Help Wanted: A Modern-Day Turing[1]

Erik P. DeBenedictis, Sandia National Laboratories
R. Stanley Williams, Hewlett Packard Labs

The National Strategic Computing Initiative will inevitably produce new computer hardware, but what about software?

The US National Strategic Computing Initiative's (NSCI's) ambitious goal to dramatically increase computer hardware performance opens the door for a complementary software effort. NSCI white papers discussing technology R&D for increased energy efficiency[1] also include a new emphasis on machine learning.[2] However, to reap the new hardware's benefits, computer scientists will need to develop a model of computation for the new energy-efficient programming primitives. And programmers will need to learn to use the new model to efficiently implement diverse algorithms. Machine learning might be the most promising option to satisfy NSCI objectives "into the middle of this century" [1] but will require an enlightened model of computation that includes machine learning.

More Pain, More Gain

Figure 1 shows the NSCI white papers' technology options organized into three stages by the corresponding amount of software change. The options align according to the principle of "more pain, more gain."[1,2] New physics inside the computer will improve its capability and energy efficiency by varying degrees, but greater gain at runtime goes hand in hand with greater programming and computer architecture effort. No stage is fundamentally better or worse than any other, but each stage will apply in its own set of circumstances.

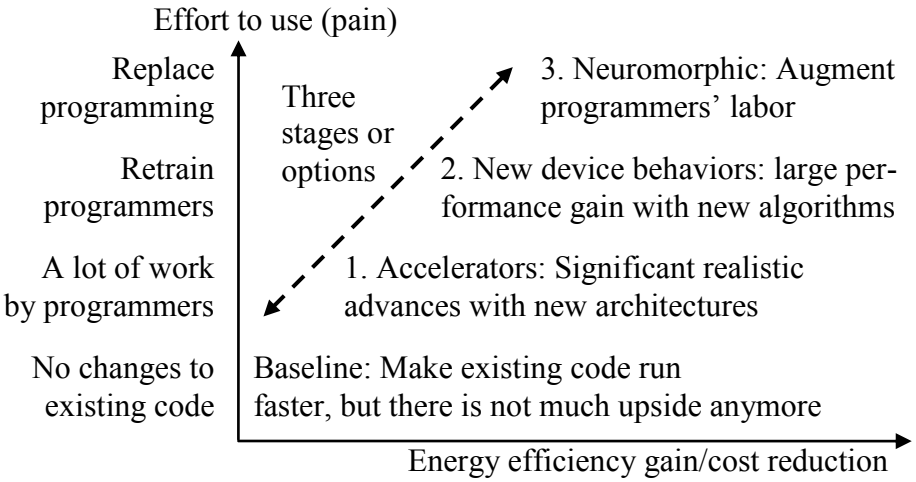


Figure 1. Progress in computing based on the principle of "more pain, more gain."

[1] Sandia National Laboratories approved for unlimited unclassified release SAND2017-8987 J
Published in Published as DeBenedictis, Erik P., and R. Stanley Williams. "Help Wanted: A Modern-Day Turing." *Computer* 49.10 (2016): 76-79. **DOI:** 10.1109/MC.2016.299

Programmers' pain or effort can be viewed as an opportunity. The US Office of Science and Technology Policy (OSTP) has challenged the technical community to develop a computer that can "solve unfamiliar problems using what it has learned,"[3] which is a programmer's task. In lieu of seeing programmer effort as undesirable overhead, stage 3 will use machine learning to assist the programmer.

Stage 1: CMOS Accelerators

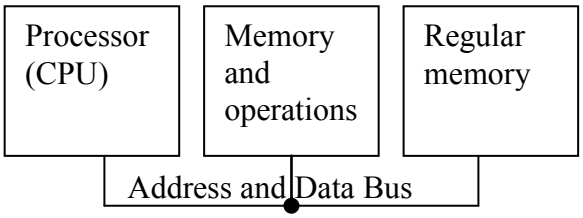
Programmers will have a critical role in augmenting today's computer architecture with performance-boosting accelerators.[2] Scaling semiconductor line width produced a straightforward result, but designing accelerators is different because it affects subsequent programming.

Today's microprocessors dissipate just a few percentage points of their energy for arithmetic—and often much less. The remaining energy consists of overhead from the von Neumann architecture's instruction processing and programmed data movement. The overhead can be recovered by replacing programmed behavior with hardwired paths, yet leads to a specialized processor for just a few similar programs. Recent research projects provide a vastly more energy-efficient substrate for the resulting non–von Neumann architecture.[4] The emergence of 3D memory such as memristors and flash creates a new opportunity to include large amounts of data in the accelerators.

Designing and fabricating a different accelerator for every algorithm would be prohibitively expensive. To control costs, we must design a few accelerator "building blocks" that provide functional diversity when combined and, when they're accompanied by a run time cost metric, define a new model of computation.[5] A model of computation is a set of primitives that programmers combine to create algorithms and applications. Each primitive imposes an energy or time cost when used. The best model would have highly expressive programming primitives that can be combined into many algorithms efficiently so run time costs are low.

For stage 1, the community should open up the programming primitive ecosystem to a broader audience and encourage innovation in finding the best primitives. The communications fabric within a computer could be based on a plug and play open standard that enables innovators to create and try out accelerator designs in different environments. IEEE could create an accelerator standard.

Figure 2 shows the accelerated architecture. The system is a von Neumann computer at the top level,




```
class SparseMatrixOrGraph {} acc;
acc.Set(2, 3, 3.1415);
acc.MatrixInvert();
acc.TransitiveClosure()
```

containing a processor and regular memory connected by a bus. However, the bus also contains a non–von Neumann accelerator. If appropriate to its function, the accelerator could contain gigabytes of memory. This example's accelerator stores a sparse matrix or graph in its memory and accelerates a handful of programming primitives on the stored data.

Figure 2. Stage 1's CMOS accelerator architecture.

An accelerator can be represented as an object in the sense of object-oriented programming, shown as class SparseMatrixOrGraph with instance acc. The acc data structure holds the data that's manipulated by

member functions. For sparse matrices, the accelerator has Set and MatrixInvert functions. Set has arguments that specify a row and column in the matrix and the value to store there. MatrixInvert inverts the matrix internally, changing both the sparsity pattern and data values.

The accelerator as described thus far could be a C++ class with no special hardware. Such a software implementation would improve neither speed nor energy efficiency, but it might be the best way to assess the model of computation. Programmers could use the accelerator to write a broad variety of algorithms, such as matrix and vector algebra algorithms, and answer the critical question about how efficiently the functions can be combined into algorithms. Let's assume the set of member functions Set and MatrixInvert in Figure 2 is expanded to a workable set for matrix computations, adding member functions for readout, vector–matrix operations, and so forth.

Because accelerators will be expensive to build, they should have many uses. Sparse matrices are similar to graphs, so designers will likely face business pressure to add support for common graph algorithms. Calculating transitive closure is common in graph algorithms but unusual in matrix algorithms. Let's say TransitiveClosure is added to the accelerator's design to expand its usefulness, but each addition's complexity counteracts the efficiency gain; thus, the number of these additions must be controlled.

This discussion illustrates the opportunity for programmers. In 1960, Fortran defined the basic building blocks of programs as expressions, loops, and subroutines. These followed naturally from Turing's and von Neumann's computing models. The current need is to create other sets of primitives that harness the energy-efficiency improvements in technology that will become available over the next decade.[4] These primitives will likely be of much heavier weight than just arithmetic operations, raising efficiency via economies of scale. For example, Fortran scalar expressions giving way to matrix operations on gigabytes of data. The programming primitives will also likely be domain-specific; that is, a supercomputer might need different primitives than an Internet server.

Stage 2: Devices with New Behaviors

If computer scientists can make effective CMOS accelerators, repeating this activity for new physical devices should yield large performance gains. Quantum computers are best known for their astronomical upside potential in prime factorization, yet other physical devices offer advantages over a broader range of applications. Stage 2 is defined to be accelerators or entire computers for which a key information processing primitive is based on an innovative device. This stage's devices would work differently than transistors and have orders of magnitude higher speed and/or lower energy for an equivalent amount of information processing. In this way, stage 2 meets NSCI's requirements for "computing beyond Moore's law."[1]

The hardware community is planning to control factors such as crosstalk and power density so that device properties remain stable as the system scales.[2] But this isn't enough. Consider an $N \times N$ crossbar array used as a memory or a neural network. Scaling up N only makes a bigger crossbar. Compare the crossbar to a group of arbitrarily wired logic gates. The number of possible behaviors scales exponentially with the number of gates in this latter case. The opportunity here is to create a new model of computation that's based on the new device properties yet have the flexibility of the AND-OR-NOT basis of Boolean logic. Possible new devices are memristors,[6] spin-based devices,[2] and reversible gates.[7]

Stage 3: Neural Networks

Stage 3 will require more expansive programmer involvement than the other stages. OSTP has urged

the technical community to create a learning computer that operates with the energy efficiency of the human brain,[3] thereby directing human-created computational systems toward machine learning and/or AI.

Attempts to apply our understanding of brain function, incomplete as it is, have nonetheless led to computing breakthroughs, such as the AlphaGo program that beat the best human Go player, Lee Sedol, four times out of five.[8] Key parts of AlphaGo weren't programmed by humans at all but learned by playing training matches with itself, intending to duplicate the way human Go players learn to play. Because a model of computation doesn't depend on details of a specific implementation, the models of computation used by biology and the AlphaGo software running on the GPU cluster might be the same. Neuromorphic computing, a related branch of interest, attempts to create physical work-alikes to the neurons, but based on precisely defined behaviors. The challenge for computer scientists and programmers will be to develop a model of computation for the hardware resulting from government R&D interest[2] in both GPU-based neural networks and neuromorphic computing.

The current mandate is to improve computers' energy efficiency, yet energy efficiency can't improve indefinitely. To maintain continuity of the computer industry, we would be prudent to anticipate future direction changes.[9] Perhaps, for this reason, the US government's initiative includes a branch on energy-efficient machine learning. If the community follows NSCI's research agenda, the quest to make computers more energy efficient could yield to a quest to make computers smarter. After all, the purpose of computers isn't to use energy but to find answers.

Our discussion presents a staged participation opportunity for computer scientists and programmers. The door is open for CMOS-based supercomputer accelerators. This could lead to accelerators based on novel devices, yet manufacturing these novel devices will delay commercial availability. If neurons are classed as devices, they also ought to have a model of computation; deep learning–type systems offer a glimpse of what such a model might look like.

Turing greatly contributed to the computing field through the invention of his namesake machine that can solve all computable problems. However, the Turing machine has very little control over resource usage such as running time or energy consumption—and, in addition, won't solve any problem at all unless programmed. There's government and community interest in more efficient computer hardware, but its benefits will need new theory to enable real-world application. Machine learning entered the picture unexpectedly, raising the tantalizing possibility that computing might transition smoothly into AI. However, this would require an even more challenging theory of computation that includes the computer learning to compute and doing so with controlled resource usage. Turing made quite a name for himself with the original theory of computation; would any reader consider updating it?

Acknowledgments

Sandia National Laboratories is a multiprogram laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the US Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

References

1. *National Strategic Computing Initiative Strategic Plan*, The Nat'l Strategic Computing Initiative Executive Council, July 2016; www.whitehouse.gov/sites/whitehouse.gov/files/images/NSCI%20Strategic%20Plan.pdf.

2. *A Federal Vision for Future Computing: A Nanotechnology-Inspired Grand Challenge*," white paper, Nat'l Technology Initiative, Oct. 2015; www.nano.gov/sites/default/files/pub_resource/federal-vision-for-nanotech-inspired-future-computing-grand-challenge.pdf.

3. L. Whitman, R. Bryant, and T. Kalil, "A Nanotechnology-Inspired Grand Challenge for Future Computing," White House blog, 20 Oct 2015; www.whitehouse.gov/blog/2015/10/15/nanotechnology-inspired-grand-challenge-future-computing.

4. M.M. Sabry Aly et al. "Energy-Efficient Abundant-Data Computing: The N3XT 1,000×," *Computer*, 2015, vol. 48, no. 12, pp.

24–33.

5. J.E. Savage, *Models of Computation: Exploring the Power of Computing*, Addison-Wesley, 1998.

6. R.S. Williams and E.P. DeBenedictis, *OSTP Nanotechnology‐Inspired Grand Challenge: Sensible Machines (extended version 2.5)*, white paper, 20 Oct. 2015; http://sensiblemachine.org/SensibleMachines_v2.5_N_IEEE.pdf.

7. D.J. Frank, "Reversible Adiabatic Classical Computation—An Overview,"*Proc. 2nd IEEE Rebooting Computing Summit*, 15 May 2014; http://rebootingcomputing.ieee.org/images/files/pdf/7-rcs2-adiabatic-reversible-5-14-14.pdf.

8. D. Silver et al., "Mastering the Game of Go with Deep Neural Networks and Tree Search," *Nature*, vol. 529, no. 7587, 2016, pp. 484–489.

9. E.P. DeBenedictis, "Rebooting Computers as Learning Machines," *Computer*, vol. 49, no. 6, 2016, p. 84–87.

Erik DeBenedictis *is a technical staff member in the Non-Conventional Computing Technologies Department at Sandia National Laboratories. Contact him at epdeben@sandia.gov.*

R. Stanley Williams *is a Senior Fellow at Hewlett Packard Labs. Contact him at stan.williams@hpe.com.*