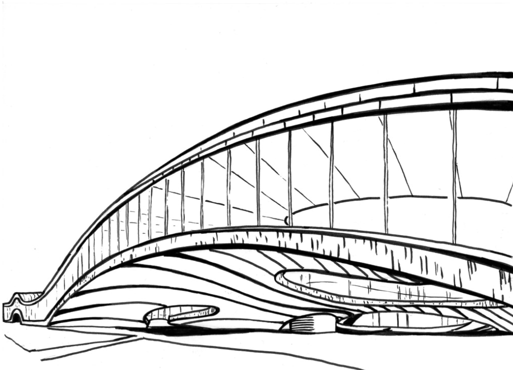
Introduction to RevKit

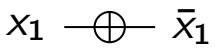
Mathias Soeken

Integrated Systems Laboratory, EPFL, Switzerland

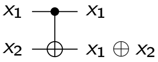
mathias.soeken@epfl.ch msoeken.github.io msoeken/cirkit download slides



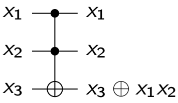
Reversible gates



NOT

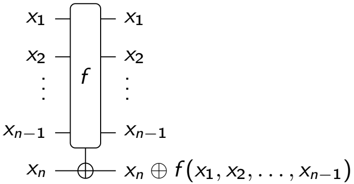


CNOT



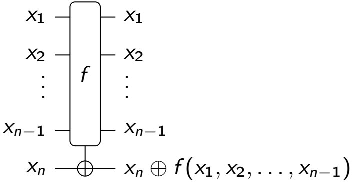
Toffoli

Reversible gates

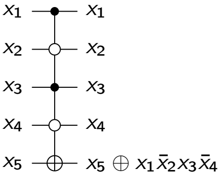


Single-target

Reversible gates

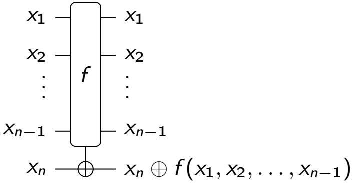


Single-target

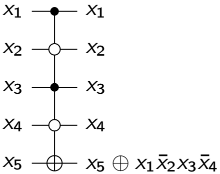


Multiple-controlled Toffoli

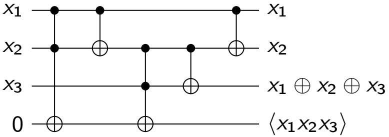
Reversible gates



Single-target

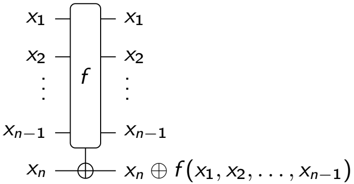


Multiple-controlled Toffoli

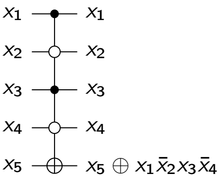


Full adder

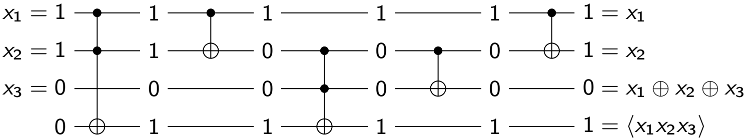
Reversible gates



Single-target



Multiple-controlled Toffoli



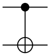
Full adder

Quantum gates

- Qubit is vector $|\varphi\rangle = \left(\begin{smallmatrix} \alpha \\ \beta \end{smallmatrix} \right)$ with $|\alpha^2| + |\beta^2| = 1$.
- Classical 0 is $|0\rangle = \left(\begin{smallmatrix} 1 \\ 0 \end{smallmatrix} \right)$; Classical 1 is $|1\rangle = \left(\begin{smallmatrix} 0 \\ 1 \end{smallmatrix} \right)$

Quantum gates

- Qubit is vector $|\varphi\rangle = \left(\begin{smallmatrix}\alpha\\ \beta\end{smallmatrix}\right)$ with $|\alpha^2| + |\beta^2| = 1$.
- Classical 0 is $|0\rangle = \left(\begin{smallmatrix}1\\ 0\end{smallmatrix}\right)$; Classical 1 is $|1\rangle = \left(\begin{smallmatrix}0\\ 1\end{smallmatrix}\right)$

$|\varphi_1\rangle$
$|\varphi_2\rangle$

$\begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&0&1\\0&0&1&0\end{pmatrix}|\varphi_1\varphi_2\rangle$

CNOT

$|\varphi\rangle\ -\boxed{H}-\ \frac{1}{\sqrt{2}}\begin{pmatrix}1&1\\1&-1\end{pmatrix}|\varphi\rangle$

Hadamard

$|\varphi\rangle\ -\boxed{T}-\ \begin{pmatrix}1&0\\0&e^{\frac{i\pi}{4}}\end{pmatrix}|\varphi\rangle$

T

Composing quantum gates

- Applying a quantum gate to a quantum state (matrix-vector multiplication)

$$|\varphi\rangle = \left(\begin{smallmatrix}\alpha\\ \beta\end{smallmatrix}\right) \;-\!\boxed{U}\!-\; U|\varphi\rangle$$

Composing quantum gates

- Applying a quantum gate to a quantum state (matrix-vector multiplication)

$$|\varphi\rangle = \left(\begin{smallmatrix} \alpha \\ \beta \end{smallmatrix} \right) \; -\boxed{U}- \; U|\varphi\rangle$$

- Applying quantum gates in sequence (matrix product)

$$|\varphi\rangle = \left(\begin{smallmatrix} \alpha \\ \beta \end{smallmatrix} \right) \; -\boxed{U_1}-\boxed{U_2}- \; (U_2 U_1)|\varphi\rangle$$

Composing quantum gates

- Applying a quantum gate to a quantum state (matrix-vector multiplication)

$$|\varphi\rangle = \left(\begin{smallmatrix}\alpha\\ \beta\end{smallmatrix}\right) -\boxed{U}-\ U|\varphi\rangle$$

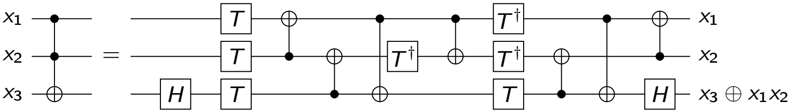
- Applying quantum gates in sequence (matrix product)

$$|\varphi\rangle = \left(\begin{smallmatrix}\alpha\\ \beta\end{smallmatrix}\right) -\boxed{U_1}-\boxed{U_2}-\ (U_2 U_1)|\varphi\rangle$$

- Applying quantum gates in parallel (Kronecker product)

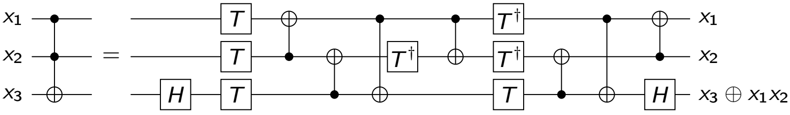
$$\left.\begin{array}{l} |\varphi_1\rangle = \left(\begin{smallmatrix}\alpha_1\\ \beta_1\end{smallmatrix}\right) -\boxed{U_1}- \\ |\varphi_2\rangle = \left(\begin{smallmatrix}\alpha_2\\ \beta_2\end{smallmatrix}\right) -\boxed{U_2}- \end{array}\right\} (U_1 \otimes U_2)|\varphi_1\varphi_2\rangle$$

Mapping Toffoli gates

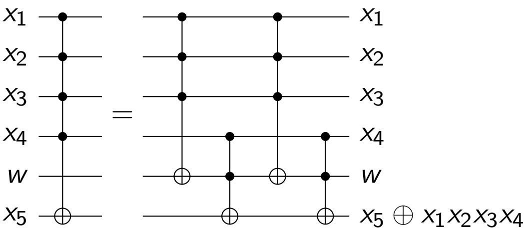


Clifford+T circuit [Amy *et al.*, *TCAD* **32**, 2013]

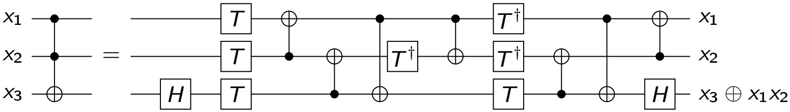
Mapping Toffoli gates



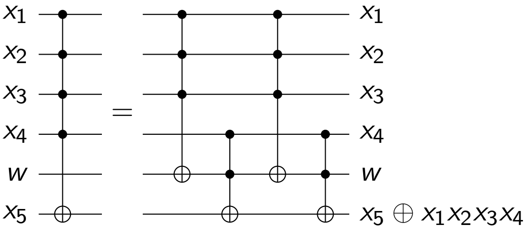
Clifford+T circuit [Amy *et al.*, *TCAD* **32**, 2013]



Mapping Toffoli gates



Clifford+T circuit [Amy *et al.*, *TCAD* **32**, 2013]



Costs are number of qubits and number of T gates

RevKit

- Open source C++ framework for reversible logic synthesis (since 2009)
  - Implemented as add-on in CirKit, an open source C++ framework for logic synthesis
  - Provides command-line interface shell (CLI) and API
  - CLI allows batch processing and can be used via Python API
  - No Python API for C++ API
  - Implement core functionality in C++ and expose it via CLI commands

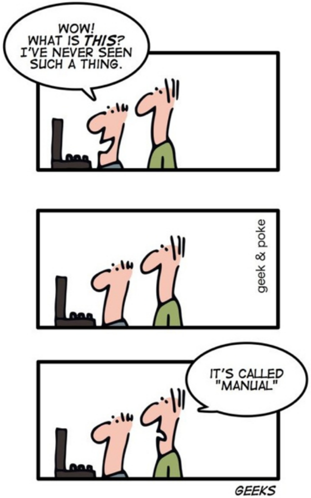
RevKit [Installation]

- Obtain from Github: github.com/msoeken/cirkit
  - Works smoothly on modern Mac OS and Linux distributions
  - Works on Windows OS using Ubuntu subsystem
  - Install dependencies via package manager (in Mac OS, e.g., brew)

RevKit [Help]

- **help** — prints a list with all commands
- **help -d** — prints a detailed list with all commands
- ***command*** **-h** — shows the help of a command
- **help -s** ***text*** — search for some text in all commands' help texts

- revkit -ef addons/cirkit-addon-reversible/demo.cs
- cirkit.readthedocs.io

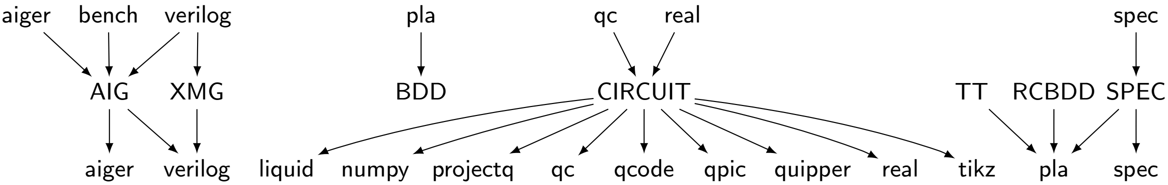


RevKit [Generalities]

- RevKit is commands plus stores
- Stores for each relevant data structure: e.g., reversible circuits, reversible truth tables, AND-inverter graphs, binary decision diagrams, ...
- Stores can contain several instances, commands work on current element

```
revkit> read_real -s "t a b c, f b a c"
revkit> store -c
[i] circuits in store:
  *  0: 3 lines, 2 gates
revkit> tof
revkit> ps -c
Lines:         3
Gates:         4
T-count:       14
Logic qubits:  4
revkit> write_liquid file.fs
```

RevKit [File formats]



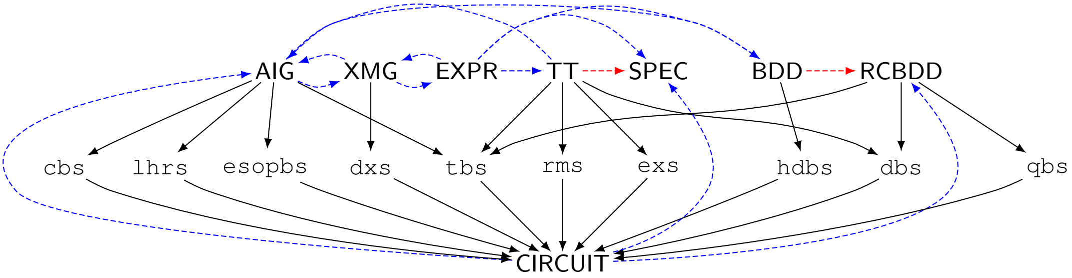
RevKit [General commands]

- **alias** — creates an alias
- **convert** — converts one store element into another
- **current** — changes current store element
- **help** — prints a list with all commands
- **print** — prints current store element as ASCII
- **ps** — prints statistics about current store element
- **quit** — quits RevKit
- **set** — sets global (settings) variable
- **show** — generates visual representation of current store element (as DOT file)
- **store** — interact with the store

RevKit [Synthesis commands]

	line opt.	*gate opt.*	**nonreversible func.**	**reversible func.**
functional	✔	✔		SAT-based exs Enumerative
functional	✔	✘		Transformation-based tbs, rms, qbs Cycle-based cyclebs Decomposition-based dbs Metaheuristic Greedy
structural	✘	✘	ESOP-based esopbs Hierarchical cbs, dxs, hdbs, lhrs Building block	

RevKit [Synthesis and formats]



RevKit [Functional synthesis (i)]

```
revkit> revgen --hwb 6                    # generate reversible HWB-6 function
revkit> tbs                               # transformation-based synthesis
revkit> dbs -n                            # decomposition-based synthesis
revkit> store -c                          # show store for reversible circuits
[i] circuits in store:
     0: 6 lines, 141 gates
  *  1: 6 lines, 89 gates
revkit> rec                               # reversible equivalence checking
[i] circuits are equivalent
revkit> reverse                           # reverse current circuit
revkit> concat -n                         # concat both circuits
revkit> store -c
[i] circuits in store:
     0: 6 lines, 141 gates
     1: 6 lines, 89 gates
  *  2: 6 lines, 230 gates
revkit> is_identity                       # does circuit realize identity?
[i] circuit represents the identity function
```

RevKit [Functional synthesis (ii)]

```
revkit> revgen --prime 4                              # nth_prime_inc rev. func.
revkit> tbs; ps -c                                    # transformation-based synthesis
Gates:    16
T-count: 138
revkit> revsimp; ps -c                                # simplification algorithm
Gates:    15
T-count: 131
revkit> dbs; ps -c                                    # decomposition-based synthesis
Gates:    12
T-count: 120
revkit> dbs --ordering "2 0 1 3"; ps -c               # try different options
Gates:    11
T-count: 99
revkit> exs -ns 10; ps -c                             # can we do it with 10 gates?
Gates:    10
T-count: 99
revkit> exs -n; ps -c                                 # find the optimum!
Gates:    8
T-count: 74
```

RevKit [Mapping into quantum gates]

```
revkit> tt --maj 5                                # create majority-5 function
revkit> tt > aig                                  # create AIG from truth table
revkit> esopbs -ae; ps -c; gates                  # ESOP-based synthesis
Gates:   8   T-count: 152
   controls |   0 |   1 |   2 |   3 |   4 | total
Toffoli     |   0 |   1 |   0 |   2 |   5 |     8
revkit> revsimp --methods ep; ps -c; gates        # simplify
Gates:  11   T-count:  94
   controls |   0 |   1 |   2 |   3 |   4 | total
Toffoli     |   0 |   5 |   2 |   2 |   2 |    11
revkit> pos; gates                                # remove negative controls
   controls |   0 |   1 |   2 |   3 |   4 | total
Toffoli     |  18 |   5 |   2 |   2 |   2 |    29
revkit> rptm -n; ps -c; gates                     # map to Clifford+T using
Gates: 235   T-count:  94                         # relative phase Toffoli gates
   controls |   0 |   1 | total
Toffoli     |   0 |  87 |    87
Pauli roots | 112 |   0 |   112
Hadamard    |  36 |   0 |    36
```

RevKit [Checking quantum circuits]

```
revkit> store -c                              # show store of reversible circuits
[i] circuits in store:
      0: 6 lines, 29 gates
   *  1: 6 lines, 235 gates
revkit> qec -r 0 -q 1                         # perform equivalence checking

revkit> read_real -s "H a, t a b"             # let's get entangled
revkit> circuit_matrix                        # print circuit's matrix
{{ 0.707107+0.i,  0.      +0.i,  0.      +0.i,  0.707107+0.i},
 { 0.707107+0.i,  0.      +0.i,  0.      +0.i, -0.707107+0.i},
 { 0.      +0.i,  0.707107+0.i,  0.707107+0.i,  0.      +0.i},
 { 0.      +0.i, -0.707107+0.i,  0.707107+0.i,  0.      +0.i}}
```

RevKit [Mapping commands]

- **concat** — Concatenate two circuits
- **filter** — Filter some gates
- **mitm** — Translates 2-control Toffoli gates into Clifford+T network
- **nct -t** k — Maps large (positive controlled) Toffoli gates into at most k-control Toffoli gates (default for k is 2)
- **pos** — Changes negative controls into positive ones
- **reverse** — Reverses a circuit
- **rptm** — Mapping into Clifford+T networks (Toffoli gates with up to 5 positive controls)
- **stg4** — Map small single-target gates into best-known Clifford+T networks
- **tof** — Translates Fredkin gates into Toffoli gates

RevKit [Logging]

- Write all commands line by line into a script file
- Call **revkit -f script.cs -l script.log**
- Log file is in JSON format and contains data for each executed command
- Log file can easily be analyzed using almost any main stream programming language

RevKit [Jupyter notebook]

- **Idea:** use RevKit's python interface to perform experiments
- Ideal to present and share experimental results
- Generate static web pages (e.g., for own webpage)
- Provide notebook files that allow to regenerate (and extend) experiments

Introduction to RevKit

Mathias Soeken

Integrated Systems Laboratory, EPFL, Switzerland

mathias.soeken@epfl.ch msoeken.github.io msoeken/cirkit download slides