Roundtable

Roundtable Panel Discussion at DAC 2019: Evolutionary Computing or Heuristic Forever?

Giovanni De Micheli, Antun Domic, Massimiliano Di Ventra, Martin Roettler, and Jason Cong

As demands for computing have been continuously increasing, various solutions have been proposed. Some approaches deal with improving algorithms and software programs, mainly through the tuning of advanced heuristics and learning methods. Some other tackle the problem by providing specialized hardware to enhance computation. Other revolutionary approaches, such as memcomputing and quantum computing, explore new paradigms in computation to beat the barrier of computational complexity.

A highly attended plenary panel at the 2019 Design Automation Conference (DAC) in Las Vegas, NV, USA, with the provocative title "Evolutionary Computing or Heuristic Forever?" spurred a lively discussion that is reported in this Roundtable. It is moderated by the panel organizer and moderator Giovanni De Micheli, EPFL, Switzerland. Panelists include Antun Domic, former CTO and senior VP at Synopsys; Jason Cong, University of California Los Angeles (UCLA); Massimiliano Di Ventra, University of California San Diego (UCSD); and Martin Roettler, Microsoft Research.

Digital Object Identifier 10.1109/MDAT.2020.2968279
Date of current version: 24 June 2020.

Giovanni De Micheli: In 1948, while computers were in their infancy, Arthur Clarke wrote a piece for the BBC called the *Sentinel*. It was a forward-looking piece and was later used as the subject of the movie, 2001 A Space Odyssey in 1968. The movie actually featured an eye that could look at you and could implement AI at its best. It could understand the moving of the mouth, understand the speech, the will, control humans, and even kill humans.

Giovanni De Micheli: Fortunately, this computer could be disabled by just unscrewing the memory parts and eventually the computer recognized that it was designed in Urbana (home of the University of Illinois). Maybe the problem was that the original designers who were too smart for the time! All these pose some questions about how computing is evolving today and this brings us to the issues that we would like to discuss today. In particular, computation in EDA (as an example in CS) is limited by complexity and we look for means to go beyond current limitations.

Giovanni De Micheli: Even simple problems like graph covering that are very common in EDA belong to complexity classes. For small instances, exact solutions can be provided by satisfiability (SAT) solvers. But when the size of the problem is large enough, solutions cannot be computed because of the exponential growth of computational needs. At the same time, our appetite for better results is large. That's called *quality of results* (QoR) and companies actually do spend a lot of money to achieve tools that can give you better results at the price of longer computational time. But at present, we know that very few problems can be solved exactly in the sizes that are relevant for the industry, and we rely mainly on heuristics.

Giovanni De Micheli: So, what's the future as problems get more complex and as the size of the problem gets larger? We do need some change of paradigms. One alternative is to have better heuristics. I do see machine learning (ML) as part of this because ML works fine on many problems, but we don't know why it works. So, in my opinion, ML is a heuristic like any other. Alternatively, we could extend our computers by adding computational accelerators or we could actually change the computation paradigm, like going from digital to mixed digital analog computing and/or relying on converging dynamical system to solve some problems. Going further, we need to consider quantum computing that allows us to exploit parallelism on a different computational fabric that intrinsically has parallelism.

Giovanni De Micheli: I will start now by leaving the mike to the first speaker Antun Domic. Antun, please come to the podium.

Antun Domic: Thanks, Nanni. Let me give a few comments with respect to this. I'm looking at this area from the very parochial perspective of the EDA industry, meaning software tools that get used to design ICs. One of the problems with EDA is the very wide spectrum of applications in which it gets used, and the success in using specialized hardware in EDA has been quite limited. I can point out two exceptions: the emulators that are significant in the industry today. These are based on special processors or FPGA boards.

Antun Domic: The second was a company called *Brion* that got acquired by ASML. They did a special machine to process polygons' databases. One can look for other cases, but the truth is success has been very limited. There were attempts to put EDA software on GPUs, and while you could do it, the payoff was very, very limited. Well, some of the reasons for this was that the problems deal with a range of very small to very large data. A SPICE simulator for an analog circuit could be dealing with a few hundred transistors. But if you are doing a DRC for the large chip, you have hundreds of billions of polygons, so you have a wide spectrum. The operations that you do are very different. You may be using super high precision in static timing analysis, but in other places, integer operations are all that you need, and obviously, Nanni mentioned the many NP-complete problems.

Antun Domic: If you look at history, algorithms have been key to the progress of EDA, and not only theoretically but also for the commercial tools. One can point to many cases but I mention only one: cell placement went from using simulated annealing to quadratic minimization algorithms in the early 1990s.

Antun Domic: So, the influence of research on the EDA industry was very direct and one can pick hundreds of cases. Another case you could see is when research enabled completely new applications. The first incarnation of formal verification was really limited to equivalence checking, I'm pointing here to a very famous paper of Randy Bryant. A couple of years later, people in Digital Equipment, IBM, and other places that were fiddling with these types of techniques deployed the first equivalence checkers and the commercial industry soon followed. One can choose other examples such as delay calculation, property checks, RC reduction schemes, and so on. So, algorithms have been very important to the progress. The current CPU story, on the other hand, has not been very positive for EDA because the single processor speed has not improved very much. You get many more cores, and you try to parallelize and do multithreading. I was happy to see that Intel finally is announcing a CPU with a 5 GHz clock frequency. It took a long time for them to get there and we'll see how they do.

Antun Domic: The problem is that there are many algorithms in EDA that benefit very little from parallelization. A second problem is that this parallelization and threading has been extremely costly for the EDA industry because you have to go all the way down to the data structures of the programs to be able to really take advantage of these capabilities but we'll see. Also, the tools have gotten more complex, a place and route tool today has over a thousand commands. The internal flows inside a tool go through many algorithms, and they're very difficult to control.

Antun Domic: An approach that has been mentioned is ML: why don't we train the tool. You could use specialized chips that now are coming out but there are problems. You need a large set of examples for training, as the scope of a problem you address is wide. Why some tuning done for designs on a 28 nm process would work at 7 nm is very unclear in some areas. So for me, full flow training is not realistic now. But if you target narrow areas, you could be very successful in replacing internal algorithms. Advanced chips are a very small percentage of designs. Less than 3% of the designs are in 10 nm or below. So, you don't have huge data, you have the confidentiality problems, and the technology keeps changing. So, why learning for some technology applies equally well for others is a question there. Now you can look at totally new computer architectures.

Antun Domic: You have things like in-memory computing, super low-temperature electronics, and quantum computing. Of course, success would have to be based on being able to cover a wide range of problems as opposed to just one algorithm. The second part is that the IO of any specialized processor, compared to the more traditional processors, will be very important, but also it would force us to reconsider algorithms. An algorithm that was pretty good on a standard computer may not be at all appropriate for execution on a quantum-type machine. The IO problem is very serious, and also the memory access, which is another problem too to look at.

Antun Domic: My comments would finish with, number one, algorithmic development that continues to be very badly needed as traditional computers will do the bulk of the work of the over the next, five years, let's say. Heuristics keep deteriorating as the problems get larger. So, basic research, particularly with parallelization, would be very important. Also, if you could have changes in algorithms to go to, let's say quantum computer, which also would require research. The successful applicability of ML and big data type techniques, in my opinion, can be achieved if you correctly target the narrow parts of flows so that you would get significant improvements there. I think a full flow for complete chip design is not a very feasible thing at this moment. New architectures could be of great help. As I say, if you had quantum or in-memory computing or similar things, the key part for the commercial EDA programs would be how wide a problem you can attack, as opposed to narrow areas such as having just a very fast covering algorithm for logic minimization.

Antun Domic: If you're looking at running a covering algorithm a million times, reducing the time of the covering is not going to help you that greatly, but maybe other areas could be different. So, for me, the true key thing would be wideness of application and solving the IO-type problems.

Giovanni De Micheli: Thank you Antun. The next speaker will be Jason

Jason Cong: Thank you Nanni for organizing the session. Unfortunately, the topic of my talk was dismissed by Antun in the very first bullet. He says, "There's no customization needed for EDA tools." So, let me try to make a case, try to salvage some of these points. The reason we actually need customization is that we are toward the end of the more Moore's Law scaling, and the CPU performance is not getting much better, certainly not doubling every 1–2 years. So, what other ways to improve that? But we still have a lot more transistors. Therefore, we think these transistors can be used to do specialization. Someone may ask, "What about the multicore?" You can do that, but if you do a calculation, even with multicore, very soon you hit the power density limit. You can fit a thousand cores, but you cannot turn them out at the same time, that's the concept of dark silicon, unfortunately.

Jason Cong: So, we argue that it's actually the right time to put in a lot of specialization to maximize the performance. Maybe they don't have to be totally dedicated, they can be programmable. So what's the implication? So, we should revisit some of these old techniques that achieved 100× – 1000× improvements through specialization. Let's think about the domain of EDA, circuit emulation is definitely a good example. We can speed up the circuit simulation by a factor of 100,000 through hardware emulation. That is somewhere around a half-billion to a billion dollars commercial market out there. People buy these expensive emulation machines. What else is possible? This is also an example Antun mentioned: There was some work over 10 years ago to speed up lithography simulation, which my group was involved.

Jason Cong: In deep submicron technologies what you see is *not* what you get. You have to do an inverse transformation to figure out the right mask to have. You have to do this optical imaging process which is much computationally expensive. So, we started with the Hopkins equation and generated an FPGA-based accelerator through our high-level synthesis (HLS) tool, we could actually get 15× speed up and also a 100× energy efficiency improvement. It sounds like a good idea for an EDA startup, but it's a bit too late if you start now. A company called *Brion Technology* provided a commercial solution based on FPGA acceleration doing exactly this and it got acquired by ASML about 10 years ago for $270 million. But we don't have to be limited to EDA. In fact, we have a unique opportunity to enable and allow more people to actually use hardware acceleration techniques to improve performance and energy efficiency in many domains. Let me tell you another example we did that came out of an Expedition in Computing (funded by NSF) project that we had 10 years ago that was working within the medical domain.

Jason Cong: One thing I learned from that project is not to do a computed tomography (CT) scan if you can help it because a CT scan takes about 2000 X-rays of you, and that's very close to your lifetime limit for radiation exposure. You may worry about the radio exposure going through airport security. This is a million times worse. So, we come up with the idea that maybe we can reduce the CT radio exposure using the concept called the *compressive sensing* to do a low-dose CT scan. It's a new and exciting algorithm. Unfortunately, then the computations time becomes very long, it's like 18 hours. You don't want to wait that long for the result. So, this is where we come in with the hardware acceleration. We use a system that has four FPGAs on Intel's front-side bus (FSB) together with a CPU as our customized architecture.

Jason Cong: We did bring it down from 18 hours to 6 minutes! This is a good example to show you there are a lot of acceleration opportunities by actually thinking about algorithm and hardware codesign to make algorithms such as compressive sensing practical. Now the good news is that it's much easier to do this customization. You don't even have to buy an FPGA card and put it into your servers; you just go to Amazon AWS which offers the FPGAs for acceleration.

Jason Cong: Now, I use it for my graduate class and undergraduate class: The students have both GPUs and FPGAs, and you pay a dollar and half for an hour to use the FPGA roughly. You use it (online) as long as you want or as short as you want. The challenge, however, is how to program such a beast? I want to show you that you don't have to always come in with an RTL design. Some of you may remember that we presented a number of papers at this conference on our HLS work in the last decade.

Jason Cong: Our research led to a spinoff company called *Auto ESL*, which is now part of the Xilinx, and the HLS tool is called the *Vivado HLS*. You can start writing C or C+ programs we can compile into Xilinx FPGAs. This is also on Amazon now and you can go to use that. This improves the designer productivity significantly. If you do a Google scholar search for example, you'll see there are 3000 plus papers citing that the Vivado HLS. It's been widely accepted. That's just academic work. There are probably equal or more industry usage which didn't show up as publications. However, I have to admit that even though we are the original inventor, or developer of Vivado HLS, it's not that easy to use. You have to add in a lot of pragmas to do unrolling, data tiling, pipelining, and so on, to get good performance.

Jason Cong: So, we have been working on further improvement of FPGA programmability in the past five years. There's another spinoff out of our research program called *Falcon Computing*. It provides the Merlin compiler which is very easy to use. It supports OpenMP like programming style. If you can program for multicores, you should be able to program for FPGAs. Experimental results show 35× performance gain and 5× to 10× productivity gain. In the current research, we want to make it even easier. You don't even have to start from C, you can start from Caffe, Tensorflow, Spark, Halide, which are all called the *domain specific languages (DSLs)*, for ML, image processing, etc. We have an efficient backend to compile these DSLs into optimized microarchitectures, such as systolic arrays, stencils, and a new class of architecture called the *CPPs* stands for compostable parallel pipelines.

Jason Cong: We also use ML techniques to search to get the best possible result. Our goal is to democratize customized computing for FPGAs and accelerator designs. For the EDA community, this is an exciting time to think about what else we can accelerate beyond circuit emulation for logic verification. Nanni mentioned SAT. I think it's a good idea because if you can do the SAT, it has several applications, such as various kinds of graph algorithms. They are all fundamental to EDA. By offering automated compilation flow for customizable computing, we can enable many more applications domain experts to use hardware acceleration for their application domains.

Jason Cong: We should feel very proud of what the EDA industry has done. When I was an intern at National Semiconductor, this is back in the late 1980s. Next to me, was this row of layout designers doing channel routing, global routing, who are now all replaced by automated tools. Then, later on, I spent some time at Intel as a consultant. You can see they have a room for a circuit logic synthesis designer and now it's all probably replaced by Synopsys tools. So, what we are doing is actually AI, right?

Jason Cong: The funny part is that once we completely automate some highly intelligent human tasks, with comparable quality as human designers, it is no longer called AI. Our field was given a name called *design automation*. They make a big fuss if one can compete with a chess player or play Go. But we can beat the circuit designer and logic designers. Somehow it's not considered AI. Here is a story. I'm on the department hiring committee. When our AI faculty proposed a few names for consideration, I said, "We just hired several ML faculty in the past two years." They said, "Oh no, no Jason, these are not ML faculty, these are AI Faculty." I said, "Well, is there a big difference between AI and the ML?" They say, "Yeah, absolutely." So, I went back to search, what's the difference between AI and ML? The most interesting answer I find out says, "If some code is written in Python, most likely it's ML. If it's written in PowerPoint, that's AI." Unfortunately, we implement on our EDA tools in C or maybe assembly, so we are way beyond AI. Of course, that's intended to be a joke.

Giovanni De Micheli: Thank you very much Jason. I think the HAL computer was AI, the one of movie "a space odyssey." So, let me switch to the third speaker, Massimiliano Di Ventra, from San Diego, University of California at San Diego.

Massimiliano Di Ventra: Thank you Nanni, for inviting me to this panel and thank you all for being here. So, I'll be talking about a new computing paradigm, we call mem (memory) computing. In fact, I will be talking about the digital version of memcomputing, which is scalable. Memcomputing stands for *computing within memory*, and by memory, we don't mean just storage, but generally time nonlocality. The ability of a system to actually remember its past.

Massimiliano Di Ventra: If you're interested, I urge you to look at the literature on memcomputing. I also cofounded the company MemComputing Incorporated which is releasing the software as a service to solve the problems that I'll show you in a moment. These machines are, as I said digital in the sense that they map integers into integers. This is fundamental to actually have scalable machines, but once you know that input and output are digital, there is nothing that prevents us from using anything in between.

Massimiliano Di Ventra: At the moment, we solve problems using algorithms, but we would like to use physics to go from input to output. However, unlike quantum computing that uses certain quantum features we want to use nonquantum systems, and ideally, we would like to have machines that can be fabricated with present technology. That's what I will show you.

Massimiliano Di Ventra: So, as I said in between the input and output we will have a type of physics that uses dynamical systems with memory. Following these ideas, we came up with a concept, we call universal memcomputing machines, which apart from an input and an output, and a control unit that tells the machine what type of program to execute, they perform computation in memory by what we call mem-processors. Dynamical systems with memory execute the processing of information and the information is stored at the end of the computation by those dynamical systems.

Massimiliano Di Ventra: So, the information never gets out of the memprocessors. This is not just mathematics but can, in fact, be built in practice, either with memory elements or even in CMOS that emulates memory elements. Here is an example of an AND-logic gate, and we use electrical circuits to represent it. For example, if this terminal is a logic one and this is a logic zero, then it is not satisfying the logical function of an AND gate. So, this terminal reads the other two terminals and realizes it's in a wrong configuration. It will inject current so as to satisfy the logical proposition of the gate.

Massimiliano Di Ventra: So, these gates, which we call self-organizing logic gates, are not like standard gates because they can accept the signals from both the standard input and the standard output. They're able to self-organize to the correct logical proposition, irrespective of where the signal comes in. This is fundamental because we essentially transform logic into physics so that we can invert literally Boolean problems or Algebraic problems as we have done those as well. These gates can be as I said realized: in practice even within CMOS. So, you don't need special materials. Here is an example of a problem we actually tackled which is a MaxSAT problem where you have a Boolean formula with clauses related to each other by logical ANDs. The logical circuit is then a collection of logical ORs related to each other, and what we do is essentially let this system self-organize to the correct solution.

Massimiliano Di Ventra: Here is an example that actually was not done by us, but by the supercomputer center on a single core using Matlab code. They chose to compare our solver with two of the winners of the 2016 MaxSAT competition. This is the time versus the number of variables. You can see the standard solvers, state-of-the-art solvers, following an exponential curve, but our solver scales linearly and if you extrapolated the standard solvers, it would take more than the age of the universe for a problem with 2 million variables while our solver did it on a single core in 2 hours. In fact, I don't have time to show you all the other problems that we tackled, but we have now a lot of examples, and we tackled several NP-hard problems from MaxSAT to Max-Cut, etcetera, and compared with the winners of the MaxSAT competition and we are always orders of magnitude better than standard algorithms. By the way, we are simply numerically solving the differential equations that represent these machines.

Massimiliano Di Ventra: We are literally using standard computers to simulate differential equations and look for the steady states of these equations, which are the solutions. Then, we extended this to integer linear programming, which is algebraic, it's not Boolean, and we succeeded in solving in 60 seconds, a problem that was unsolved for about 10 years. This was actually certified by the computer scientists that maintain the MIPLIB library in Berlin. In fact, if you go on their website, it says the first feasible solution was found by memcomputing. Then, we looked at accelerating deep learning and we do better in software unsupervised learning than D-Wave in hardware and better than the state of the art in supervised learning. We applied memcomputing also to Spin Glasses where we compared with the standard algorithms and our scales again polynomially, and the company actually is engaged with other companies to tackle several other problems.

Massimiliano Di Ventra: For example, here you see the fifth Airbus loading problem, which is one of the five problems that Airbus put out for quantum computing when it is available. The fifth one is essentially an integer linear programming problem. The person who actually did it could solve it in subquadratic time. Now we have tackled over 100,000 instances of very tough problems, in several classes algebraic or Boolean. So let me conclude. What I showed is that there is a new class of machines we call universal memcomputing machines that are able to compute complex problems efficiently.

Massimiliano Di Ventra: These are physics-based machines and they're nonquantum. So, you have two possibilities to implement them. You can either do it in hardware and as I said you don't need special materials. Of course, if you have, for example, resistive memories that you can integrate with transistors, which would be ideal, but you actually can emulate time nonlocality (memory) with CMOS. Unlike quantum computing that needs to be built in hardware, you cannot simulate a quantum computer on a standard computer efficiently.

Massimiliano Di Ventra: Our machines are nonquantum and so the equations of motion of these machines are simply coupled ordinary differential equations, and we can actually simulate them efficiently in software. I showed you several examples. If you want to do real-time computing, of course, you need the hardware. You cannot do it offline. But, in general, you can solve many of these problems offline. Besides these, machines are very robust against noise and disorder, and we actually showed that using topology.

Giovanni De Micheli: Thank you Max, and the fourth speaker, last but not least, is Martin Roettler from Microsoft and he's going to talk about quantum computing.

Martin Roettler: Thank you very much Nanni and thanks for having me on this panel. Let me just begin by saying that this is a very exciting time for quantum computing and that we are at a juncture where these technologies are emerging and we are beginning to see actual devices. Yesterday, there was a great talk in one of the breakout sessions by Leon Stok who reported on the IBM device. So, you can actually explore these devices already now. Here I'm going to take a slightly different twist on the problem. I'm going to show you what a quantum computer is potentially good for and then I also want to give you a glimpse of the impacts of EDA on quantum computing, hopefully giving you an idea of what kind of problems we can tackle using EDA for the problem of compiling quantum programs.

Martin Roettler: Let me begin by briefly mentioning the type of problems for which a quantum computer is good. This is by no means an exclusive list, but those are the problems that might have a business value. The list includes things like cryptography, where historically those were the first applications for quantum computers and they still are. It's a very active research field, like what impacts a quantum computer has on cryptography. But from the business point of view, that's probably not a big market where we can sell that or offer services to many customers.

Martin Roettler: So, the applications are more likely to be in areas like computational chemistry, questions around how to design chemical reactions, how to design catalysts that have high efficiency and high yields. Those typically lead to hard computational problems. Computational chemist tries to solve these problems but the computations suffer from either being fast but inaccurate or if the methods are accurate, then they scale very poorly with the dimension of the problem, which is exponential.

Martin Roettler: If you have higher orbitals in your molecules, the methods are very, very slow. In material science, similarly, you have problems that could be tackled by a quantum computer. A prototypical problem is the Hubbard model which is a representative model problem and a hard problem for classical simulations. Sort of our guinea pig or our benchmark problem, what we can solve in principle with the quantum computer. Then the big promise is that in areas such as ML, quantum computers can help because they have amazing capabilities for specific tasks. It turns out that you cannot map everything well to a quantum computer. Rather, the quantum computer is good for specific tasks. Research showed that quantum computers are good for finding periods of exponentially long functions. That's amazing but that doesn't seem to have a lot of real-world applications. What's more interesting is the ability to simulate the process that's described by a Hamiltonian and the ability to amplify amplitudes.

Martin Roettler: Finally, you can also invert linear systems of equations exponentially faster than you can do on a classical machine. There are several caveats about that last application. It's not as easy as saying, "Look here's my linear equation please invert it." You have to kind of carefully make sure that the problem needs to be well conditioned essentially, and you must have access to the matrix element of your transform. But if you have that then the quantum computer can give you an exponential speed up.

Martin Roettler: There is a big hope that will help in ML tasks. Okay, so briefly on the chemistry side, this slide is to just give you a flavor that even quantum algorithms, if they are truly inefficient, it will basically not help you. So, for computational chemistry, the typical approach is to map the problem to the Hamiltonian. This is a description of the electronic structure of a molecule. You can describe it in the so-called *Born–Oppenheimer approximation*. Then the chemist asks questions like what are the ground state energies of that molecule? What are the excited states energy energies of the molecule? Those are hard problems.

Martin Roettler: For classical machines, once the number of spin orbitals in your system goes beyond a hundred there's absolutely no hope to get a very accurate estimate of the ground state energy. For a quantum computer, if you have an algorithm that scales poorly, the first algorithm that was found for the problem had a scaling of N to the 11th. It would still take billions of years to solve it even on a quantum computer and that's not good.

Martin Roettler: There were many algorithmic improvements that finally brought it down to a range of a few minutes, and there were several scientific breakthroughs that got us there. It was by no means trivial to go from the simple method that's based on "trotterization" to refined methods that cleverly reorder terms to get a cancelation. So, basically you order the terms so that you can peep-hole a lot of the terms away in the circuit end results. Classical simulations don't go very far, so you can maybe simulate up to like 30, 40 spin orbitals. For really interesting molecules such as a ferredoxin or nitrogenase—nitrogenase is an enzyme used in nature to harvest nitrogen from the air—you actually need that. So, you need about a hundred spin orbitals at a minimum to make a good chemistry model.

Martin Roettler: Briefly, let me mention the crypto side. At Microsoft, we analyze for instance what it really takes to tackle RSA on a quantum computer. We wanted to map out the entire circuit and optimize it and, and see it in front of us, test it and test vectors through and we can do it for RSA. For Shor's algorithm, we can basically test the entire circuit at scale as it is just classical logic, except two Hadamard gates and a phase gate. Then, the rest of the algorithm consists of 10^8 gates, which are all classically describable, namely the so-called *Toffoli gates*.

Martin Roettler: The beauty of that is strictly you can send in a classical test vector and send it to the network and get out a classical bit vector and you can check whether that's the correct one. We did this for RSA and we did it for elliptic curve cryptography, basically for the curves that are underlying most of the digital currencies. These are curves over large prime fields with 256 bit primes. For the estimates, we implemented the entire finite field arithmetic. It turns out you need about 2,300 logical qubits, which means error-corrected and really good qubits.

Martin Roettler: This was recently improved by the Google group. They showed that by further optimizations you can actually solve RSA 2048 with about 20 million noisy qubits. They claim they can do it under certain assumptions on the scaling and the clock speed. I'm not quite sure that you can really solve RSA 2048 in 8 hours, but there is definitely new thinking also around these crypto problems and it has a high impact on that field. These developments lead to a new kind of cryptography that's emerging right now in the United States as an effort by NIST to benchmark this and there's a heavy competition.

Martin Roettler: I think there's still 60 contenders in the competition for the new standards for postquantum cryptography. Other speedups are the already mentioned computational chemistry and linear algebra problems. Also, some optimization problems can experience a speedup. Let me talk about the programming side now. What I mentioned so far were abstract ideas of algorithms. The next step is to program these algorithms and to write code. To help with this, at Microsoft we develop a language called Q#. The aspiration is to be high level and scalable. High level here means that as a programmer you don't want to be bogged down with writing assembly code all the time.

Martin Roettler: When you design a new library, you probably want to work at a lower level and write optimized code and make sure your library is as efficient as possible. But when you then design new algorithms, you don't really want to think about circuits and internals of libraries all the time. You want to think more in terms of "I want to have that operation whenever I want to perform a modular addition" or "I want to have a floating-point operation." For those cases, we have a rich set of libraries already and Q# allows you to plug them together in new and creative ways. In addition to this, there is a framework that's very similar to Visual Studio, and Visual Studio Code that allows you to get on-the-fly incremental type checking and helps you to develop unit tests.

Martin Roettler: There's a lot of documentation available for Q# and it runs cross platform. That means, you can run Q# on NET core, which is open source, and you can run it on Mac or Linux as well. When it comes to new libraries, we look into what the actual domain-specific aspects are. What do we actually need? That is kind of the underpinning, but we also need chemistry libraries. That's why we partnered up recently with a government lab called *PNNL, Pacific Northwest National Labs*. They have this flagship product called *NWChem*. It's a chemistry software for modeling. There are many other chemistry packages out there.

Martin Roettler: NWChem runs on all the leadership machines, you can for instance run it on Summit and other HPC machines. We use NWChem to output a format that we can then read into our quantum development kit which Q# is part of. There you can actually simulate a chemistry model. For this, you need an actual mapping of the chemistry problem to a quantum algorithm. Eventually, you can feed the knowledge about the quantum algorithm back and learn something about chemistry modeling. That means you can in principle get a virtuous cycle. These complex workflows are examples of things you can do to program a quantum computer at scale. You can perform resource estimations, you can analyze the circuits, compute their depth and the critical path. And you can already today investigate the memory footprint of various algorithms. This is important as there is typically a lot of choices of how to implement a given algorithm, even for the problem of mapping a chemistry problem even to a quantum computer. For this specific problem, there are already three or four different choices you can encode the problem at the top level.

Martin Roettler: But then you can pick different algorithms to simulate, and each algorithm has distant different instantiations typically of parameters inside. That is a big space to be explored. Today, you can already do this exploration at the resource level. You can count the number of qubits and the number of primitive gates. The so-called *T-gates* is a very important primitive gate. They are the basic fault-tolerant gate that you need to make a universal quantum computer and they are expensive to realize.

Martin Roettler: Ironically, T-gates are not very expensive at the physical level as they are just a basic pulse applied to one of the qubits. But at the fault-tolerance level, the situation reverses and T-gates are actually very difficult to manufacture from the so-called *T-states*. You need special parts of the chip that create these T-states. This costs a lot of area on the chip. Clearly, we want to minimize how many T-gates are in a circuit. Here, you can already explore the different molecules for various chemistry benchmarks, how many T-gates you would need, and then you can optimize your algorithm and see what the impact is.

Martin Roettler: Another problem where EDA can make an impact on quantum computing is the so-called *oracle problem*. I don't think that's a great name. Oracles are basically subroutines that you need to execute on a quantum computer. Those subroutines are described by classical programs but they have to be executed in a quantum superposition of the inputs. Oracles are important in the context of search problems, for SAT problems and various other problems that have classical descriptions. There are great technologies to synthesize oracles and many are developed in EPFL by Mathias Soeken and Nanni. Also, there are other groups such as the one by Robert Wille at Linz and Dmitri Maslov at IBM does this kind of work too.

Martin Roettler: The oracle problem consists in describing a problem classically as a Boolean function and then translating it into a reversible circuit. There are great tools and compilation flows for that. There are several open-source tools also that you can use to perform reversible circuit synthesis. "Reversible" here means that you can run the circuit forward and backward and it doesn't destroy information. Turning irreversible classical programs into reversible circuits is nontrivial and leads to an increase of memory. The basic issue is that it is not easy to do a memory garbage collection on a quantum computer.

Martin Roettler: Essentially, you have to un-compute whatever you did and return memory in the same clean state that you had at the beginning of the computation. If you don't do that, you cannot have interference in your quantum computer. The technical reason for this is that otherwise the computations would be distinguishable and that destroys quantum interference. This means that the problem of quantum garbage collection is a fundamental one. Early methods such as Bennett's groundbreaking ideas from the 1970s can be applied but they use up a lot of quantum memory. Another method we develop helps to control the amount of memory used.

Giovanni De Micheli: Thank you. I will start with a general question for all the panelists. If you just look forward, let's say 25 years in the future, what are the types of problems that you would like to solve? Would the computers or the computing system that we can foresee be adequate and how will we design them? So, I will ask all of you to give your opinion.

Martin Roettler: Well I can stick my head out first and try to answer it from the quantum computing perspective. Twenty-five years is certainly at a time range where quantum computers might come into fruition, even scalable ones. What I would really like to see at that point is like applications, for instance, in drug discovery. There are problems in drug design that can in principle map to a quantum computer which then, in turn, could explore a big space of possibilities of how to optimize the design of new drugs. Quite possibly, this exploration can be guided by a gradient descent procedure where a quantum computer can help to evaluate a cost function. Today, we cannot evaluate the cost functions as they are hard for classical computers.

Martin Roettler: That problem in itself will have a lot of EDA-related or computer science-related components to it because you need to map it to an architecture where you can eventually run it. Think of the quantum computer of the future of sort of like an FPGA-like structure. Having a two-dimensional mesh of computing elements.

Martin Roettler: You have to somehow go from the high-level description of your algorithm to the actual gates set on your mesh. There are many stages in there, involving several intermediate representations of the program that implements the algorithm. There are software and hardware efforts underway today to define those stages, where there are many hardware platforms, and programming languages. IBM has a language, Rigetti has a language, and Microsoft has a language. There are opportunities there to improve and evolve these languages and to standardize the process. There is a whole economy that can in principle grow around this. In a 25-year timeframe, I can certainly see applications arising within a supporting quantum economy. I think it would benefit our humankind tremendously if these things were to happen.

Giovanni De Micheli: Just to inject some controversy. QC is useful because it uses nature to implement parallelism. The question is: Can we do it in a different way? I would like to ping Max and Jason on this issue to see whether actually we could achieve something similar without having to go down to 20 mK which will make portable quantum computing complicated except in outer space.

Massimiliano Di Ventra: Yes, the answer is yes. We already have shown a lot of results. Indeed, I didn't have time to show it but the parallelism of our machines, mem-computing machines, comes from the behavior of a system that is at a phase transition. So, essentially our machines go through a phase transition and they develop what we call *long-range order*. The ability of the machine to correlate at long distances even though the interaction between the components is local.

Massimiliano Di Ventra: It is a misconception, no offense to the quantum computing guys, but it is a misconception that you need quantum effects like entanglement or tunneling to actually solve complex problems very efficiently because physics allows you to use many other phenomena like time nonlocality, and I'm pretty sure you may actually do it in other ways. The incredible thing is that if you use nonquantum systems like the ones that I just showed then you can do it even in software, because you can actually simulate the differential equations of the physical machines that we are suggesting, unlike quantum computers that require really the hardware to be built at milli-Kelvins, to actually see their full advantage. You cannot simulate a quantum computer on a classical machine efficiently.

Massimiliano Di Ventra: Our machine, on the other hand, can be simulated, and we have now already simulated them efficiently on a wide variety of very tough problems. I would like to see, like Martin, application of mem-computing to material science problems, so drug design would be another one. So essentially trying to find the spectrum of quantum Hamiltonians in general. We are already working on this and probably we don't need 25 years to succeed. Again, because we actually have now physical systems that use a type of parallelism that is not entanglement and it doesn't need cryogenic temperatures to actually survive.

Giovanni De Micheli: I would actually put the question back to Jason. Do we need completely new architectures to exploit accelerators or could we just build them around standard system architectures?

Jason Cong: First to get back to the first question about what's the architecture in 25 years? That's a long time. It's very hard to predict even beyond 10 years. I'm quite sure that two things will stay. First, heterogeneity, as I don't think that one type of technology or one type of device will dominate. Second, these devices have to be programmable. At UCLA we had been advocating for quite some time, we use this term called *accelerator centric architecture*.

Jason Cong: The von-Neumann-based processors will do very less work because it's inefficient, even though highly programmable. Our objective is to offload tasks to accelerators. In my opinion, accelerators are just like experts. So, quantum computing will never replace a standard microprocessor, but it's a very powerful accelerator, so are some of these in-memory computing technologies and neuromorphic computing.

Jason Cong: I'm passionate about FPGA, just because it is programmable. We can argue whether it has the right granularity. Maybe it's too fine grain, maybe we can add some course-grained computation. Maybe it doesn't have to be all synchronous, maybe it can be asynchronous, maybe it can be event-driven and all of those, but it has got to be programmable. The challenge is how do I decompose complex computations in an intelligent way so that I can make use of all these accelerating technologies.

Jason Cong: Think about the job of CTO of a large corporation, Antun. You have all these experts: some people know how to do the engineering, some people know how to do marketing, some others how to do the sales. When something comes in and you say, "We've got to increase the revenue by another billion. How do you achieve that?" So, the decomposition of the problem into the hands of these experts is the key.

Antun Domic: Well, let's look a little bit at history because some of us were in this business 25 years ago. In the early 1990s, I used to work at the Digital Equipment microprocessor group and it crossed the 1 million transistors mark, and then Intel, IBM, everybody crossed that magic numbers '91, '92, and '90. Today, they are that 10 billion plus transistors, so it's four orders of magnitude in 25 years. Given that we don't have to use all of them, Jason was mentioning dark siliconand all that. For example, the Intel processor that would cross 5 GHz in all processor systems is a chip with only eight processors. They have been restricted due to power.

Antun Domic: So, if you assume that we would go to another four orders of magnitude in 25 years in whatever a transistor is in 25 years, if you don't have a radically different power structure, which means that you are going to have only some processes operating. In my opinion, it should open the opportunity to put specialized processors and maybe configurable processors directly on a chip base. Then, they would be much more widely available as opposed to what you're doing today in things like Amazon with a CPU and FPGA's and all that.

Antun Domic: Maybe that will be all done on an on-chip and with much more standard programming. The accelerators, as they have been mentioned, like a quantum computer and other things will be the auxiliary pieces that are important but will be used to offload certain things. It should open possibilities in totally new areas that I'm not going to speculate on, but, for example, medical applications should be something where more computing could make a significant difference from that perspective.

Giovanni De Micheli: Let me just pose another question to the panel and then of course. I would like to address the problem of bootstrapping. In the past, we have designed processors to design newer processors. Now we can have accelerators to design better accelerators. Can we use other technologies, for example, QC or memcomputing, to design more powerful QCs and mem-computers respectively?

Antun Domic: Let me give you a quick one. For example, suppose you have a quantum computer that solves some specific problems in EDA. Let us consider placement today in a large block. Today, place and route is limited on a flat basis on the 10 million cell numbers. Placement for a block of that size takes between one and two days.

Antun Domic: If you could replace the placements step by something that takes 10 minutes, not only it would be worth paying the overhead of dumping the data and bringing it back, it would be a significant improvement, and it would offer possibilities that are never explored today. Because if you are going to spend two days doing placement, you are not going to explore too many floor plans and things that could give you better results. So not only it will be very beneficial, but it would open possibilities that people cannot exercise today.

Massimiliano Di Ventra: Yeah. Following up on this, I fully agree. I think that maybe mem-computing could help as several levels in EDA by accelerating substantially certain tasks right? Routing for example and others. I see that. Not being myself an expert in EDA that's the best I can tell you in that respect. I know in general terms what it is, but it's worth trying to use it in specific parts of EDA to accelerate it.

Antun Domic: The differential equation part is very critical in circuits simulation, for example.

Massimiliano Di Ventra: Yes. Following up on this, I fully agree. I think that maybe memcomputing could help at several levels in EDA by accelerating substantially certain tasks. Routing for example and others. Not being an expert in EDA that's the best I can tell you in that regard. It's worth trying to use it in specific parts of EDA to accelerate it.

Antun Domic: The differential equation part is very critical in circuits simulation for example.

Jason Cong: Even at a higher level, you write a C program and compile it into FPGAs. You can decide where to insert pragmas, but how do you come out with the right architecture? So, this is the part that we are trying to automate and that this is also itself an area in need of acceleration because today we limit it to run for half a day. Then using reinforced learning and using some large-scale optimization, it actually can become very competitive.

Martin Roettler: Great question. From the quantum computing point of view, let me first build on Antun's answer. In principle, we can experience quadratic speedups over a classical algorithm that solves the same underlying problem. It is perfectly reasonable to expect that for constraint SAT problems you can obtain such a quadratic speedup. The other applications I can see are more specific to synthesis tasks for approximating unitary operations.

Martin Roettler: The issue here is that quantum computers cannot perfectly execute all the gates that are feasible. It's not a continuous kind of an analog computation. You have discrete gates set, and you can use that to make finer and finer approximations. That leads to number-theoretic problems, in particular, if you want to find the approximations fast. Those problems, in turn, benefit from a quantum computer. If you had one already, you could actually solve number-theoretic problems like factoring. In fact, one has to solve quadratic equations over finite fields, but you can reduce them to factoring. If you had already a large quantum computer, you can factor more efficiently and improve the compilation step, which is a kind of a bootstrapping idea. But other than those specialized applications, I think it's not likely that we can speed up arbitrary computations in the quantum computer.

Giovanni De Micheli: We are taking new questions from the audience. Please use the mike and introduce yourself.

David: I am David Pan from UT at Austin. I like to challenge Antun's comment about placement. You said the largest blocks for placement have 10 million cells and takes one or two days to place.

Antun Domic: You can do 10 million cells, meaning counting a standard cell as a movable object. Today between four and seven days depending on how complicated the rules and the final refinements. So, if you look at that, and you assume that the placement itself takes 20% of the time, you see a very attractive target there because it is also an energy minimization problem on a space that has a very full range of peaks and valleys as you move through. But anyways, that's the number.

David: Sorry. I guess my main point is that actually our group recently made some really nice progress, which will be reported tomorrow at tomorrow's session. We developed a new placement engine called *Dream Place*, 10 million cells in 5 minutes, state-of-the-art result. One of the main reasons why we can do this is that we cast this problem into equivalently solving a neural network learning problem, so we can greatly leverage the GPU hardware and software.

David: Then also it is massively parallel

David: But now with this we can actually get a lot of parallelization, right? So, I just want to point that out. In this sense, we recast the problem into acceleration, leveraging the current acceleration that the community has spent a huge amount of effort around both hardware and software.

Giovanni De Micheli: Let me inject another question, that's about the QoR. There are two parts to this question. First: How much would you pay for having 5% better QoR. Second: what about computing time and what about time spent in developing software for this.

Antun Domic: I'll give you my opinion irrespective of affiliations, but if you look at the issue that I did mention and look at the public data. The number of chips that are being designed at 10, 7, and 5 nm is relatively small. But if you look at the percent of silicon that this small number of chips take in the world, how much is manufactured, it is about a third of all chips. So, and that percentage has been increasing, meaning a third of all silicon area, and the percentage keeps increasing. So, if you went to the companies that are designing chips that are going to have 100, 200 million copies, 5% reduction in area, or 10% reduction in power or so is huge.

Antun Domic: I will not quote numbers, but the total semiconductor industry revenue was about 450 billion last year. So, you can do the calculation of how much people would pay for a 10% reduction in area, for example, if you went to Apple or Qualcomm.

Jason Cong: Let me offer a different view. I mean for ASICs I totally agree with you. But then that's where the design starts have decreased. I am quite different for FPGAs. Unfortunately, vendors like Xilinx or Intel have this terribly slow place and route tools that take days sometimes, to just finish the placement and routing.

Jason Cong: In the past, Auto ESL company has been spun off, and it got acquired by Xilinx. Another company, Neptune Design Automation, had the goal to speed up place and route. So at the time, I told them that for every 5× gain you get in QoR you can degrade tool performance by 5%. But then this is taken care by accelerators.

Jason Cong: So, David your example is fantastic, but I just suggest to forget about ASICs and do it for FPGAs. I want to compile circuits at the same speed I compile for processors.

Giovanni De Micheli: QC should give us an opportunity to solve problems exactly. Unfortunately, it doesn't run right because mother nature is not nice to us and there are issues like coherence and noise. What is the roadmap to make QC robust and use it to enhance QoR?

Martin Roettler: That's a great question. In a quantum computer, you have that choice of where in your computation you want to be highly accurate and where you don't. This means you have a choice where to distribute precision in computation. Finding the optimal distribution of precision is a nonconvex optimization problem. We did some initial research of the problems and found that the particular choice of precision distribution can make a difference of several orders of magnitude in terms of circuit size. Precision here is the target approximation of the given unitaries but it can also mean how many bits of floating-point precision you want to give your arithmetic.

Martin Roettler: Besides this, there is also the precision of the rotations as in how accurate you want to do them. In some parts of the algorithm, there might be a lot of slack and you don't need a lot of precision as it will not impact the final output by much. In other parts of the algorithm, you might have to be really precise if you want to have a good answer. That's an interesting problem from the synthesis and design point of view. Generally, it is helpful to think of a quantum computer as a sampler.

Martin Roettler: A given run may or may not give you the answer and often you can check the candidate solution. This works, for instance, for factoring where you can multiply the candidate solution out to check it. On the contrary, for optimization problems, we just care about a good enough answer. In that sense, a quantum computer can also be made to operate similar to a classical annealer. Ultimately, you can experience some speedups on top of the classical methods, as Max mentioned earlier. For some problems, we can show that we get a quadratic speedup over the classical annealer.

Giovanni De Micheli: Max, you're building systems with feedback that converge to stable states. So you have coupled dynamical systems. So, isn't this reinventing analog computing?

Massimiliano Di Ventra: With a major difference, the input and output are digital. So, you can read and write with finite precision. If it didn't have that, and you had real numbers that represented both input and output, then you would need infinite precision in principle, which makes the machine susceptible to noise. That's why I showed only the digital version of memcomputing machines. So, the input and output are digital. That's fundamental, otherwise you wouldn't have scalability. Of course, you can do memcomputing also in a fully analog way, but it's not scalable. It would be susceptible to noise.

Giovanni De Micheli: Robert, ask your question.

Robert: I am Robert Willie from JKU in list Austria. You talked about the vision and what will happen in 25 years. If I understood you correctly, right now the main problem is how to formulate the problems for the corresponding technologies. Let's assume in 25 years you have resolved current problems, maybe with push-button methods. For example, I will push a button and I get the best possible solution for a particular technology. Do you think then that the next problem will be to actually decide what technology to use for what application or is it already clear and or will it always, always be clear that for a particular application, I'm going to use a specific technology, such quantum computer, and for another application, I'm going to use another technology? Do you think this is a problem which can emerge in the next years?

Jason Cong: It is a real problem, even today, forget about 25 years into the future. I have two students sitting in the room here. They have to answer this question all the time: "Shall I use GPU to accelerate or I use an FPGA to accelerate." We have made some good progress by now. So, an important key issue is whether to adapt the architecture to the application, right? But how do we adapt, which architecture do we choose?

Jason Cong: So, you don't want an application expert to change the code too much. It must stay at a higher level.

Robert: Then, if I may, is this a problem that we can automate in the future or that needs to be addressed by the designers themselves?

Robert: I think it depends on how the EDA industry wants to grow. EDA should not just focus on the existing \$200 billion semiconductor industries. There are several trillion dollars in the computing industry to look at it. EDA companies should broaden their views.

Martin Roettler: I can offer a take from the quantum side. I really like your question. It's a poignant question. How do we even express new ideas at a high level? It might be really hard to program these future devices because we just don't know how to do that. How do we even express a new quantum algorithm? In that sense, the entire quantum computing effort could be slowed down as it is hard to find new applications and in addition, it is hard to express them conveniently in a programmatic form. One idea maybe to help with that will be to have a language that allows you to design like even across different technologies. It could be a language that is so high level that you don't have to worry about the underlying compute fabric, whether it is an FPGA or a quantum computer. I'm not sure if that will ever be feasible, but there is hope.

Martin Roettler: The reversible synthesis problem is another example where a programmer can express new quantum algorithms without ever having to worry about quantum computing at all.

The compiler takes care of that, and it translates from the classical program to the quantum circuits. Of course, to do this compilation right is nontrivial, and there are also certain restrictions of the classical program: for instance, there might be only limited support for concurrency and there will probably be no interactive user input. Also, in contrast to standard computing, in quantum computing, all circuits are combinational as otherwise you would have to unroll a sequential computation and undo it. This leads to many restrictions on classical programs. On the other hand, reversible synthesis is a powerful concept as it enables the programmers to write quantum programs even though they don't have to be so familiar with the underlying quantum computing elements, which is really you cannot expect that from a general programmer.

Antun Domic: If we really had machines of any type with significantly more compute power, we should move upward in the level of abstraction. There have been successes with C-based synthesis for FPGAs. On the ASIC side, the success has been much more limited. Verilog was introduced in 1987 and yes, there was SystemVerilog including the test bench and a lot of progress but it's still very much Verilog that drives the ASIC hip design downward. Maybe something else there would be much better.

Giovanni De Micheli: One more question and this relates a little bit more to EDA, but we can go beyond that. So, what problems are there for which we know that the existing solution is close to the best and which are the problems for which we know that we are still very, very far away from what we could achieve? For example, the two-level logic minimization problem is virtually solved. About 95% of the benchmarks that are relevant to engineers can be solved exactly, despite the fact that the covering problem and is NP complete. But there are some other problems for which we don't know how far we are from the best. So, what is relevant actually? What is important to us?

Antun Domic: Let me give you a couple that would be important. You still do a lot of circuit simulation that involves solving the differential equations. This is done more than people think because at the end you do need a reference. That's why many of you could claim that a SPICE-like tool has the accuracy and all that, but the speed, you always want it to be faster. There are other areas, particularly logic placement for example, if you have, millions and millions of cells, just the combinatorics tells you 10 million factorial combinations so forget it. So, there are spots where you have huge opportunities to get, as Martin says, to better solutions, not necessarily the optimal, but something much better.

Antun Domic: Problems in formal verification continued to show up in practical situations, where some equivalence checker ends up stuck after days.

Jason Cong: Nanni, that's a very good question. First, if you were to analyze it from a theoretical point of view we get stuck very easily because most of these problems are shown to be NP-hard and then you basically throw up your hands since we cannot do very much with it. But remember we are all solving a problem up to some constant known size. For example, we have 10 billion transistors in large chips today, as Antun said. If I can design with 10 billion transistors in a good way, then I'm happy.

Jason Cong: But then how do we know whether the place and route algorithm and the logic synthesis algorithm are doing a good job for that. So, we want to measure that numerically, but that's also very difficult because you cannot get optimal solution using branch and bound and they using whatever ILP to get that. Antun reminded us that we did this work 15 years ago. Using actually a quite clever method, we created a set of circuits, we call them *PEKO* circuits for placement, which stands for placement examples with known optimal.

Jason Cong: But the question you're asking is very important, what we have to measure and then how to improve it. If we cannot measure, we cannot improve or even do not know where we are. So, I can tell you that the real challenge is to design accelerators.

Jason Cong: Just by looking at the ML accelerator, we have maybe one of the first contributions in the area and we ask ourselves if we can get another 2×, 4× improvement. But I know it's very natural for us to ask the question about how far away from the optimal are we. There's very little theory even to establish some kind of the complexity hierarchy for you to argue, right?

Antun Domic: Yeah, but that Jason it says the following. Even though we do have solutions for these 10 million cell circuits, given the size of the search space, we don't know if we are 50% from the optimal or 10% or 90% but I would bet that we are in the two-digit range from the optimal. But there is room for optimization that would be extremely appreciated, as it relates to profits.

Jason Cong: By the way, those circuits are still there, you can welcome to download but just try those.

Giovanni De Micheli: Just as a side comment. For example, at EPFL we keep up a public site where we have benchmarks, libraries, tools, and solution sets. For some problems, we keep track of the best solution obtained so far by the community. Everybody interested is welcome to upload benchmarks and try to beat the current solutions.

Giovanni De Micheli: We are coming to the end of our discussion. I would really like to thank the panelists for their time and effort in preparing this panel. I'd like to thank the audience as well. Let us give the panelists a round of applause. ■

About the participants:

Giovanni De Micheli *is at EPFL, Switzerland.*
Antun Domic *is at Synopsys.*
Massimiliano Di Ventra *is at the University of California San Diego (UCSD).*
Martin Roettler *is at Microsoft Research.*
Jason Cong *is at the University of California Los Angeles (UCLA).*

■ Direct questions and comments about this article to Roundtables editor David Yeh; david.yeh@src.org.