Chair for Design Automation
TUM School of Computation, Information and Technology
Technical University of Munich

The Munich Nanotech Toolkit (MNT)

Design Automation Tools and Software for Nanotechnology

Robert Wille and Team
Contact: robert.wille@tum.de
https://www.cda.cit.tum.de/research/nanotech/mnt

Abstract

The need for green computational nanotechnologies is growing rapidly. Their design at the atomic scale requires automatic, efficient, and user-friendly software tools that cater to the needs of researchers and developers at every level of the design flow. The *Munich Nanotech Toolkit* (MNT) is a collection of design automation tools and software for nanotechnology developed at the *Chair for Design Automation* at the *Technical University of Munich*. This flyer provides an overview thereof. For each step in the design flow, numbered nodes indicate the correspondingly available software repositories (summarized on the back of this flyer). All software is available as open-source.

Data Structures & Core Methods 4 5 6 7 8

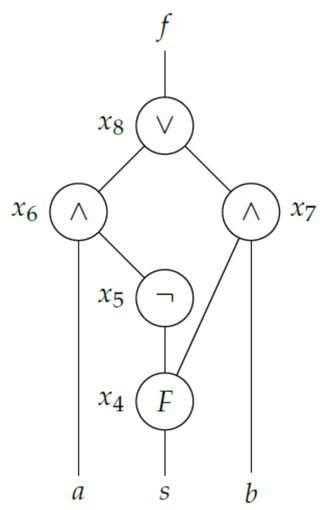
To tackle the complexity of important design tasks, the MNT utilizes data structures to efficiently store and manipulate layouts with millions of elements. Furthermore, dedicated core methods for the realization of optimal solutions are prevalent:

- Logic Networks
- SAT/SMT Solvers
- Machine Learning
- Efficient Heuristics

For performance reasons, all tools are implemented in C++, include convenient Python bindings, and compatibility with prominent tools.

Logic Synthesis 4

- From RTL descriptions to technology-specific logic network representations.
- Unconventional synthesis, novel cost functions, and constraints.
- Crossing and buffer count minimization.
- Delay optimization.



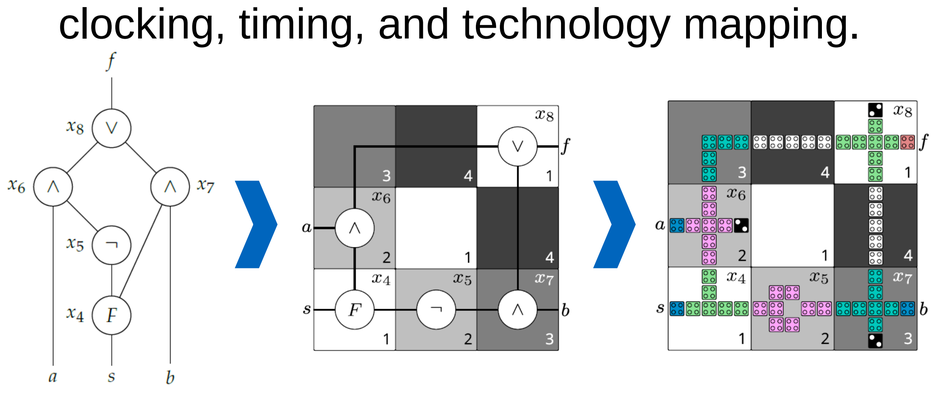
Logic Network

Logic Synthesis

Physical Design

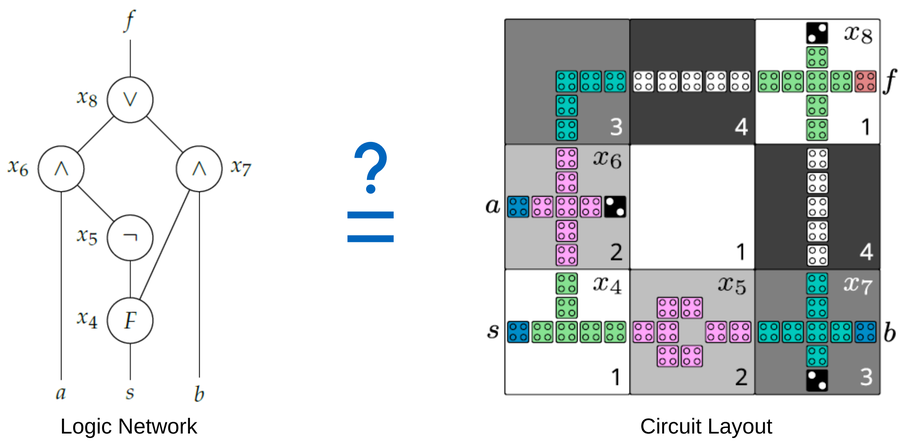
Physical Design 5 7 13 15

- Obtainment of circuit layouts from logic descriptions that involves placement, routing, clocking, timing, and technology mapping.



Verification 6

- Layout validation via design-rule violations.
- Equivalence checking of circuit layouts.
- Verification of physical design results.
- Incorporation of delay and clocking.

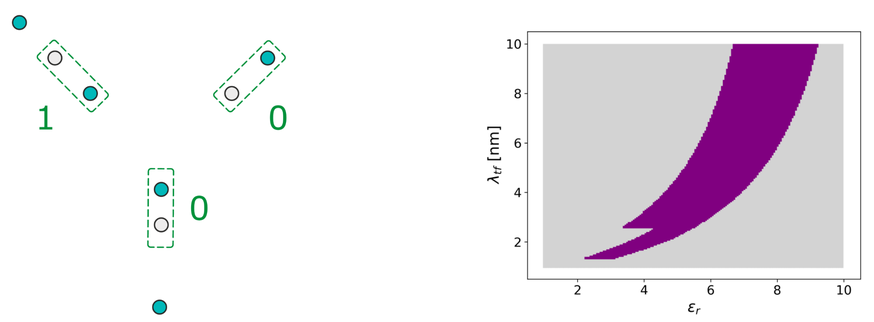


Verification

Simulation

Simulation 8 14

- Exact and heuristic electrostatic ground state simulation for SiDBs.
- Temperature and defect influence regions.
- Fast operational domain computation.



Tool Support

Tool Support 9 14

- File format and plug-in support for a variety of domain-specific tools.

All tools are available as open-source implementations on GitHub.

Chair for Design Automation
TUM School of Computation, Information and Technology
Technical University of Munich

1 MNT *fiction*

Framework

2 Command-line Interface

Application

Quick and easy access to logic synthesis, physical design, verification, technology mapping, and simulation via a CLI that works on every major operating system.

3 Experiment Sandbox

Application

Convenient reproduction of experiments and support for open data in research via a scripting sandbox that also enables quick evaluation of ideas and conceptions.

4 Logic Synthesis

Module

Wire, crossing, and delay cost-aware logic synthesis for area-efficient circuit layouts that includes MAJ synthesis for technology-specific applications.

5 Physical Design

Module

Exact and heuristic placement, routing, clocking, and timing that abstracts from concrete technological implementations while preserving common traits.

6 Verification

Module

Effective layout validation methodologies that include design-rule violation and equivalence checking of logic networks and circuit layouts via formal methods.

7 Technology Mapping

Module

After technology-independent physical design, the resulting layouts are mapped to technology-specific cell implementations using gate libraries.

8 Simulation

Module

Exact and heuristic technology-specific electrostatic ground state simulation, temperature behavior analysis, defect influence, and fast operational domains.

9 Tool Support

Module

File format and plug-in support for established tools in the domain including but not limited to *SiQAD*, *QCADesigner*, *ToPoliNano & MagCAD*, and *SCERPA*.

10 Header-only C++ Library

Code Library

The core library that includes all the efficient data types and state-of-the-art algorithms for the above design tasks in the nanotechnology domain.

11 Python Bindings

Code Library

Python bindings for the C++ library that unite the best of both worlds: the performance of compiled code and the flexibility and tooling of Python.

PyPI

github.com/cda-tum/fiction

12 MNT Bench

Application

Benchmark suite and design library for various nanotechnologies.

www.cda.cit.tum.de/mntbench
github.com/cda-tum/mnt-bench

PyPI

13 MNT NanoPlaceR

Physical Design

Physical design for nanotechnology using reinforcement learning.

github.com/cda-tum/mnt-nanoplacer

PyPI

14 MNT SiQAD Plugins

Simulation

Backends for the *SiQAD* design tool for SiDBs based on our physical simulators.

github.com/cda-tum/mnt-siqad-plugins

15 MNT Bestagon Gate Library

Technology Mapping

Gate library for the SiDB technology using hexagonal standard tiles.

github.com/cda-tum/mnt-bestagon-library

Technische Universität München