Design tools for quantum computing

Robert Wille, Professor at the Johannes Kepler University Linz and Software Competence Center Hagenberg, walks us through design tools for quantum computing

Our world is permeated by electronic systems: from personal computers to smartphones, from cars to industrial machines. The groundwork for this has been laid down by engineers and scientists several decades ago who built the first electronic circuits. Over time, those circuits became smaller and more powerful – eventually leading to their ubiquitous presence in all areas of our lives. Today, those systems are made out of millions (often billions) of components which is why we need computer scientists and efficient tools to properly design them. Without dedicated programming languages, compilers, synthesis tools, verification and testing methods, as well as debuggers, the development of your next smartphone or the next AI solution would not be possible.

The power of quantum computing

At the same time, we are currently at the dawn of a new computing age. With a long history of theoretical considerations and underpinning, physicists and engineers are making leaps in actually building physical quantum computers. These machines exploit quantum mechanics to solve many important problems faster than any conventional computer ever could. Since the physical realisation of quantum computers are currently in the noisy intermediate-scale quantum regime, not all conceivable algorithms are executable (yet). Nonetheless, many promising near-term applications exist, for example, in chemistry, finance, and machine learning. In the long-term, further applications in cryptography, database search, and more will become viable.

"We need design tools for quantum computing. Otherwise we may have powerful physical quantum computers, but no proper automated means to exploit their potential."

The power of these machines comes from the exploitation of quantum mechanical effects such as *superposition* (different configurations can be represented at the same time) as well as *entanglement* (configurations of different parts in a system can influence one another). While these effects are the main reasons for the superiority of quantum computing in many fields, they also create new challenges in the design. This affects how we currently conduct design automation for quantum computing or, more accurately, how we do not.

Established approaches and solutions for conventional design, such as programming languages, compilers and verification tools, are not applicable to quantum computers. Often, the design is still done manually, in tedious and error-prone processes thus far. Continuing this way will lead us to a situation in which we have powerful physical quantum computers, but no proper automated means to exploit their potential – the dreaded design gap.

Design tools to utilise quantum power

The research teams at the Johannes Kepler University Linz, Technical University Munich and Software Competence Center Hagenberg develop design automation methods and software tools to help keep the design gap as small as possible. They get their inspiration from design automation of conventional systems in which corresponding tools and methods proved hugely successful. A similar success story is anticipated for design automation for quantum computing. However, merely adapting existing (conventional) methods will not cut it for quantum computers. Instead, the different computational primitives, limitations and gains have to be addressed. How this can be accomplished is briefly sketched by the following typical design tasks:

Simulation is one of the core tasks in design automation, especially in the early development of quantum algorithms. While comparatively easy for conventional digital systems, the description of quantum states or quantum operations require vectors and matrices, respectively, that scale exponentially with the number of considered quantum bits – leading to a memory complexity that brings even today's biggest supercomputers to its limits. Sophisticated data structures such as decision diagrams drastically reduce the required memory, in many cases by exploiting redundancies in those descriptions. Experiments showed that, for certain experiments, this can cut the required memory by multiple orders of magnitude – including instances in which a simulation could be optimised from requiring 32 gigabytes of memory to just 50 megabytes.

Compilation aims at translating a high-level description of a quantum algorithm into a sequence of commands the quantum computer understands and can execute. This is similar to the compilation of high-level programming languages into machine code in the conventional world. But while we have decades of experience with conventional compilation, available solutions for the quantum realm are still in their infancy and have lots of room for improvement. Again, design automation techniques can draw inspiration from the conventional world (for example, solutions to scheduling, placement and routing problems), to provide efficient approaches for the compilation of quantum circuits.

Finally, **verification**, or more precisely, the subtask of checking whether two quantum circuits realise the same functionality is commonly required to check whether the result of a compilation step is realising the same functionality as the originally provided circuit – a crucial requirement ensuring correctness of a design flow. Again, suitable data structures such as decision diagrams help to reduce the memory complexity, since the representation is similar to quantum operations in the simulation task. However, by additionally exploiting characteristics of quantum computing (in particular, its inherent reversibility) the problem can even be tackled without ever building up a representation of their entire functionality – avoiding the requirement of an exponential amount of memory in many cases. The long-term vision is to have software for integrated development where the result of a compilation step is automatically checked for equivalence, ensuring a correct result.

This is only the beginning

Promising preliminary results have been attained and incorporated into several open-source tools available to all researchers and engineers in the field (see box below). But corresponding evaluations and case studies also unveiled further challenges and obstacles to overcome – motivating to continue the work towards utilising design automation for quantum computing. The final goal is to get a comprehensive software stack to aid designers in efficiently realising their quantum application. At the same time, we are aiming to build bridges between the design automation community and the quantum computing community to establish a common language and facilitate exchange between both communities. We hope to lay the foundation today, so we can avoid the design gap in the future.

This project has received funding from the European Union's HORIZON 2020 Research programme under the Grant Agreement no. 101001318.

The tools described in this article are publicly available under an open-source license at https://github.com/iic-jku/. Furthermore, a web-based (installation-free) graphical interface showcasing decision diagrams can be accessed at https://iic.jku.at/eda/research/quantum_dd/tool/.

Robert Wille
Professor
Johannes Kepler University Linz and
Software Competence Center Hagenberg
Tel: +49 176 23 44 09 64
mail@rwille.de
www.rwille.de
www.twitter.com/rbrtwll