AJC
Academic Journal on Computing, Engineering and Applied Mathematics
EAM

2022
Volume 3 Issue 2

Academic Journal on Computing, Engineering and Applied Mathematics

UFT
UNIVERSIDADE FEDERAL
DO TOCANTINS

ISSN: 2675-3588

Editorial (Português): uma breve história do hardware, seus desafios, e impacto sobre o consumo de energia

Tiago da Silva Almeida[1,2]

[1] *Universidade Federal do Tocantins, Palmas / TO, Brasil*

[2] *Universidade de Campinas, Instituto Computação, Campinas / SP, Brasil*

Este artigo representa a opinião do autor sobre o campo da arquitetura de computadores e os desafios futuros. Este trabalho tenta estabelecer como a computação se tornou tão importante como é hoje. Além disso, procura elencar alguns dos principais desafios da área de projetos de computadores. Assim, este artigo foi escrito para estudantes e pesquisadores que desejam ver o panorama geral do assunto, e nenhum destes é amplamente discutido, depois, essa tarefa é quase impossível e não é a intenção. No entanto, alguns grandes *insights* podem surgir deste artigo, uma vez que o futuro parece ser *co-design* e aproximado.

I. Quais as consequências da Lei de Moore?

A capacidade de fabricar um transistor em escalas cada vez menores foi algo sem precedentes e possibilitou os avanços cada vez maiores na computação. Essa capacidade fez com que Gordon Moore, em seu artigo [1], previsse "*The complexity [of integrated circuits] for minimum component costs has increased at a rate of roughly a factor of two per year*". Essa frase pode ser mal interpretada, ela fala sobre o custo e o tamanho do transistor. Ou seja, a cada dois anos podemos construir transistores mais baratos e menores em uma mesma área de silício.

Essa afirmação tem duas implicações principais: i) podemos construir circuitos mais complexos sem aumentar a área, e ii) um circuito mais complexo pode aumentar o desempenho, mas não necessariamente a potência diminuirá na mesma proporção[1]. Além disso, um transistor menor pode levar a desafios adicionais e problemas de fabricação também. Voltarei brevemente nestes problemas mais adiante.

Ao longo de 65 anos, os chips se tornaram mais densos a uma taxa de 1,8 anos, o desempenho aumentou a uma taxa de 1,5 anos e a eficiência energética aumentou a uma taxa de 1,5 anos. Este estado também pode ser conhecido como "lei de Dennard". A Lei de Dennard[2] afirma que as dimensões de um dispositivo diminuem, assim como o consumo de energia. Enquanto isso, transistores menores funcionavam mais rápido, usavam menos energia e custavam menos [2]. No entanto, os resultados em energia podem ser explicados por melhorias no consumo de energia de um material de fabricação específico para outro.

No artigo de Jonathan Koomey[3] [3] foi citada uma pesquisa feita pelo físico Richard Feynman em 1985, que mostrou que há um espaço teórico para melhoria na eficiência energética por um fator de 10^{11} naquele momento. E os autores estimaram um aumento real por um fator de 10×10^4, ou seja, ainda há muito espaço para melhorias, pelo menos teoricamente [3].

[1] Faço aqui um adendo sobre o consumo de energia. Há duas formas principais, o consumo estático e o consumo dinâmico. O consumo estático está diretamente relacionado a área em silício, áreas maiores e transistores maiores levam a maiores consumos estáticos. Consumo dinâmico é o consumo devido a atividade de chaveamento dos transistores, o qual reflete a sua utilização, ou a carga de trabalho de computação, cargas maiores levam a maiores consumos.

[2] `https://en.wikipedia.org/wiki/Robert_H._Dennard`

[3] `https://en.wikipedia.org/wiki/Jonathan_Koomey`

Lembrando, o consumo de um transistor é devido ao tamanho da fonte ao dreno, ou seja, quanto menor o caminho, menor a potência, menores custos e menor tempo de computação. No entanto, o consumo total dos computadores inclui outros componentes, como discos, RAM (*Random Access Memory*)), redes etc., além da CPU (Unidade Central de Processamento).

O encolhimento dos transistores e as melhorias no consumo permitiram o surgimento da computação móvel. Este fator ligado aos comportamentos de consumo levou ao crescimento rápido do número de computadores pessoais nas décadas de 80 e 90, e posteriormente os computadores móveis, como os celulares.

Não é apenas o CPU que melhorou a eficiência energética, mas também outros componentes, por exemplo, a tela ficou mais barata [as telas LCD (*Liquid Crystal Display*) usam cerca de um terço das telas CRT (*Cathode Ray Tube*)], o que diretamente influencia no consumo de energia.

O paralelismo do processamento, certamente tem um impacto importante no consumo de energia. Além da estrutura física dos *data centers* e os *data warehouses* também influenciam. A utilização do de serviços em nuvem são tópicos da proxima seção.

II. E A COMPUTAÇÃO EM NUVEM?

Em partes, o serviço de nuvem surgiu para que começasse a ser vantajoso economicamente toda a estrutura *data centers* e *data warehouses*. A quantidade de energia consumida pelo servidor é enorme [4]. A maior parte desse consumo está relacionada à refrigeração e infraestrutura e seu impacto no meio ambiente também é enorme.

Quando comparamos servidores com *laptops*, ou qualquer tipo de computador pessoal, o uso da CPU é diferente. Por exemplo, em computadores de baixo consumo, a eficiência energética é pensada no desempenho máximo e na energia do modo ocioso. No entanto, em servidores na maioria das vezes o uso da CPU é de cerca de 30% e raramente atinge 100% de uso. E por um período considerável fica perto de 3%, mas raramente em modo inativo. Este uso não é por acaso. Os servidores são projetados para ter folga no uso da CPU para evitar algum estresse na taxa de transferência e não ficar em modo ocioso para evitar o desperdício de recursos. Outras estratégias, como aplicações que usam o modo ocioso com menor uso de dados ou reduzem a energia em discos, podem ser muito difíceis de gerenciar.

O desempenho pode ser definido como solicitações por segundo e eficiência energética dividindo a potência máxima pelo seu uso. Em um servidor otimista ainda consumirá metade da energia sem nenhum uso. A partir deste ponto, não há eficiência energética. O que pode crescer à medida que o uso também cresce. Se olharmos para 20% a 30% de uso, o uso típico para a maioria das aplicações, há uma lacuna visível em eficiência e espaço para melhorias.

Então, como os servidores podem ser mais eficientes? A resposta é difícil, mas os projetistas devem estar atentos à carga de trabalho do servidor para projetar servidores que aumentem a potência à medida que a carga de trabalho também aumenta, tornando-os mais proporcionais, ou seja, mais proporcionais à energia [4].

Uma solução alternativa a esse é a Computação em Névoa (*Fog Computing*). Temos a internet com bilhões de coisas conectados à ela. Quando uso a palavra “coisa”, quero dizer literalmente coisa. Qualquer coisa pode se tornar conectada à internet, ou como chamamos IoT (*Internet of Things*).

Se pensarmos em grandes sistemas de computação em rede, ligado à vários dispositivos ou sistemas de computação, como em uma cidade inteligente. As coisas, ou dispositivos das pessoas geram uma grande quantidade de dados para ser processada somente no *data center*. Então por que não fazer parte desse processamento também nas bordas dessa rede? Ou seja, nos próprios dispositivos, já que eles possuem uma demanda menor de energia e são sistemas de computação embutidos nas coisas.

Mas qual a complexidade de se projetar esses sistemas? Certamente enorme. Ele se torna agnóstico em relação a aplicação que ele precisa executar? Certamente não. Aplicações diferentes terão demandas de dispositivos diferentes na borda, transdutores diferentes[4]. E certamente com necessidades de

[4]Faço aqui um adendo aos transdutores, referendo-me à sensores e atuadores.

desempenho diferentes, o que impacta diretamente no consumo de energia.

A eficiência energética nesses casos, passa por um planejamento e balanceamento da carga de trabalho. Ou seja, o propósito da aplicação, juntamente com os dados de entrada, importam para otimização tanto a performance, como também o consumo de energia[5]. O que é discutido na próxima seção.

III. Qual o impacto do software no hardware?

Isso nos leva a outra questão, como projetar de forma eficiente considerando a aplicação? Muitos dos projetos de software são feitos somente por engenheiros / analistas de software. Esse não é o cenário ideal. Os arquitetos de hardware e software podem fazer grandes avanços em um projeto cooperativo chamado *co-design* de hardware/software.

Ao projetar hardware com o software em mente, é possível encontrar oportunidades para melhorar o desempenho e a energia consumida. O desperdício de energia pode surgir de arquiteturas genéricas em alguns cálculos para uma carga de trabalho específica. Assim, a entrada deve ser considerada também no projeto.

Isso pode ser visto em processadores de celulares modernos, onde existem *designs* heterogêneos com CPUs e aceleradores diferentes no mesmo SoC[6] (*System-on-Chip*). Por outro lado, esses SoCs criam gastos adicionais em área/consumo e complexidade. Então, como podemos ser mais agressivos em eficiência energética? A computação aproximada é um paradigma promissor para responder a essa pergunta.

IV. O que é computação aproximada?

A computação aproximada tenta explorar agressivamente a natureza da própria computação. Em essência, todos os computadores possuem algum nível de aproximação na computação, por exemplo, em uma operação de arredondamento ou truncamento, ou a discretização de algum sinal ou representação da realidade. Para algumas aplicações, onde não há uma resposta exata, essas aproximações podem ser ainda maiores sem perda de confiabilidade na saída [5, 6, 7, 8]. A computação se torna simplesmente boa o suficiente.

Em alguns casos, técnicas de "julgamento aproximado" podem ser utilizadas, ou seja, dentro de um determinado limite a saída pode apresentar erros, ou que não impacte na qualidade dos dados relacionados. Assim, é possível construir sistemas computacionais tolerantes a desvios de resultados, que são reduzidos e/ou otimizados, levando a sistemas que ocupam uma área menor em silício e consequentemente mais rápidos (pensando na implementação e finalidade) e economia de energia.

As principais abordagens para computação aproximada são [8, 6, 5, 9]:

- Software Aproximado:
  - Transformação de código – baseada na identificação de partes do código que podem ser descartadas sem exceder um limite de erro na saída da aplicação [10]. Nesses casos também pode ser usada o *loop peforation* [11], que identifica automaticamente partes resilientes a erros dentro de um código que pode ser ignorado mantendo o erro dentro de um intervalo predefinido. Todos eles são executados principalmente pelo compilador. Outra possibilidade é o relaxamento de tempo, onde relaxa-se a sincronização em programas paralelos e explora as propriedades de algoritmos seriais para aumentar o paralelismo eliminando dependências entre computações [11];

[5]Programas diferentes possuem consumos de energia diferentes, independente da plataforma de hardware utilizada. O mesmo programa com entradas diferentes possuem consumos de energia diferentes. Isso porque o consumo de energia dinâmica é determinado pelo fator de chaveamento do transistor.

[6]O sistema em um chip é essencialmente um circuito que integra um sistema eletrônico ou de computador, inteiro, dentro do chip. Normalmente possuem uma ou mais CPUs idênticas (homogênea) ou diferente (heterogêneas), controladores de periféricos e memórias.

  - Redução de padrão – nesses casos pode-se alterar a precisão (número de bits) dos dados, sendo possível melhorar o desempenho e o consumo de energia [6]. Um programa de ponto flutuante pode ser aproximado para um programa de ponto fixo. O compilador poderia procurar a precisão mínima na mantissa em um código de ponto flutuante, levando em conta as restrições em relação ao erro de saída. Ou mesmo o método de *transprecision* poderia ser usado em tais cenários [12, 13, 14];

- Arquitetura Aproximada:

  - Extensões do ISA (*Instruction Set Architecture*) – muitas oportunidades de aproximação podem surgir da alteração do ISA. Por exemplo, algumas instruções ou segmentos de código escolhidos podem ser executados em modo aproximado, ou seja, em hardware aproximado; truncamento e/ou alteração de caminhos críticos em circuitos para aumentar o desempenho em detrimento da precisão [11]. Isso significa que, para computação aproximada de baixa granularidade, segmentos aproximáveis podem ser carregados para acelerar a eficiência energética, e escolhas arquitetônicas semelhantes à computação aproximada de granulação fina podem ser feitas para núcleos com baixo consumo de energia, mas não confiáveis em tempo de execução [9];
  - Armazenamento – em detrimento da qualidade dos dados, desempenho, eficiência energética e área do chip, pode-se relaxar memórias e armazenamento [11]. Assim, as memórias híbridas com matrizes de memória não confiáveis com eficiência energética permitem uma redução de escala excessiva de tensão, por exemplo. Esses métodos podem ser aplicados em caches, discos ou até RAMs. Mas no último caso, cuidados adicionais devem ser aplicados para separar os dados do código [15]. Outra opção inclui inserir um co-processador na memória para evitar tráfego de dados entre a CPU e consequente economia de energia no *throughput*;

- Circuitos Aproximados:

  - Lógica imprecisa – referindo-se a projetos de componentes imprecisos e a síntese de circuitos lógicos aproximados [9, 6, 11]. Neste caso, as aproximações podem ser introduzidas em diferentes níveis de abstração, por exemplo, desde *design* funcional (para introduzir comportamento errôneo) [16], *netlists* (para caminhos críticos mais curtos) [17], para álgebra booleana (para reduzir o número de transistores) [18, 19];
  - *Voltage overscale* – este método consiste em diminuir a tensão de alimentação do circuito para obter uma redução de energia, mas também podem ocorrer erros de temporização. Esta é uma técnica de granularidade fina e só melhora o consumo de energia. Ele poderia usar um circuito menos preciso, mas mais eficiente em termos de energia para computação ou reduzir propositalmente a tensão de alimentação de certos componentes de hardware para compensar energia e precisão. Essa ideia pode ser aplicada em diferentes componentes e níveis, mas principalmente em memórias [11, 9].

É interessante notar que existem algumas interseções entre essas abordagens. Compiladores aproximados podem ser criados para otimizar e encontrar boas oportunidades com *loop peforation* e redução de padrões. Aproximações na memória podem usar *voltage overscale* em algumas seções de dados da memória. Tudo isso pode ser aplicado com algumas instruções específicas no compilador, arquitetura e circuito. Ou ainda, componentes lógicos extras aproximados, como somadores ou multiplicadores, poderiam ser usados para reduzir o caminho crítico para algumas instruções, permitindo verificar a saída para alguns dados críticos, ao custo de um *overhead* desprezível na área.

Considerando a folga para melhorar a eficiência energética mencionada anteriormente, as aproximações em hardware (arquitetura e/ou circuito) têm a possibilidade de serem mais agressivas na econo-

mia de energia, ou esforço computacional, de várias maneiras. Se considerarmos apenas aproximações em software, a economia é limitada pelo hardware atual.

V. O QUE SÃO ACELERADORES?

Por outro lado, uma forma de ganhar desempenho e eficiência energética seria através de aceleração de hardware. Se pensarmos bem, os computadores já foram acelerados. Na década de 80 e possivelmente 90, haviam várias placas de conexão nos computadores que possuíam especializações para algumas tarefas específicas, como processamento de áudio, vídeo, modem etc. Enquanto a lei de Moore ainda estava em vigor, fomos capazes de eliminar essas placas e a computação passou a ser majoritariamente em software, já que um processador mais complexo conseguia escalar em desempenho, mesmo com uma carga de trabalho maior.

Já que a escala não é mais uma opção, voltamos a pensar em um processamento dedicado para algumas aplicações, ou seja, acelerados. Evidente que hoje não voltamos a usar uma grande quantidade de placas. Ou invés disso, colocamos aceleradores dentro de um mesmo chip, permitindo uma escala ainda maior da latência de comunicação entre esses núcleos, sendo eles agnósticos ou específicos. Ou seja, como já comentado, os SoCs.

Em um *design* tipicamente acelerado, um processador *host* tem a responsabilidade de entregar algumas partes da aplicação a um acelerador de dispositivo e, ao fazê-lo, acelerar as aplicações. Algumas aplicações de CPU/GPU (*Graphics Processing Unit*) podem atingir mais de 2000x de aceleração [20]. Ao introduzir aproximações no projeto do acelerador é possível ser mais agressivo na diminuição do esforço computacional. No entanto, considerando um espaço tridimensional, qualidade de resultados, consumo de energia e desempenho, encontrar o projeto ótimo é um problema não polinomial completo, considerando a complexidade do que é qualidade para algumas aplicações e como uma pequena alteração no projeto pode impactar no erro e no esforço computacional.

VI. QUAIS DESAFIOS ESTÃO ABERTOS?

Já que o *design* de sistemas heterogêneos é um problema de difícil solução. Certamente esse é sim um grande desafio em aberto. Mas quando se fala em computação, há muitos outros problemas que precisam ser solucionados para que continuemos avançando para novas tecnologias. Um *design* tipicamente heterogêneo, um SoC com ao menos uma CPU para gerenciamento de energia, alguma CPU para operações de maior precisão, como em ponto flutuante, alguma outra CPU com um *pipeline* menor para tarefas simples, e algum modelo de acelerador. Isso já é uma realidade, mas ainda há muito o que melhorar em termos de comunicação entre as partes[7], em como o software é compilado e como o ambiente de execução deve se comportar para que o sistema tenha o melhor desempenho. Esses são problemas difíceis de serem resolvidos.

Mas e qual acelerador usar no SoC? Existem muitas variações em relação a estrutura e funcionamento desses aceleradores. Além da clássica GPU, exitem as TPU (*Tensor Processor Unit*), NPU (*Neural Processor Unit*), SAM (*Systolic Array Multiplication*), STA (*Systolic Tensor Array*), e para complicar ainda mais, aceleradores dedicados à uma única aplicação. O leitor pode ter notado que todos esses modelos de aceleradores são focados em aplicações de IA (Inteligência Artificial)[8].

De fato, aplicações de IA são muito populares e possuem uma demanda muito alta de recursos computacionais. A própria demanda por novos métodos de IA é muito grande, e ainda há muito o que pesquisar nessa área. Contudo, considerando todo o esforço computacional das aplicações de IA, projetar um sistema energeticamente eficiente para IA é um grande desafio em aberto.

O *co-design* já é uma realidade, mas pode se tornar cada vez mais presentes nas empresas para que

[7]Uma possível solução são o NoC (*Network-on-Chip*). Onde a rede de comunicação entre as partes é fabricado no chip, e permite ganhos de latência entre os módulos do SoC.

[8]Ao contrário do que um usuário leigo possa pensar, uma IA não é realmente inteligente. Em primeiro, lugar a própria definição de inteligência é complexa de se chegar. Em segundo lugar, IAs são modelos algébricos e estatísticos, com altíssima precisão, com capacidade adaptativa, e certamente é capaz de realizar muitas tarefas melhores do que os humanos.

se possa chegar em sistemas computacionais mais eficientes. Evidentemente o *co-design*, e o próprio *design*, é feito em modelos que tentam representar a realidade. E qual é o melhor modelo? Qual o melhor nível de abstração? Quanto mais abstrato é o modelo, mais fácil é encontrar uma boa solução para o problema, porém, mais distante ele é da realidade. Então precisamos repensa-los também.

Como há sempre um distanciamento da realidade, há um erro inerente no *design*. Além do próprio erro contido na computação, como já comentado. Qual é o nível de erro aceitável? O quanto e como relaxar a computação (ou inserir aproximações)? Podemos ainda pensar na computação em névoa. Como distribuir e balancear a carga de trabalho? Como diminuir a latência de comunicação? Como aumentar a segurança do sistema, especialmente em dispositivos com capacidade de processamento mais limitada? São outros problemas que precisam ser levados em conta e resolvidos. E todas as soluções conhecidas adicionam algum *overhead* no sistema.

Deixei ainda de fora da discussão, questões muito sérias sobre o processo de fabricação em si. Que geram efeitos *dark silicon*[9] e aceleram o envelhecimento do hardware, o que por sua leva a falhas de sincronização e até a falhas de segurança no sistema. E se o sistema gerar alguma falha, e não acusar nenhum erro? O leitor pode pensar que isso não é possível, mas não só possível, como acontece todos os dias em grandes *data centers*.

Concluo, dizendo que não podemos nunca pensar na computação como atividade meio para resolver algum problema mercadológico, precisamos pensar na computação como atividade fim. Acredito que o futuro seja em névoa, aproximado e heterogêneo. O engenheiro de software pode ser um grande amigo do engenheiro de hardware, não só em empresas de tecnologias, mas em outros segmentos de mercado também.

[9] *Dark silicon* é uma limitação física no processo de fabricação, no qual o circuito não é capaz de operar em potência e frequência máximas, causando um "desligamento" de algumas áreas do processador ou SoC. Quanto menor a escala de fabricação, maior exponencialmente é o efeito de *dark silicion*.

Expediente

Realização
Fundação Universidade Federal do Tocantins (UFT)
Quadra 109 Norte, Avenida NS-15, ALCNO-14 | Bloco III | sala 214 |Plano Diretor Norte | 77001-090 | Palmas / TO | Brasil

Periodicidade

Este periódico possui periodicidade semestral e utiliza a Licença Creative Commons 4.0 - CCBY 4.0. Contudo, a publicação dos artigos em modalidade avançada ou ahead of print, ou seja, tão logo os manuscritos aprovados sejam editados para publicação, é possível. O AJCEAM não possui taxas de publicação, tanto pouco de submissão de manuscritos, sendo totalmente gratuita para autores e leitores.

Indexadores
Google Acadêmico, desde 9 de maio de 2020
International Standard Serial Number – ISSN, desde 28 de maio de 2020
Crossref, desde 7 de junho de 2020
Revistas de Livre Acesso – LivRe, desde 24 de junho de 2020

REFERÊNCIAS

[1] G. E. Moore, "Cramming more components onto integrated circuits, reprinted from electronics, volume 38, number 8, april 19, 1965, pp.114 ff." *IEEE Solid-State Circuits Society Newsletter*, vol. 11, no. 3, pp. 33–35, 2006.

[2] R. Dennard, F. Gaensslen, H.-N. Yu, V. Rideout, E. Bassous, and A. LeBlanc, "Design of ion-implanted mosfet's with very small physical dimensions," *IEEE Journal of Solid-State Circuits*, vol. 9, no. 5, pp. 256–268, 1974.

[3] J. G. Koomey, S. Berard, M. Sanchez, and H. Wong, "Implications of historical trends in the electrical efficiency of computing," *IEEE Annals of the History of Computing*, vol. 33, pp. 46–54, 2011.

[4] Luiz André Barroso and Urs Hölzle, "The Case for Energy-ProportionalComputing," no. December, pp. 33–37, 2007.

[5] H. B. Barua and K. C. Mondal, "Approximate computing: A survey of recent trends—bringing greenness to computing and communication," *Journal of The Institution of Engineers (India): Series B*, vol. 100, pp. 619–626, 2019. [Online]. Available: https://doi.org/10.1007/s40031-019-00418-8

[6] A. Aponte-Moreno, A. Moncada, F. Restrepo-Calle, and C. Pedraza, "A review of approximate computing techniques towards fault mitigation in hw/sw systems," *2018 IEEE 19th Latin-American Test Symposium, LATS 2018*, vol. 2018-Janua, pp. 1–6, 2018.

[7] L. Kugler, "Is "good enough" computing good enough?" *Communications of the ACM*, vol. 58, pp. 12–14, 2015.

[8] J. Han and M. Orshansky, "Approximate computing: An emerging paradigm for energy-efficient design," *Proceedings - 2013 18th IEEE European Test Symposium, ETS 2013*, pp. 1–6, 2013.

[9] Q. Xu, T. Mytkowicz, and N. S. Kim, "Approximate computing: A survey," *IEEE Design and Test*, vol. 33, pp. 8–22, 2 2016.

[10] S. Umesh and S. Mittal, "A survey of techniques for intermittent computing," *Journal of Systems Architecture*, vol. 112, 2021.

[11] M. Shafique, R. Hafiz, S. Rehman, W. El-Harouni, and J. Henkel, "Invited: Cross-layer approximate computing: From logic to architectures," *Proceedings - Design Automation Conference*, vol. 2016-Augus, 2016.

[12] M. Alioto, V. De, and A. Marongiu, "Energy-quality scalable integrated circuits and systems: Continuing energy scaling in the twilight of moore's law," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 8, pp. 653–678, 2018.

[13] S. Mach, D. Rossi, G. Tagliavini, A. Marongiu, and L. Benini, "A transprecision floating-point architecture for energy-efficient embedded computing," *Proceedings - IEEE International Symposium on Circuits and Systems*, vol. 2018-May, pp. 1–5, 2018.

[14] A. C. I. Malossi, M. Schaffner, A. Molnos, L. Gammaitoni, G. Tagliavini, A. Emerson, A. Tomás, D. S. Nikolopoulos, E. Flamand, and N. Wehn, "The transprecision computing paradigm: Concept, design, and applications," *Proceedings of the 2018 Design, Automation and Test in Europe Conference and Exhibition, DATE 2018*, vol. 2018-Janua, pp. 1105–1110, 2018.

[15] J. F. Filho, I. B. Felzmann, R. Azevedo, and L. F. Wanner, "Axram: A lightweight implicit interface for approximate data access," *Future Generation Computer Systems*, vol. 113, pp. 556–570, 2020.

[16] J. Castro-Godinez, J. Mateus-Vargas, M. Shafique, and J. Henkel, "Axhls: Design space exploration and high-level synthesis of approximate accelerators using approximate functional units and analytical models," *IEEE/ACM International Conference on Computer-Aided Design, Digest of Technical Papers, ICCAD*, vol. 2020-Novem, 2020.

[17] D. Hernandez-Araya, J. Castro-Godinez, M. Shafique, and J. Henkel, "Auger: A tool for generating approximate arithmetic circuits." Institute of Electrical and Electronics Engineers Inc., 2 2020.

[18] M. S. Ansari, B. F. Cockburn, and J. Han, "A hardware-efficient logarithmic multiplier with improved accuracy," *Proceedings of the 2019 Design, Automation and Test in Europe Conference and Exhibition, DATE 2019*, pp. 928–931, 2019.

[19] S. Gandhi, M. S. Ansari, B. F. Cockburn, and J. Han, "Approximate leading one detector design for a hardware-efficient mitchell multiplier," *2019 IEEE Canadian Conference of Electrical and Computer Engineering, CCECE 2019*, pp. 54–57, 2019.

[20] C. Ye, P. Zhang, Z. Wan, R. Yan, and D. Sun, "Accelerating cfd simulation with high order finite difference method on curvilinear coordinates for modern gpu clusters," *Advances in Aerodynamics*, vol. 4, no. 1, 2022. [Online]. Available: www.scopus.com