Mapping to Photonic QC Devices

Draft 2 (February 12, 2015)

Fine Print: This document is not finished, but you're welcome to read it anyway. It has not been peer-reviewed, or even thoroughly researched. These are my own notes and speculations, with shiny pictures added. If you have questions, comments, or really good jokes, please contact me at ej@machinelevel.com

Overview

Silicon-based photonic quantum computation devices are one of the most exciting QC innovations to date. They are inexpensive, silicon-printable, and may have a very wide variety of applications in the future. For example, if such a device manages to integrate photon generation and detection, then it may be printed in unused space on an existing digital logic device, giving that device some limited quantum computation ability.

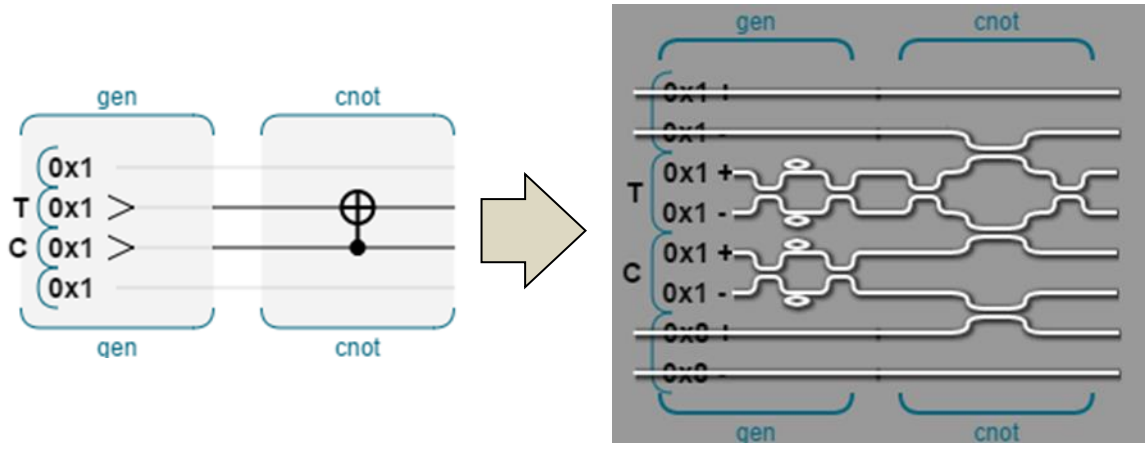
The purpose of this informal whitepaper is to explore the process of mapping QC programs onto photonic devices as I currently understand them, so that QC programs of arbitrary complexity may be (someday) physically constructed via quantum photonic etched circuits.

Note: As of 2015, much of this is impractical, and involves a scale of complexity beyond what's reasonable to fabricate. That's the point; when the devices are ready to handle more complexity, we'll be ready to make use of it.

A live functional sim of this whitepaper's examples is here: http://qc.machinelevel.com/qcengine_unit_tests_staff.html

Existing Photonic CNOT Gates

The paper *Silicon Quantum Photonic Circuits for On-chip Qubit Generation, Manipulation and Logic Operations* (Silverstone et al, 2013) provides a good summary overview, which will be used as a basis for the following explorations. There are several related papers which discuss silicon-based photonics or QEO (quantum electro-optics). In general, they currently (as of 2015) achieve a single CNOT gate by etching waveguides on silicon which look something like this:



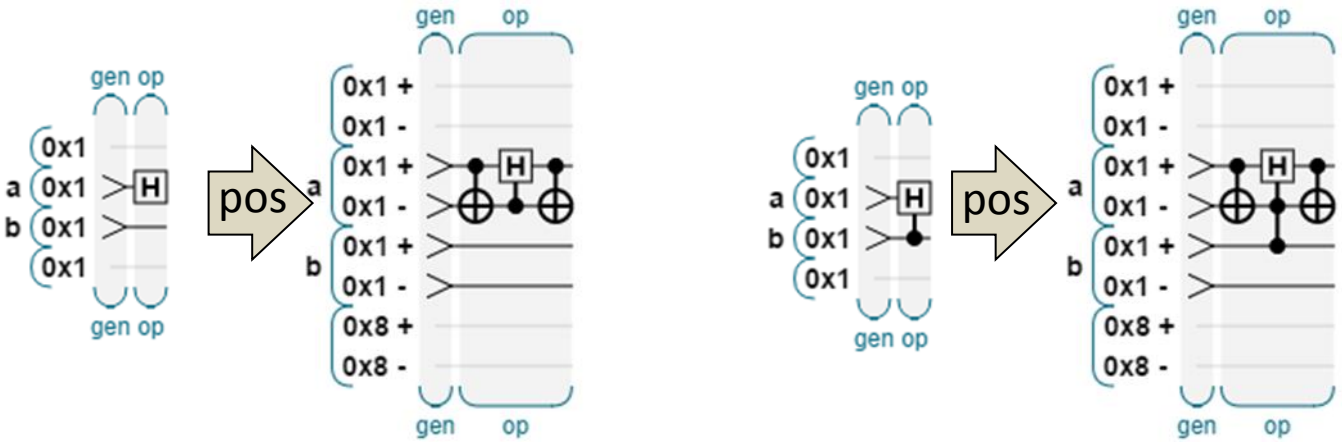
The odd and enchanting look of the silicon version forces the introduction of some new concepts.

New Concept #1: Position Encoding

Each qubit is encoded by a the presence of single photon, on one track or the other. I've marked the tracks + and –, though this is not a standard convention. If the photon is on track+, its value is considered to be |1>. If it's on track–, the value is |0>. **If it's on both tracks, or neither**, this is an invalid state, and the computer is not operating correctly.

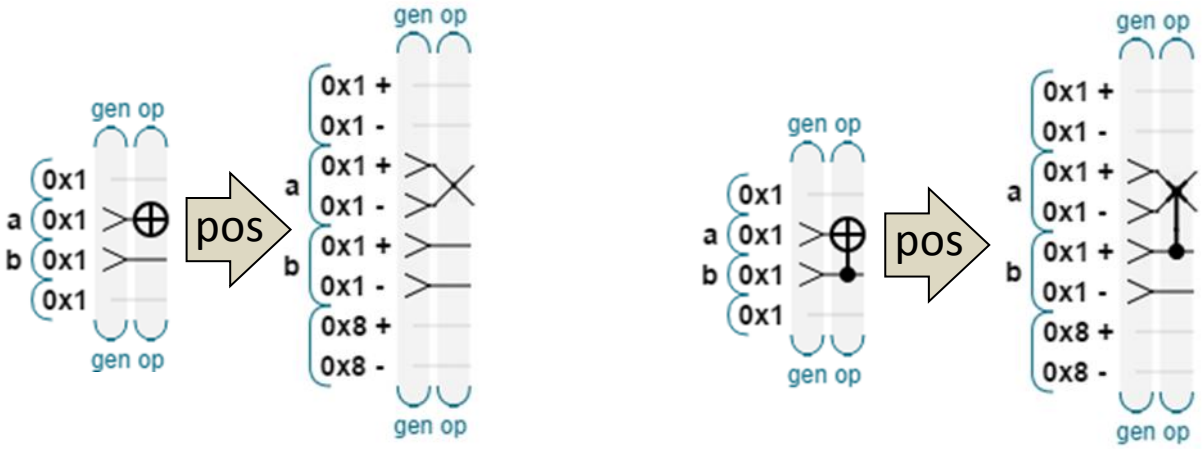
With this in mind, the silicon CNOT gate above is actually sort of a CEXCHANGE gate. If the C photon is present, the T photon crosses from + to –, or vice versa, for whatever superposed values exist for it.

To generally convert a QC program to position encoding, I've been able to use cnot-brackets to exchangify the operation, like this:

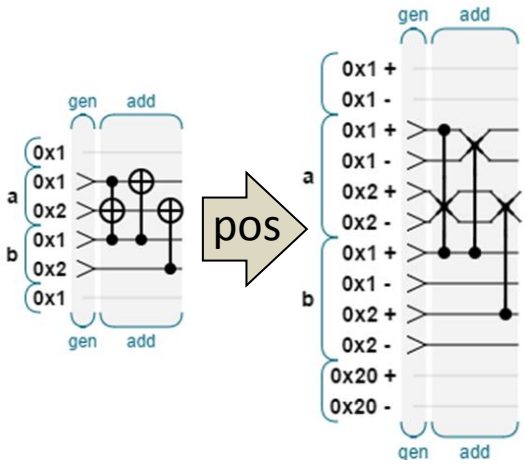


To discover and test this technique, I used the Circle Paper mechanism, along with the QCEngine simulator. **Note**: If you're *positive* your machine is never in an invalid state (see above, any +/– pair in both-or-neither state), then the condition dots on **a–** in the diagram above are not actually necessary. However, if they're excluded while the machine is in such a state, the gate results will be incorrect.

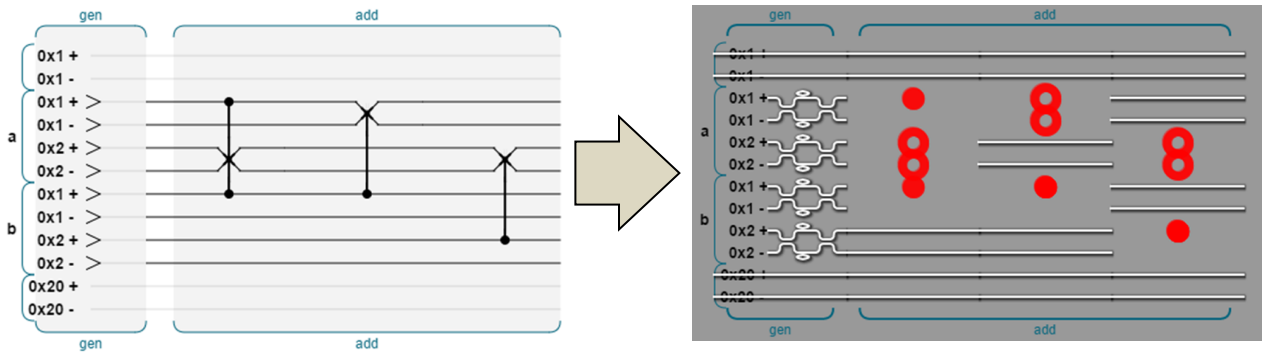
Conveniently, if the operation is a NOT, it simply turns into three CNOT gates, which are equivalent to an EXCHANGE gate.



So here's the same conversion applied to a QC program which performs 2-bit addition:



If we use QCEngine to try to etch this in silicon, some problems will be highlighted automatically:



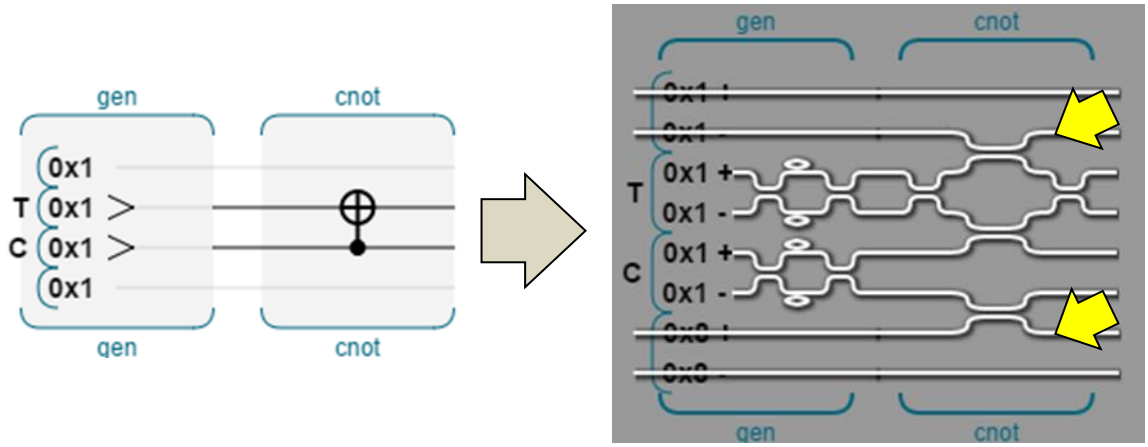
None of our exchange gates can be etched yet, and there are two different problems.

- **Issue #1: Adjacency** – The qubits are not adjacent, so there's no way to bring the silicon traces close enough to cause quantum interference.
- **Issue #2**: **Multi-Condition** – The first gate has two condition bits. Our lives (and this whitepaper) would be greatly simplified if we had a Toffoli-exchange (or even a normal Toffoli) gate, but for now I'm going to assume we have no such thing.

We'll solve both of these, but before we do that we need to make a couple of blind (but useful and probably reasonable) leaps.

Blind Leap #1: Those Extra Bits

We'll do the most complicated "blind leap" first. On the first page of this whitepaper, you may have noticed that two of the waveguides used in the photonic CNOT gate aren't actually part of either of the gate's logical bits:



This style of CNOT (or CEXCHANGE) gate is referred to as non-deterministic. If I understand the literature correctly, these tracks balance the device (and each photon's available options), and if a photon is ever detected on one of them, then the computation failed. The only valid input states to the pictured device are (just looking at the six tracks involved):

	T		C			
	+	–	+	–		
0	1	0	1	0	0	(valid)
0	0	1	1	0	0	(valid)
0	1	0	0	1	0	(valid)
0	0	1	0	1	0	(valid)

Remember, the only valid machine states are ones with a photon in *either* + or – for each qubit. Just inspecting the topography of this gate, and assuming the CNOT gate cannot generate photons, but might accidentally absorb them, the possible outputs appear to be every combination of 0, 1 or 2 photons:

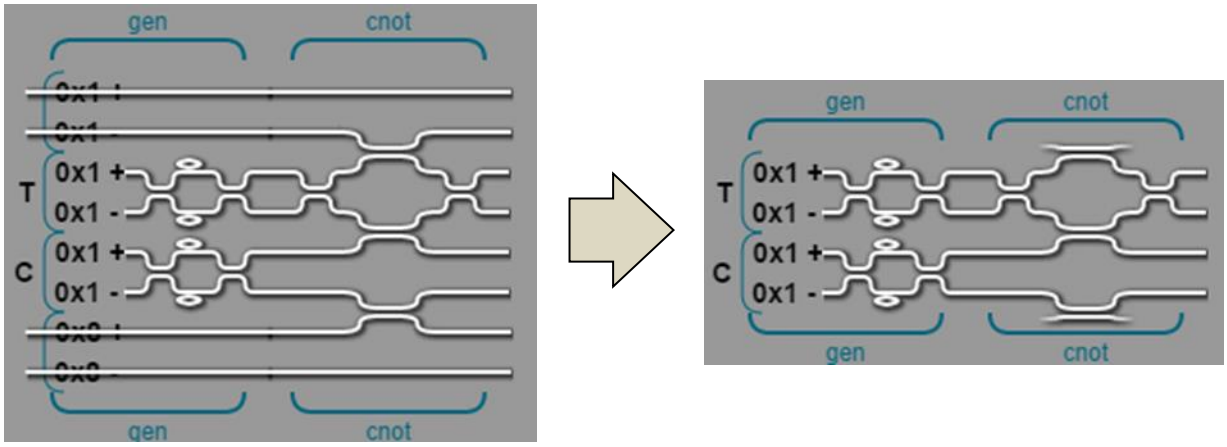
	T		C			
	+	–	+	–		
0	1	0	1	0	0	(valid)
0	0	1	1	0	0	(valid)
0	1	0	0	1	0	(valid)
0	0	1	0	1	0	(valid)
1	1	0	0	0	0	(invalid)
1	0	1	0	0	0	(invalid)
1	0	0	1	0	0	(invalid)
1	0	0	0	1	0	(invalid)
1	0	0	0	0	1	(invalid)
0	1	1	0	0	0	(invalid)
0	0	0	1	1	0	(invalid)
0	0	0	0	1	1	(invalid)
0	0	0	0	1	1	(invalid)
0	0	0	1	0	1	(invalid)
0	0	1	0	0	1	(invalid)
0	1	0	0	0	1	(invalid)
1	0	0	0	0	0	(invalid)
0	1	0	0	0	0	(invalid)
0	0	1	0	0	0	(invalid)
0	0	0	1	0	0	(invalid)
0	0	0	0	1	0	(invalid)
0	0	0	0	0	1	(invalid)
0	0	0	0	0	0	(invalid)

The literature I've found so far calculates that this particular gate has a probability of functioning correctly of 1 in 9. So each time you run a pair of photons through it, there's about an 11% chance that you'll get a valid answer. If it's valid, however, then it will also be correct (meaning the quantum CNOT was performed).

Assuming we try to build our 3-gate adder from the previous section, those probabilities multiply, giving us a 1 in 729 chance of correct functionality. Photons are inexpensive and quick, so we can run it a jillion times and watch for correct answers, but for a machine of any real complexity (say, a million gates) this is **a huge problem**, to which I do not yet know a solution.

The result of all of this is that these two "extra" tracks can be used to detect errors, but if they ***ever*** see a photon, an error has occurred, and the computation is not valid. They aren't actually *part* of the computation, logically.

So here's the Leap: Since they're not actually part of the computation, and each extra qubit doubles the resources required by the QCEngine simulator, I'm going to sweep those extra tracks (pretty much literally) under the rug.

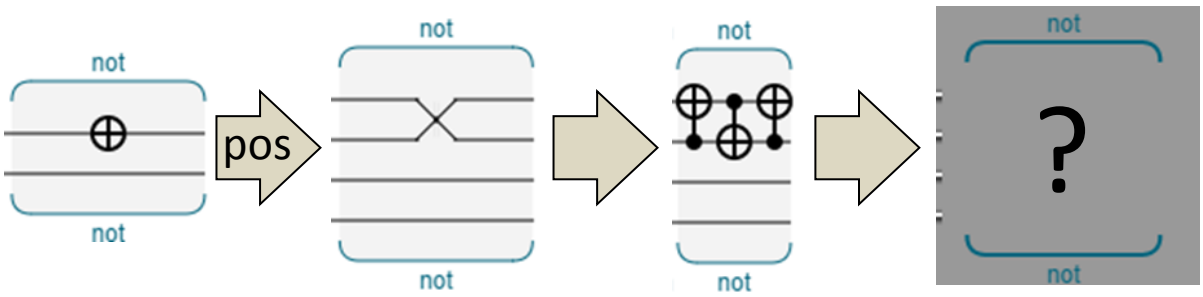


Notice that I'm not comfortable ignoring the issue altogether, so I've left some "placeholder" marks there, sort of like the error-bits are magically submerged into the silicon, and surface when they're needed. It's not actually that unreasonable; these could be little 3D silicon "staples" there to dump stray photons into an error detector, without actually needing to take up extra space on the chip.

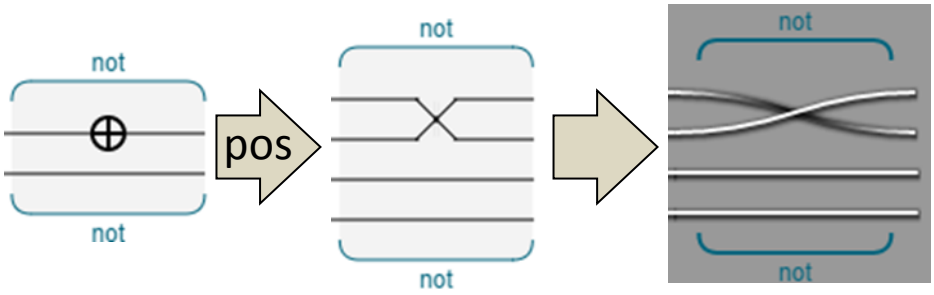
I mentioned this is a blind leap. Let's move on.

Blind Leap #2: Twisted Waveguides

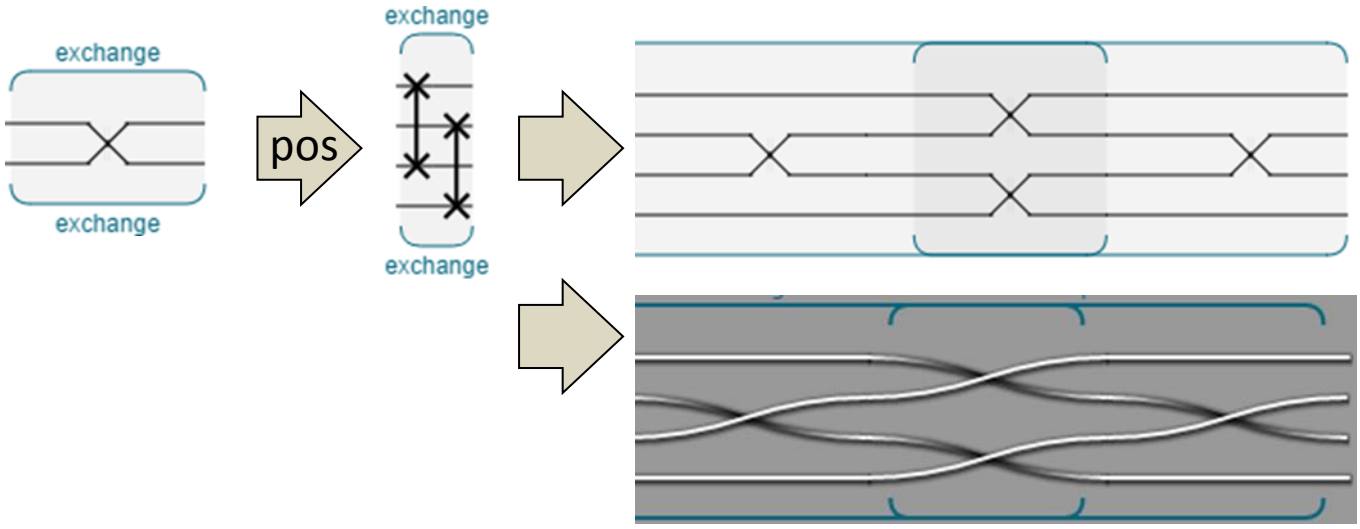
This one is pretty simple. As we've seen, the position-coded version of a NOT gate is an exchange. If we need to etch everything in a single layer of silicon, we could try to use three CNOT gates. Besides adding complexity, though, we don't actually *have* a CNOT gate. We have CEXCHANGE.



…so for the purposes of this casual whitepaper, I'm going to use the simplest possible exchange: twist the wires. This requires some way to create a two-layer device with smooth transitions between the layers, but this is likely to be simpler than most alternatives.



Using this, a logical exchange (which maps to two exchanges), produces something like this:

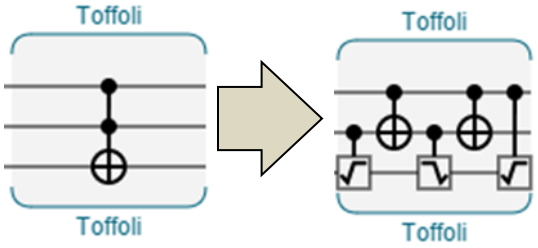


Visually, it's very easy to see that this is an exchange operation which swaps both tracks of the affected qubits. There may be practical problems with this method. For example, the photon tracks become different lengths from one another.

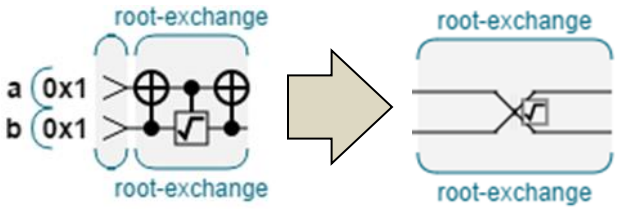
Blind Leap #3: Square Root of Exchange

To do any complex logic, we're going to need a logical Toffoli equivalent. As mentioned above, I'd love to assume there's a silicon-photonic Toffoli which also is perfectly balanced and requires no error-tracks, but for now I'm going to assume that it doesn't exist.

In order to construct a normal Toffoli gate from two-qubit operations, we can do something like this (where the root-boxes signify the quantum root-of-not operation and its inverse):



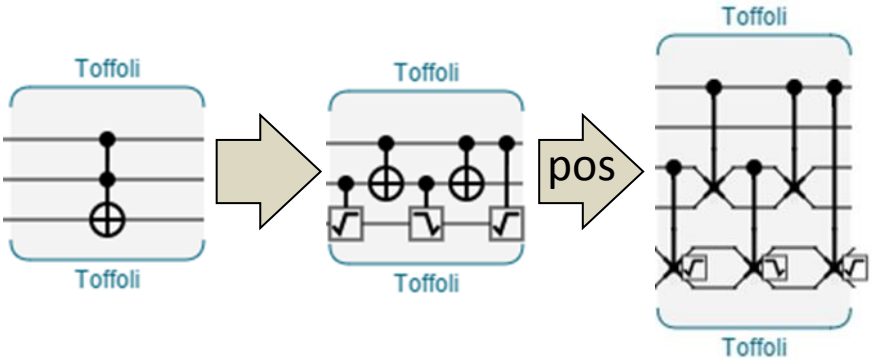
Using those same gates, I'll construct a "root-of-exchange" gate, such that if you apply it twice it performs an exchange operation.



I'm not happy with the symbol, but I can't seem to find a commonly-used symbol for this.

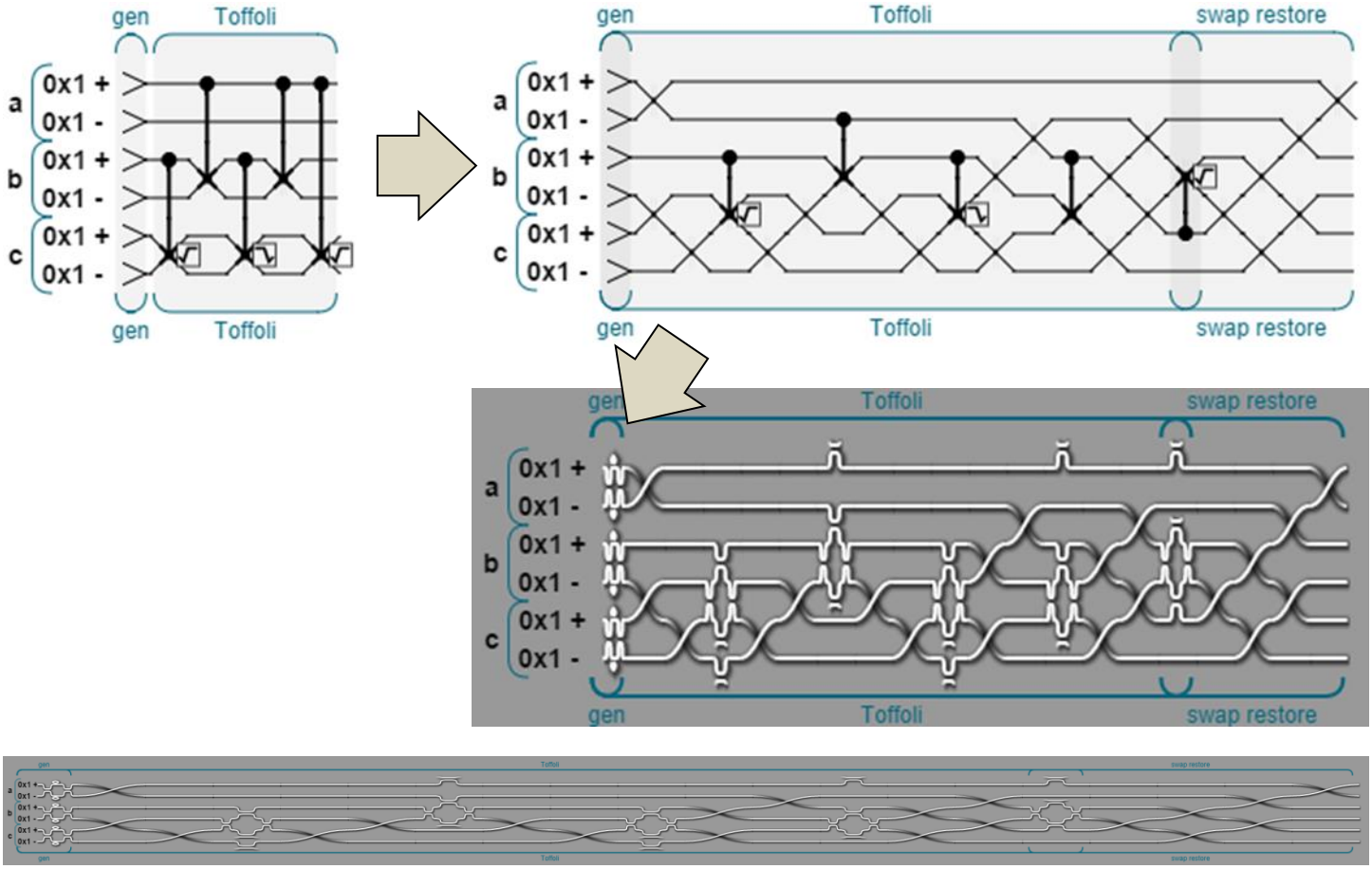
Here's the Blind Leap: I'm going to imagine that simply reducing the length of a section of the photonic silicon-etched CEXCHANGE gate will result in a functional c-root-of-exchange gate.

As long as we have c-root-of-exchange, then when we convert our Toffoli to position-encoding, it will look something like this:



Note that I'm assuming that if we have root-of-exchange, then we also get to have its inverse. If not, then three root-of-exchange operations in a row will do the same thing, but it's reasonable to assume be done in a single gate.

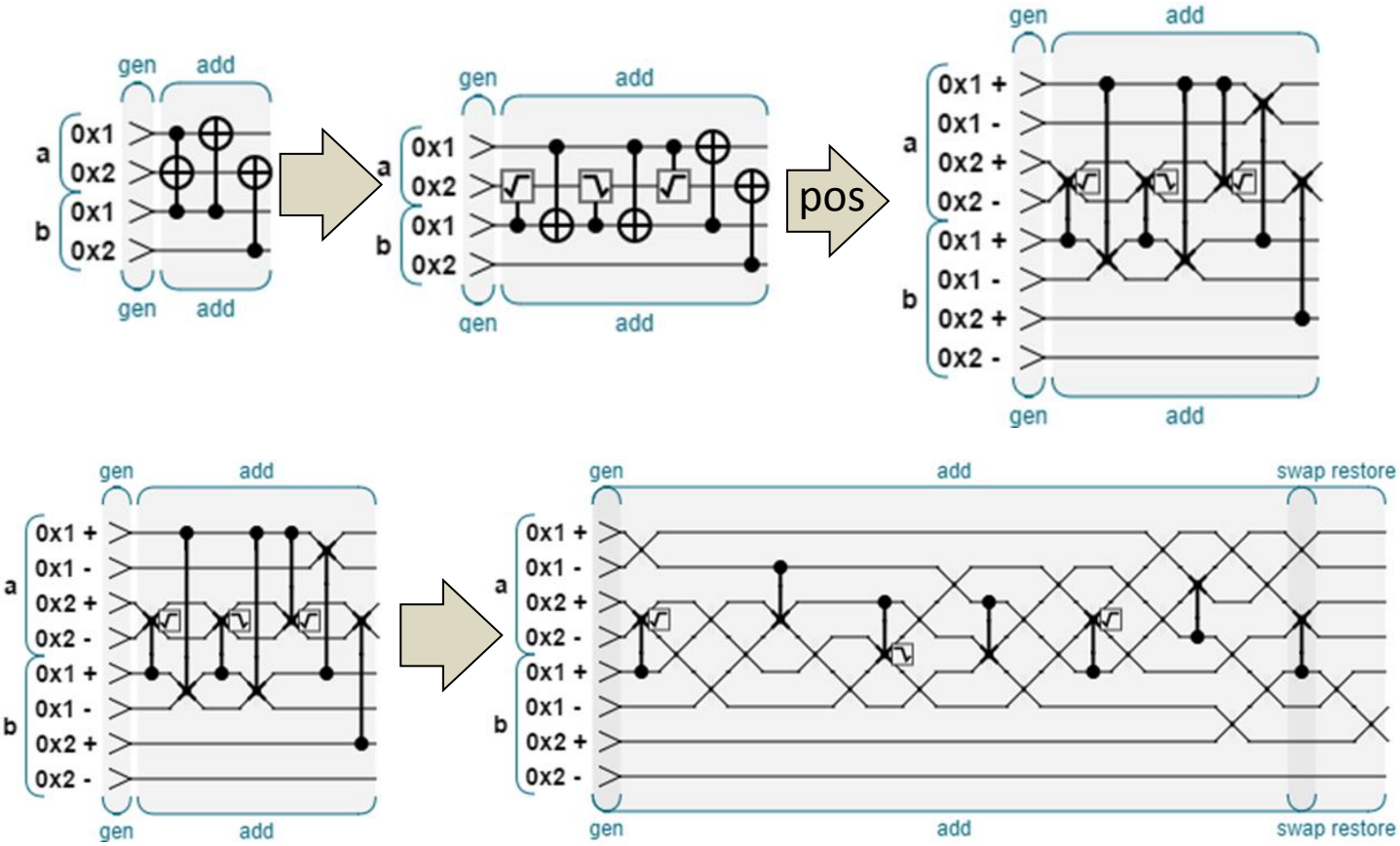
Of course, in order to make that actually silicon-etchable, we need to make those two-qubit operations adjacent, using our twist-exchange:



Aha. Now we've got something we can etch in silicon (so long as out Blind Leaps are valid). We can also see that of our 22 "gates" (including the twist-exchanges), 17 of them are spent just getting our waveguides to the right place, and only 5 are really used for computation. If we consider each horizontal gate-width one "tick" for our device, then this device performs 5 operations in 15 ticks.

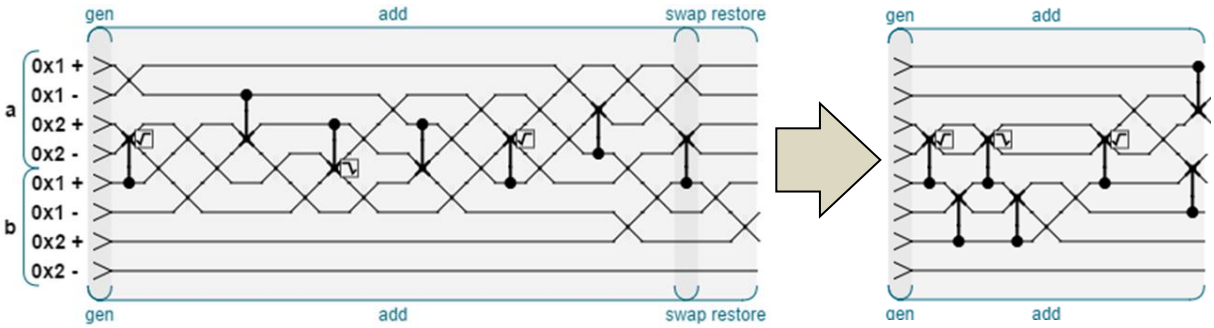
We can actually remove 5 of these (and increase efficiency by 33%!) ticks just by rearranging the start and end positions of each waveguide.

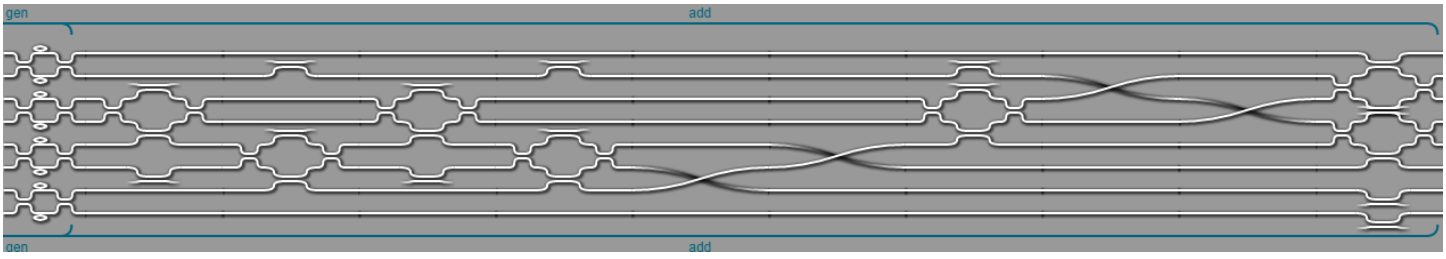
Still, we now have a complete Toffoli gate! With that, we can use this QCEngine example to try out our 2-bit adder from the beginning of this whitepaper (the result takes 22 ticks, and I'll show the "squeezed" version as well, just because it's easier to see):



Optimizing the Result

While this ***could*** work, simplification and optimization are going to be a critical part of constructing early devices. With that in mind, we can remove most of the twist-exchanges by simply agreeing to re-label the inputs and outputs. That's certainly worth it, especially considering this result:





That's much better. It's only four twists short of being best-possible perfection, and the last two gates (which don't depend on each other) are done in parallel, if the 'overlapping trash bits' issue is addressed. This i/o remapping can certainly be done in software, and it's certainly a worthwhile optimization. (QCEngine figures out this optimization on its own, there's a one-button operation to do it.)

So there it is. It can't be built yet (in 2015), but maybe soon.

Future Work: 3D Hex-Packed Mayhem

If we could reduce the amount of swapping, these devices might be much more efficient. Suppose instead of flat silicon-etching we had some way to 3D-print from a voxel grid. in that case, we could put the waveguides in a hex-close-packed arrangement, so that instead of having two nearest neighbors, each waveguide had six. More explorations on that soon, but the foundation can be found in this 2012 whitepaper.

Conclusion

With a few not-too-fanciful leaps, we've got a near-future plan for mapping QC programs of arbitrary complexity onto silicon chips, making them available for productive use. The 1-in-9 success probability for the silicon CEXCHANGE gate is certainly the largest barrier to this. If you find a solution to that one, please send me a link.

Useful References

- *Silicon Quantum Photonic Circuits for On-chip Qubit Generation, Manipulation and Logic Operations*
- Minds, Machines, and the Multiverse, by Julian Brown
- Elementary Gates for Quantum Computation (1995) arXiv:quant-ph/9503016v1
- Quantum Computing for Computer Scientists (Yanofsky and Mannucci), 2008
- IBM QC advances 2/28/2012:
    - http://ibmquantumcomputing.tumblr.com/
    - http://www.nytimes.com/2012/02/28/technology/ibm-inch-closer-on-quantum-computer.html
- [TODO: add other papers and books I've found useful]

About the Author

EJ and his muse live in a secret laboratory in San Francisco. Everything else you really need to know is either posted here, or can be obtained by sending an email to ej@machinelevel.com.