Foram selecionados os algoritmos de Dijkstra, ordenação QSort, compressão de dados bzip e descompressão de dados bunzip a partir das suítes MiBench [22] e cBench [26]. Essas aplicações apresentam um maior uso da memória, e, portanto, são mais suscetíveis ao tipo de aproximação de dados



(a) (b) (c)

Fig. 1. Relação entre energia e erros em memórias: (a) Tensão de operação; (b) Energia; (c) Impacto esperado em uma aplicação.

explorada neste trabalho. A qualidade das saídas sujeitas a erros da aplicação QSort é medida conforme a fração de elementos iguais aos da ordenação correta. Para implementar a métrica de qualidade da aplicação Dijkstra, a saída foi modelada com base na construção de tabelas de roteamento, na qual cada dado na linha i e coluna j indica o próximo *hop* para chegar ao destino j estando na origem i. Dessa forma, a qualidade é a fração de próximos $hops$ corretos na tabela de saída. As aplicações bzip e bunzip foram utilizadas para comprimir e descomprimir arquivos-texto, analisando qualidade pela similaridade entre as cadeias de caracteres do conteúdo dos arquivos.

As execuções de cada aplicação foram repetidas 100 vezes, submetidas a variadas probabilidades de erros de leitura e escrita. As probabilidades foram definidas desde o ponto de ocorrência de 1 erro dentre todas as instruções no *kernel* na aplicação (conforme o número de instruções executado) até 1%. Para cada execução, foram analisadas a resiliência da aplicação – que é a probabilidade de ocorrência de uma falha que impeça o término da execução – e a qualidade final do resultado gerado.

Na arquitetura proposta, quando uma determinada execução não atinge um limiar mínimo de qualidade, o resultado deve ser reexecutado em modo preciso, com uma penalidade em energia. Assim, as métricas de qualidade e resiliência foram agregadas para obter a probabilidade de reexecução de uma aplicação em modo preciso e o consumo de energia total, considerando-se a execução aproximada e a precisa subsequente, quando houver. A energia necessária para execução foi derivada pelo método em [4].

IV. Resultados

Nossos resultados mostram a análise de resiliência das aplicações para 100 repetições de cada execução. Além disso, foram analisadas as métricas de qualidade apresentadas na Seção III e, a partir delas, computada a energia média de uma execução qualquer.

A. Análise de resiliência

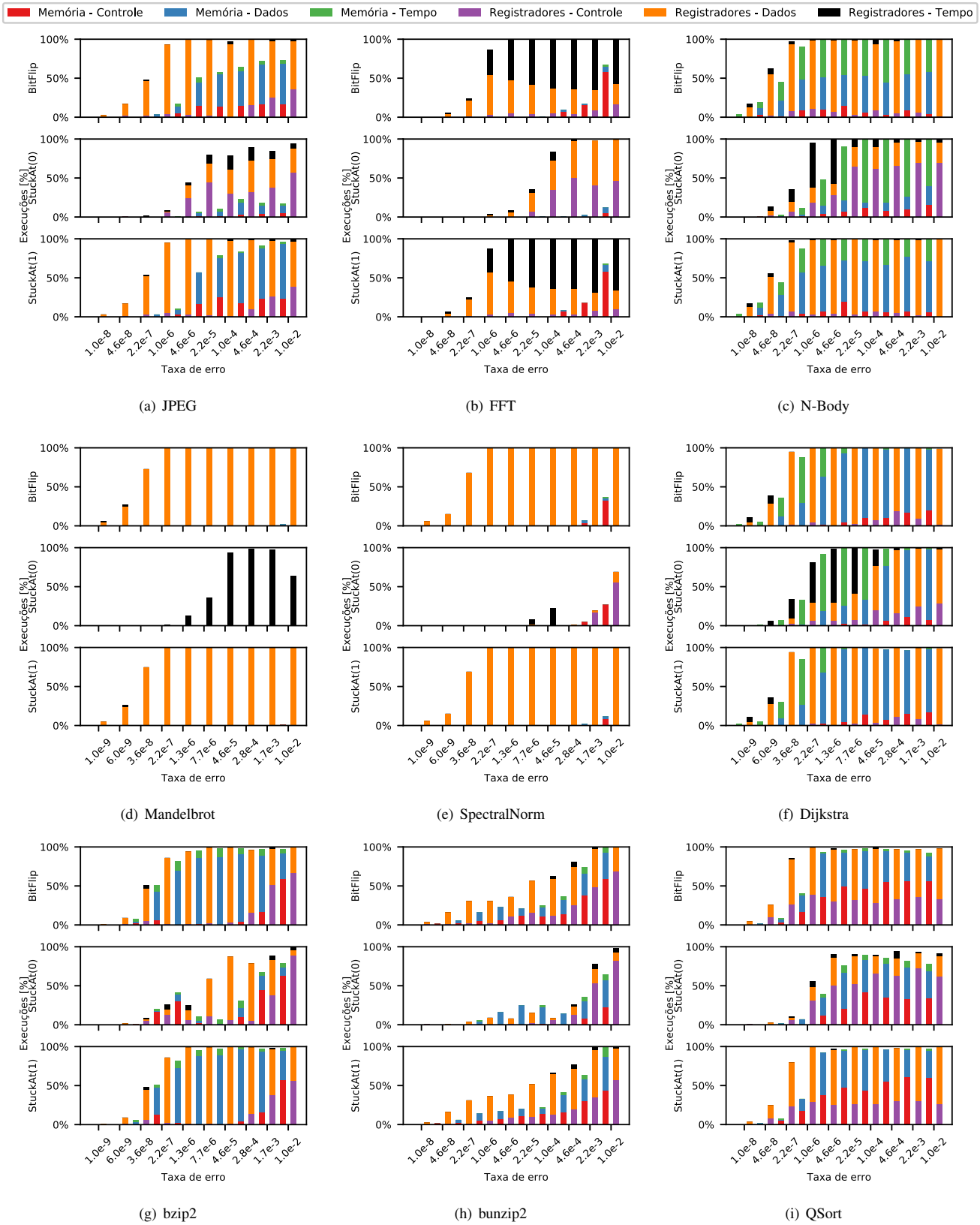
Para análise de resiliência, a ocorrência de uma falha significa que a execução foi interrompida antes da produção de um resultado válido. Tais falhas foram classificadas em três categorias:

- Falhas no fluxo de **controle**: ocorrem quando o endereço de destino de um salto é lido incorretamente, provocando um salto para um endereço inválido.
- Falhas no fluxo de **dados**: ocorrem quando um dado é buscado num endereço de memória inválido (*Segmentation Fault*).
- Falhas de **tempo**: ocorrem quando um resultado válido não é computado em tempo hábil. O tempo máximo que uma execução aproximada pode utilizar foi fixado em 5 vezes o tempo de uma computação precisa.

Os gráficos da Figura 2 demonstram a análise de resiliência das aplicações. De maneira geral, a suscetibilidade a erros no banco de registradores da arquitetura proposta apresenta drásticos efeitos no fluxo da aplicação. De fato, o banco de registradores armazena, além de valores locais, endereços de memória, variáveis de controle de laços e endereços de retorno de funções ou destino de saltos. Além disso, as falhas do tipo *Tempo*, em que um resultado não foi computado em tempo hábil, concentram-se principalmente em situações em que erros são aplicados aos registradores, demonstrando a baixa resiliência de estruturas de controle a aproximações.

Os erros do tipo *StuckAt(0)* mostram ser mais facilmente mascarados pelas aplicações em termos de falhas de execução. Esse tipo de erro, quando afeta endereços de memória, tem a tendência de alterar o dado para um endereço que ainda pertence ao programa executado, possivelmente na mesma página de memória, mitigando a ocorrência de falhas de dados e controle. Por outro lado, esse mesmo comportamento, quando aplicado a endereços de destino de saltos ou variáveis de controle de laços, aumenta o tempo de execução da aplicação, possivelmente criando laços infinitos, aumentando a ocorrência de falhas de tempo, especialmente naquelas baseadas em convergência, como N-Body (Fig. 2(c)) e Dijkstra (Fig. 2(f)).

A resiliência das aplicações é fator limitante do equilíbrio entre qualidade e energia, uma vez que uma falha de execução



(a) JPEG (b) FFT (c) N-Body

(d) Mandelbrot (e) SpectralNorm (f) Dijkstra

(g) bzip2 (h) bunzip2 (i) QSort

Fig. 2. Análise de resiliência.

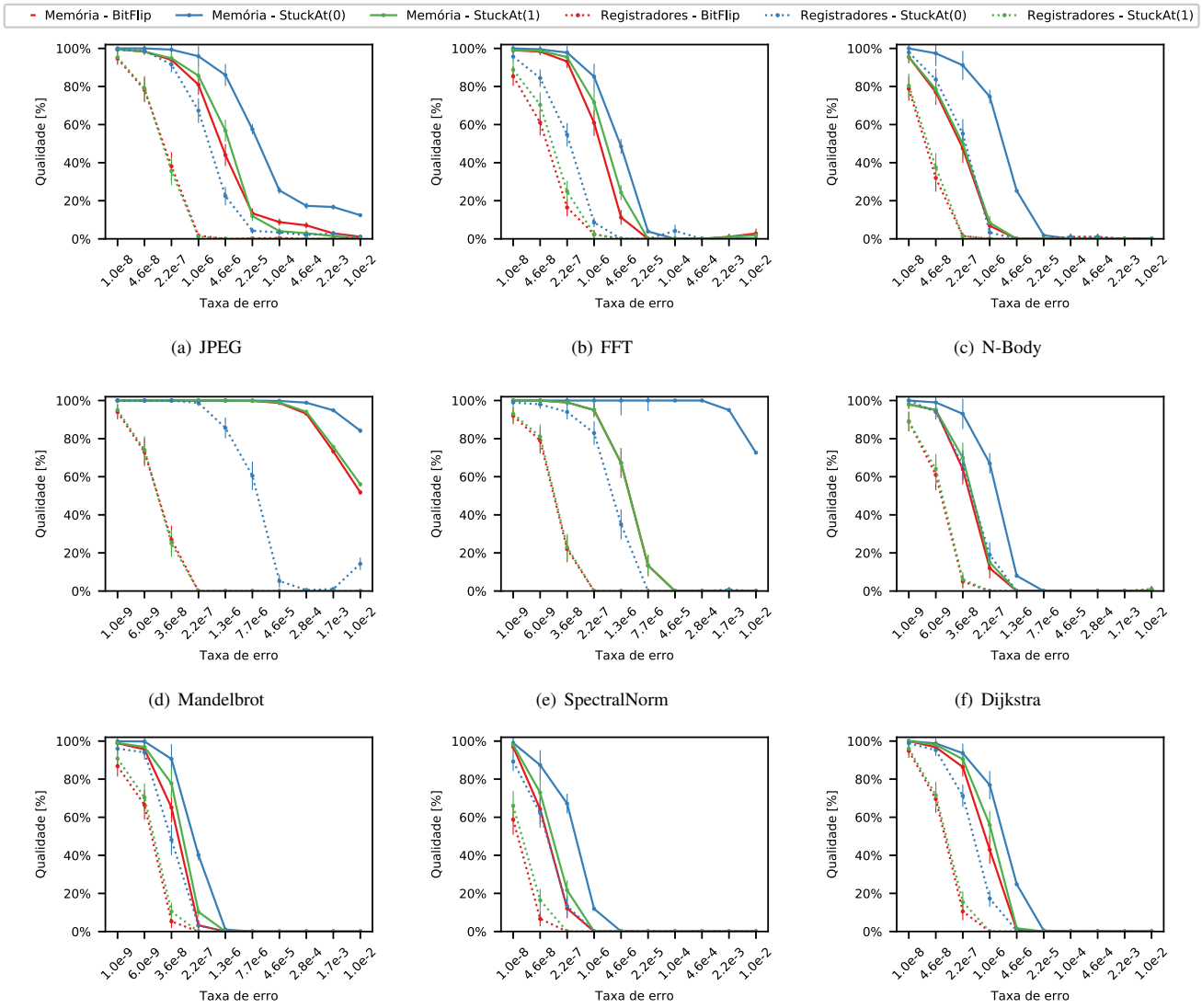


Fig. 3. Qualidade dos resultados.

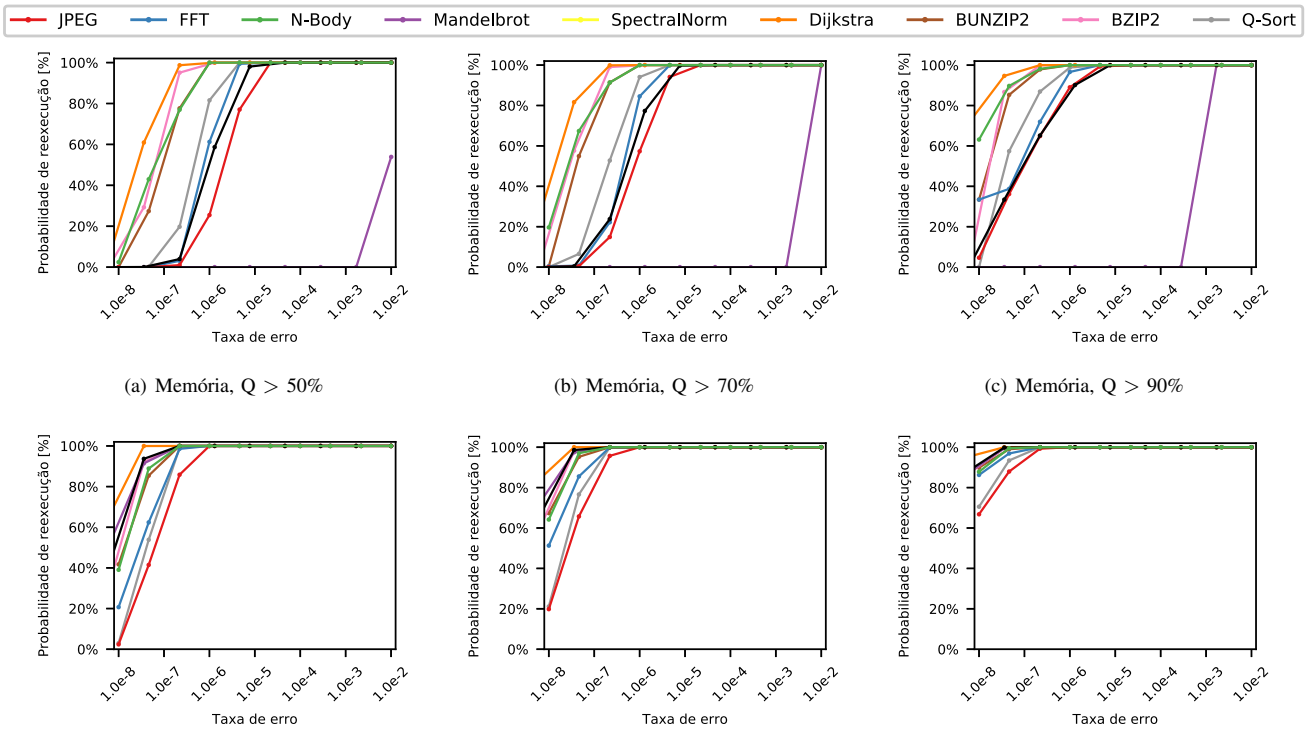
resulta em computação inútil e, consequentemente, desperdício de energia. Além disso, mesmo quando uma falha não ocorre, problemas de controle podem levar a um maior tempo de execução, com efeito negativo sobre a energia.

A maior parte das falhas decorrentes dos erros inseridos em memória são decorrentes do acesso a endereços inválidos. Esse tipo de falha pode ser mascarado com base no isolamento de áreas de memória para armazenamento de ponteiros e endereços alvo de saltos. Desta forma, as aplicações demonstrariam uma maior resiliência, já que os erros afetariam apenas os dados propriamente ditos, mas haveria a necessidade de se manter regiões de memória sempre em um modo não sujeito a falhas, o que impacta o custo energético das operações.

B. Qualidade dos resultados

Os gráficos da Figura 3 mostram a qualidade final média do resultado relativa a uma execução precisa da mesma aplicação. Para computação da qualidade, foi utilizada a média das métricas descritas na Seção III para 100 execuções, com um intervalo de confiança de 95%. As execuções que resultam em falha de execução foram computadas com qualidade zero, pois um resultado utilizável não foi obtido.

As falhas de execução são fatores dominantes na medida final de qualidade. O isolamento de estruturas de controle pode evitar falhas de execução, direcionando o impacto dos erros ao resultado final. A análise dos pontos iniciais dos gráficos da Figura 3 evidencia que o detrimento de qualidade é menos abrupto em execuções que completam com sucesso, o que



(a) Memória, Q > 50%

(b) Memória, Q > 70%

(c) Memória, Q > 90%

(d) Registradores, Q > 50%

(e) Registradores, Q > 70%

(f) Registradores, Q > 90%

Fig. 4. Probabilidade de reexecução.

indica uma possibilidade de maior eficiência energética.

Outro efeito da baixa resiliência das aplicações é o maior impacto na qualidade causado pelas aproximações em registradores. Ainda assim, desconsiderando-se as falhas, a aproximação de registradores provoca maior degradação de qualidade, pois como as operações ocorrem em maior número, são mais afetadas por erros. Isso indica uma desvantagem da técnica de aproximação quando aplicada a registradores.

C. Energia

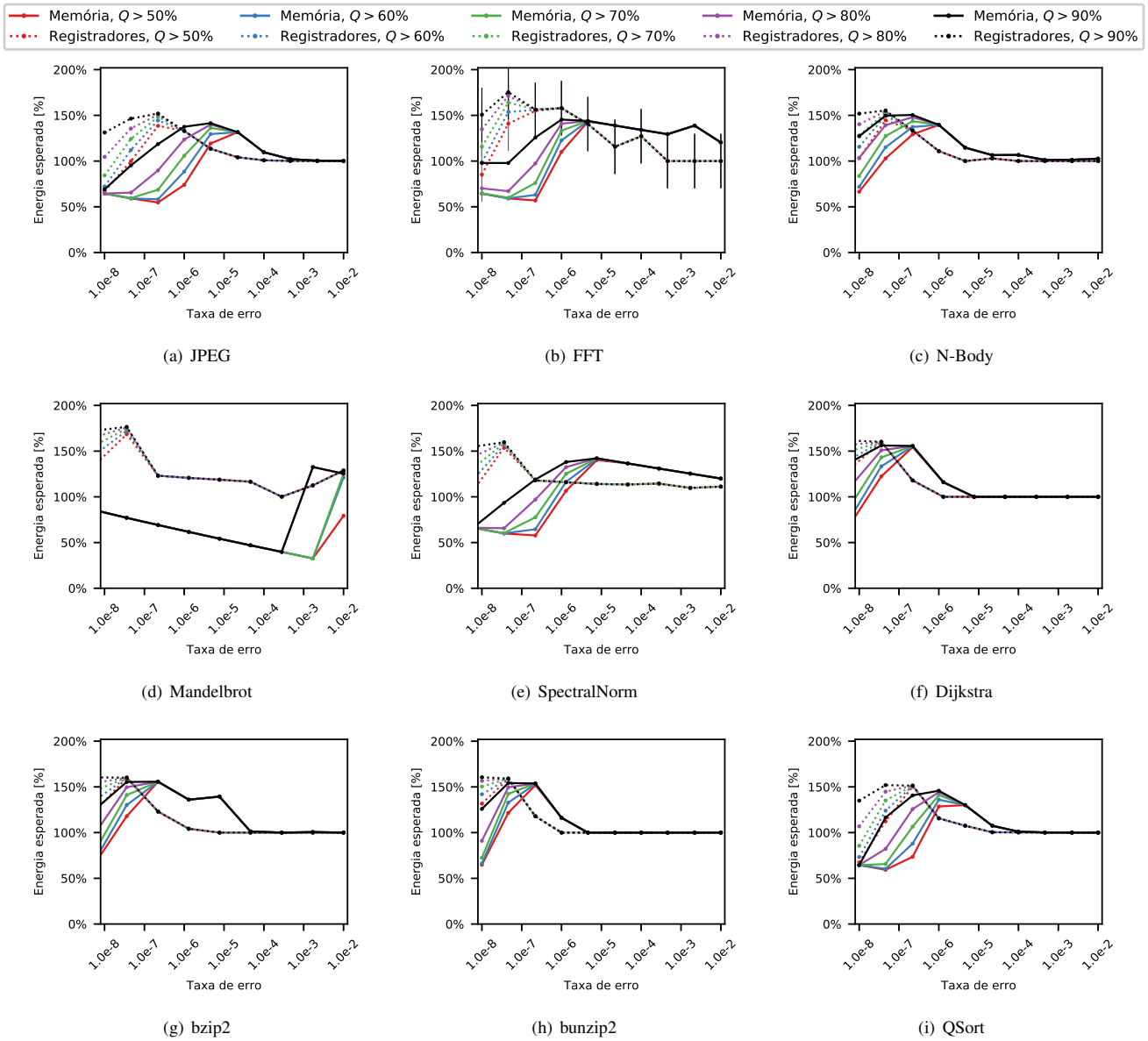
Embora o uso de estruturas de memória aproximadas ofereça um ganho em economia de potência no sistema, a ocorrência de falhas de execução e a degradação da qualidade do resultado podem requerer a reexecução de determinadas instâncias de uma aplicação. Quando uma reexecução em modo preciso é necessária, existe um impacto no consumo de energia. Com base na qualidade do resultado esperada para cada aplicação (Seção IV-B), calculamos a probabilidade de uma determinada execução resultar em qualidade abaixo do requerido e ter de ser reexecutada em modo preciso. A Figura 4 mostra essa probabilidade quando deseja-se uma métrica de qualidade maior que 50%, 70% e 90%, com erros aplicados a memória e registradores.

Uma reexecução em modo preciso resulta em uma penalidade em energia fixa para cada aplicação. Por outro lado, a energia de uma execução em modo aproximado depende do tempo de execução até a obtenção de um resultado ou a ocorrência de uma falha. Assim, a energia relativa esperada foi computada de acordo com a média do número de instruções executadas pela versão aproximada da aplicação e do número de instruções do modo preciso.

A Figura 5 relaciona a taxa de erros na execução de cada aplicação com o consumo médio esperado de energia, com um intervalo de confiança de 95%. De modo geral, todas as aplicações demonstram alguma economia de energia com a flexibilização do requisito de qualidade, quando considerados erros aplicados em memória. Para erros em registradores, por outro lado, a maior parte dos resultados sequer demonstra alguma economia de energia, prejudicada principalmente pela baixa resiliência das aplicações.

De modo particular, a inclusão de falhas em registradores para a aplicação FFT ocasiona uma grande variação no número de instruções executadas em modo aproximado entre diferentes repetições. Um comportamento semelhante também pode ser observado analisando-se o grande número de falhas de tempo para essa aplicação (Figura 2(b)). A estrutura de laços aninhados do algoritmo da Transformada de Fourier é bastante desfavorável a erros que afetem as estruturas de controle, como os aplicados a registradores. Para as demais aplicações, por outro lado, a variação no número de instruções em modo aproximado, embora cause variações na energia de cada execução, é facilmente mitigada perante várias repetições.

Os resultados para o consumo de energia demonstram um comportamento semelhante ao esperado (Figura 1(c)), em que



(a) JPEG (b) FFT (c) N-Body

(d) Mandelbrot (e) SpectralNorm (f) Dijkstra

(g) bzip2 (h) bunzip2 (i) QSort

Fig. 5. Equilíbrio entre qualidade e energia.

o consumo diminui com o aumento da taxa de erros até o ponto em que há um maior detrimento de qualidade, tornando a reexecução um fator dominante. Esse ponto de inflexão no consumo de energia – ponto *B* na Figura 1(c) – é visível para 5 das 9 aplicações testadas. As demais aplicações apresentam uma inclinação ascendente para os pontos iniciais, indicando que o ponto de equilíbrio existe, mesmo que menos acentuado, em uma taxa de erros menor. Taxas menores, por outro lado, caracterizam uma menor economia de energia, atenuando os efeitos da exploração de aproximações.

V. Conclusão

Apresentamos um estudo do efeito de erros em memória, caracteristicamente decorrentes da exploração de técnicas de Computação Aproximada, na execução de aplicações. Nosso estudo demonstra um ponto de equilíbrio entre a qualidade final do resultado e o custo energético da utilização de memórias em um sistema computacional. A economia de energia, embora mais acentuada em aplicações inerentemente mais resilientes a falhas, está também presente em aplicações não tradicionalmente associadas com Computação Aproximada. Assim, uma maior permissividade de erros em memória demonstrou-se vantajosa, evidenciando uma boa aplicabilidade das técnicas para o aumento da eficiência energética.

Apesar das vantagens, algumas aplicações só começam a demonstrar economia de energia para requisitos de qualidade mais flexíveis. A flexibilização da qualidade, dependendo da

aplicação, pode ser indesejável, de modo que é necessário o desenvolvimento de técnicas que melhorem a resiliência das execuções. Nossos experimentos mostram que as falhas de execução decorrentes de erros em memória, como saltos para endereços inválidos, são um fator determinante da qualidade final do resultado. Para melhorar a resiliência, seria possível isolar alguns dados, como variáveis intimamente ligadas com controle e endereços de memória, em regiões não sujeitas a falhas.

O isolamento de áreas de memória para armazenamento de dados que não admitem erros, aliado à exploração de aproximações em um conjunto de dados extenso, tem o potencial de deslocar o ponto de equilíbrio entre qualidade e energia para regiões energeticamente mais eficientes. Assim, é importante o desenvolvimento de ferramentas voltadas à identificação de regiões de dados verdadeiramente essenciais, para minimizar a ocorrência de falhas nas execuções. Como trabalhos futuros, planejamos o estudo da resiliência de diversas aplicações, isolando regiões de dados essenciais, por meio de modelos de programação ou arquitetura, com o objetivo de avaliar o equilíbrio entre qualidade e energia, como base para o desenvolvimento de uma ferramenta automatizada.

REFERENCES

[1] I. Paul, W. Huang, M. Arora, and S. Yalamanchili, "Harmonia: Balancing compute and memory power in high-performance gpus," in *Proceedings of the 42Nd Annual International Symposium on Computer Architecture*, ser. ISCA '15. New York, NY, USA: ACM, 2015, pp. 54–65.

[2] K. K. Chang, A. G. Yauglikcci, S. Ghose, A. Agrawal, N. Chatterjee, A. Kashyap, D. Lee, M. O'Connor, H. Hassan, and O. Mutlu, "Understanding reduced-voltage operation in modern dram devices: Experimental characterization, analysis, and mechanisms," *Proc. ACM Meas. Anal. Comput. Syst.*, vol. 1, no. 1, pp. 10:1–10:42, Jun. 2017.

[3] B. H. Calhoun and A. Chandrakasan, "Analyzing static noise margin for sub-threshold SRAM in 65nm CMOS," in *Proceedings of the 31st European Solid-State Circuits Conference (ESSCIRC)*, Sep. 2005, pp. 363–366.

[4] J. Wang and B. H. Calhoun, "Minimum supply voltage and yield estimation for large srams under parametric variations," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 19, no. 11, pp. 2120–2125, Nov 2011.

[5] V. K. Chippa, D. Mohapatra, K. Roy, S. T. Chakradhar, and A. Raghunathan, "Scalable effort hardware design," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 22, no. 9, pp. 2004–2016, Sep. 2014.

[6] M. Gottscho, A. BanaiyanMofrad, N. Dutt, A. Nicolau, and P. Gupta, "Power / capacity scaling: Energy savings with simple fault-tolerant caches," in *Proceedings of the 51st Annual Design Automation Conference*, ser. DAC '14. New York, NY, USA: ACM, 2014, pp. 100:1–100:6.

[7] H. Esmaeilzadeh, E. Blem, R. St. Amant, K. Sankaralingam, and D. Burger, "Dark silicon and the end of multicore scaling," in *Proceedings of the 38th Annual International Symposium on Computer Architecture*, ser. ISCA '11. New York, NY, USA: ACM, 2011, pp. 365–376.

[8] L. Kugler, "Is "good enough" computing good enough?" *Commun. ACM*, vol. 58, no. 5, pp. 12–14, Apr. 2015.

[9] S. Ganapathy, G. Karakonstantis, A. Teman, and A. Burg, "Mitigating the impact of faults in unreliable memories for error-resilient applications," in *Proceedings of the 52Nd Annual Design Automation Conference*, ser. DAC '15. New York, NY, USA: ACM, 2015, pp. 102:1–102:6.

[10] P. Gupta, Y. Agarwal, L. Dolecek, N. Dutt, R. K. Gupta, R. Kumar, S. Mitra, A. Nicolau, T. S. Rosing, M. B. Srivastava, S. Swanson, and D. Sylvester, "Underdesigned and opportunistic computing in presence of hardware variability," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 32, no. 1, pp. 8–23, Jan 2013.

[11] S. Rigo, G. Araujo, M. Bartholomeu, and R. Azevedo, "Archc: a systemc-based architecture description language," in *16th Symposium on Computer Architecture and High Performance Computing (SBAC-PAD)*, Oct 2004, pp. 66–73.

[12] I. B. Felzmann, M. M. Susin, L. Duenha, R. J. Azevedo, and L. F. Wanner, "ADeLe: Rapid Architectural Simulation for Approximate Hardware," in *30th Symposium on Computer Architecture and High Performance Computing (SBAC-PAD)*, 2018.

[13] D. Marwaha and A. Sharma, "A review on approximate computing and some of the associated techniques for energy reduction in iot," in *2018 2nd International Conference on Inventive Systems and Control (ICISC)*, Jan 2018, pp. 319–323.

[14] C. Slayman, "Soft error trends and mitigation techniques in memory devices," in *2011 Proceedings - Annual Reliability and Maintainability Symposium*, Jan 2011, pp. 1–5.

[15] A. Sampson, W. Dietl, E. Fortuna, D. Gnanapragasam, L. Ceze, and D. Grossman, "EnerJ: Approximate Data Types for Safe and General Low-power Computation," in *Proceedings of the 32Nd ACM SIGPLAN Conference on Programming Language Design and Implementation*, ser. PLDI '11. New York, NY, USA: ACM, 2011, pp. 164–174.

[16] A. Sampson, A. Baixo, B. Ransford, T. Moreau, J. Yip, L. Ceze, and M. Oskin, "ACCEPT: A Programmer-Guided Compiler Frameworkfor Practical Approximate Computing," University of Washington, UW-CSE, Tech. Rep., 2015.

[17] J. Dorn, J. Lacomis, W. Weimer, and S. Forrest, "Automatically exploring tradeoffs between software output fidelity and energy costs," *IEEE Transactions on Software Engineering*, pp. 1–1, 2018.

[18] E. Carlisle and A. D. George, "Cache fault injection with DrSEUs," in *IEEE Aerospace Conference*, March 2018, pp. 1–11.

[19] D. S. Khudia, B. Zamirai, M. Samadi, and S. Mahlke, "Rumba: An online quality management system for approximate computing," in *Proceedings of the 42Nd Annual International Symposium on Computer Architecture*, ser. ISCA '15. New York, NY, USA: ACM, 2015, pp. 554–566.

[20] S. Mittal, "A survey of techniques for approximate computing," *ACM Comput. Surv.*, vol. 48, no. 4, pp. 62:1–62:33, Mar. 2016.

[21] A. Yazdanbakhsh, D. Mahajan, H. Esmaeilzadeh, and P. Lotfi-Kamran, "Axbench: A multiplatform benchmark suite for approximate computing," *IEEE Design Test*, vol. 34, no. 2, pp. 60–68, April 2017.

[22] M. R. Guthaus, J. S. Ringenberg, D. Ernst, T. M. Austin, T. Mudge, and R. B. Brown, "Mibench: A free, commercially representative embedded benchmark suite," in *Proceedings of the Fourth Annual IEEE International Workshop on Workload Characterization. WWC-4 (Cat. No.01EX538)*, Dec 2001, pp. 3–14.

[23] and A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, April 2004.

[24] A. N. Avanaki, "Exact histogram specification optimized for structural similarity," *Optical Review*, vol. 16, p. 613–621, 2009.

[25] I. Gouy, "The Computer Language Benchmarks Game," 2004? [Online]. Available: benchmarksgame-team.pages.debian.net/benchmarksgame/

[26] G. Fursin, "Collective Benchmark," 2008. [Online]. Available: https://ctuning.org/cbench/