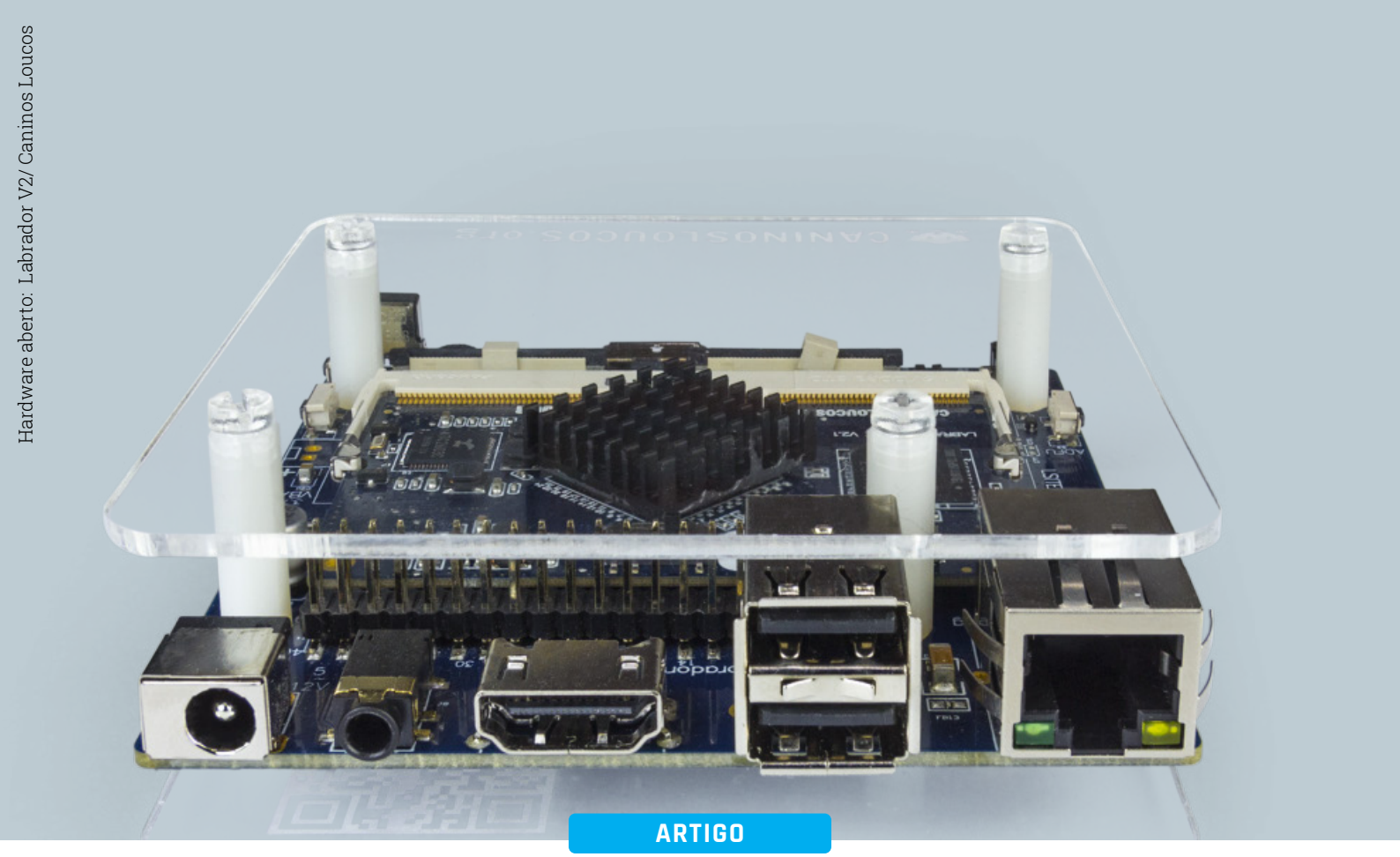

Hardware aberto: Labrador V2/ Caninos Loucos

ARTIGO

HARDWARE ABERTO, UMA ANÁLISE DE POSSIBILIDADES

POR

Luigi Carro
carro@inf.ufrgs.br

Os principais exemplos de *hardware* aberto são processadores, clássicos como CPUs RISC, superescalares, VLIW e mesmo máquinas vetoriais, FPGAs (*Field Programmable Gate Arrays*), placas de divulgação (por exemplo, Arduino) e o próprio CAD que suporta a realização de projetos sobre estas plataformas abertas, que vai desde compiladores até programas de síntese de alto nível, mapeamento, *placement* e *routing*.

Instâncias de hardware aberto

O exemplo mais completo de processador aberto é o BOOM [1], *Berkeley Out-of-Order Machine*. Este é um processador baseado na arquitetura RISC-V, e é possível modificá-lo incluindo ou retirando instruções, mas também mudando sua organização, de pipeline simples para

fora de ordem de 2, 4 ou mais *issues*. Todo o ambiente de suporte para a síntese e a simulação da CPU é fornecido e por sua vez pode ser modificado, e o processador obtido é extremamente competitivo em relação às CPUs comerciais com as mesmas organizações. Muito importante, o BOOM pode rodar Linux, tem suporte à hierarquia de memórias e à memória virtual, além de compiladores e otimizadores.

Embora o BOOM permita enorme latitude na exploração do espaço de projeto, ele em si não é uma máquina vetorial, extremamente necessária nos problemas baseados em grande volume de dados do mundo atual. Existem também máquinas vetoriais abertas, como o VESPA, um soft-processor vetorial para FPGAs, totalmente aberto e suportado pela universidade de Toronto [2]. Embora um projetista possa projetar num FPGA qualquer tipo de processador, ao se dispor de uma máquina vetorial como soft-processor (que pode ser facilmente mapeada para um FPGA), o foco do projeto pode ser então o desenvolvimento do software que irá trabalhar com a enorme massa de dados. O objetivo do VESPA é a simplificação do projeto de *hardware* de uma máquina massivamente paralela, e todo o suporte de *software* para síntese e simulação pode ser encontrado.

Processadores abertos para sistemas embarcados já foram objetos de livros, como por exemplo o VEX [3], no qual uma máquina VLIW (*very long instruction word*) foi apresentada com arquitetura e organização abertas, bem como o compilador e ferramentas de suporte. Este processador poderia explorar melhor o paralelismo, em nível de instruções, disponível em aplicações embarcadas, com custo energético muito menor que um superescalar. Extensões reconfiguráveis deste processador podem ser vistas em [4], o que garantia sua adaptação a diferentes cenários com máxima eficiência, e com toda a cadeia de ferramentas de síntese, compilação e simulação.

Muitos *soft-processors* foram feitos, na sua maioria, para serem implementados em FPGA primordialmente. Embora existam diferentes arquiteturas comerciais fechadas, existe também o projeto de um FPGA totalmente aberto, seja o hardware seja o software, o *Open Source FPGA Foundation*[5]. O objetivo da OSFPGA é acelerar a adoção de FPGAs como um componente importante do processamento de informações, para execução acelerada de *software* crítico, sem as barreiras de entrada de um determinado fabricante. O grande problema é que, como o FPGA é base para diversos outros projetos, o grau de otimização que esta plataforma tem de fornecer é extremamente elevado, e, portanto, dificilmente alcançável apenas de maneira aberta.

Finalmente, tem-se o Arduino, que é uma plataforma baseada em microcontroladores e todo o suporte de hardware e software para que funcionem quase como *plug&play* [6]. Placas Arduino podem ler entradas digitais e analógicas, como botões, sensores ou mesmo mensagens complexas, e tomar uma ação a partir disto, ativando leds, um motor, ou respondendo à mensagem e gerando novas conexões. Toda a pilha de *software* suporta a programação de diferentes dispositivos. As CPUs envolvidas podem ser microcon-

troladores de 8-bits ou mesmo CPUs mais complexas, mas o grande valor agregado está na interface de entrada e saída com o mundo real (conversores AD/DA embutidos, portas de I/O e rede, etc.) e a enorme facilidade de programação.

Pontos positivos

É evidente que os custos não recorrentes de desenvolvimento de *hardware* podem ser largamente minimizados pelo uso de plataformas abertas, como as citadas acima, e outras que não foram listadas. O maior impacto das versões abertas de diferentes arquiteturas de *hardware* se dá na educação. Com a disponibilidade de hardware complexo, os cursos podem oferecer aulas com mais profundidade, e também podem explorar diferentes possibilidades de implementação de um mesmo conceito, como por exemplo, investigando diferentes versões de uma CPU para um projeto visando a baixa energia.

Em termos de pesquisa, as possibilidades são ainda mais interessantes, já que as plataformas abertas permitem fáceis modificações e adaptações a múltiplos cenários de uso. A flexibilidade do *hardware* aberto permite aos alunos e pesquisadores uma flexibilização enorme para experimentação de novos conceitos de maneira muito rápida. Por exemplo, uma nova arquitetura para acelerar o software de um nicho de mercado, com versões com e sem pipeline ou superescalaridade é facilmente obtida pelo conjunto de ferramentas do RISC-V. Como o *hardware* é muito parecido com os efetivamente usados no mercado, o resultado dos estudos e pesquisa está muito mais próximo da realidade do mercado.

Pontos que dificultam a adoção

No âmbito empresarial, a adoção de *hardware* aberto é mais complexa. A primeira questão importante é que os custos de NRE (*non recurring engineering)* são mitigados, mas não vão a zero. Enquanto um *software* de arquitetura aberta pode ser rapidamente adaptado e transformado com apenas o custo da mão-de-obra, o *hardware* tem, além dos custos de transformação, os custos de fabricação.

Outro aspecto importante é a diferença de desempenho. Mesmo ao se prototipar uma CPU aberta em um FPGA comercial, a diferença de desempenho é tão grande que talvez as modificações que agregam valor não possam se manifestar em comparação com o HW fabricado em tecnologia comercial mais velha. Por exemplo, pode-se paralelizar um algoritmo para aumentar sua velocidade de execução, mas quando mapeado para um FPGA, este último exige um hardware maior e mais caro que uma CPU rápida não aberta, e, portanto, dificilmente o FPGA será utilizado.

Todo o *hardware* atual, mesmo o aberto, depende de uma enorme cadeia de *software* para seu correto funcionamento, desde os compiladores até os montadores e os ligadores. Uma nova arquitetura de CPU pode ser facilmente feita com o RISC-V, mas o compilador e outras ferramentas não são facilmente modificáveis (embora também abertos). Isto vai contra um tempo rápido de projeto, pois modificar um compilador pode aumentar em muito o tempo de projeto, sem falar na necessidade de uma equipe multidisciplinar para

atuação tanto no *hardware* quanto no *software*. E o projeto deixa de chegar ao mercado de maneira rápida se for necessário programar em *assembler*.

Por fim, é preciso considerar que os produtos de *hardware* não comoditizados, com alto valor agregado, são aqueles em nichos de grande demanda e procura, como por exemplo *network processors* para 5G, processadores vetoriais da Nvidia para processamento gráfico e de realidade aumentada. Para estes nichos não existe uma versão *open source*. Embora existam processadores vetoriais abertos, como citado em [2], o coração das máquinas da Nvidia é o *scheduler* e o suporte à CUDA, que facilitam enormemente a vida do programador. Neste nicho, nem o *hardware* nem o *software* são abertos.

Conclusão

O *hardware* aberto definitivamente tem seu espaço, sobretudo no meio acadêmico. Comercialmente, os desafios ainda são enormes, sobretudo pelo atraso de bases de *hardware* aberto, e pelo custo de fabricação que elas impõem. Os melhores exemplos de open *hardware* têm origem em ambientes universitários ou em empresas menores, e tendem a estar um ou mais passos atrás em relação ao produto comercializado.

Referências

1. https://boom-core.org/
2.https://www.eecg.utoronto.ca/VESPA/
3. Embedded Computing: A VLIW Approach to Architecture, Compilers and Tools
Joseph A. Fisher, Paolo Faraboschi, Cliff Young. ISBN-13: 978-1558607668
Elsevier 2005
4. The r-vex processor: http://rvex.ewi.tudelft.nl/
5. https://osfpga.org/
6. Arduino.cc

LUIGI CARRO é professor titular do Instituto de Informática da UFRGS, e pesquisador CNPq-1A. Com formação em Engenharia Elétrica (UFRGS-1985) e Doutorado em Computação (UFRGS-1996), já trabalhou na ST-Microelectronics (1989-90) e foi visiting professor em Montpellier, San Diego, Torino e Delft. Publicou mais de 150 artigos, tendo orientado 25 teses de doutorado e vários mestrados junto ao PPGC-UFRGS. Sua pesquisa atual abrange software e hardware de baixa energia