Technical University of Munich
Department of Electrical and Computer Engineering
Chair of Electronic Design Automation

acst - An Analog Circuit Synthesis Tool

Documentation

Inga Abel, Helmut Graeb

inga.abel@tum.de

Contents

1. Overview

The analog circuit synthesis tool (**acst**) supports following functionalities:

- Recognition of basic structures in analog circuits (*Structure Recognition*) [1]
- Developing rule sets for new structures to automatically recognize them in circuits (*Rule Generation*) [2]
- Partitioning of op-amps into their functional blocks (*Partitioning*) [3]
- Sizing of basic op-amps (*Automatic Sizing*) [4]
- Synthesis of basic op-amps (*Synthesis*) [5, 6]
- Automatic generation of a topology library currently featuring around 4000 op-amps (*Topology Library Generation*)

1.1. Structure Recognition

The structure recognition methods is able to recognize basic structures as current mirrors and differential pairs in op-amps. It is detailed described in [1, 7].

1.2. Rule Generation

The rule generation functionality allows to automatically develop recognition rules for new structures in analog circuits that repeatedly occur like level shifters. The structures are stored in defined libraries which can be used to analyze circuits for this structures. The method is described in [2].

1.3. Partitioning

The partitioning method automatically recognizes functional blocks in op-amps. These functional blocks are the amplification stages, their subblocks (load, transconductor, stage bias), the op-amp bias circuits, and basic compensation structures. A list of supported example circuits which can be partitioned is given in Appendix C. The method is described in detail in [3].

1.4. Automatic Sizing

The automatic sizing method generates initial device sizes for provided analog op-amps. A list of supported example circuits is given Appendix C. The method is described in detail in [4].

1.5. Synthesis

The synthesis method generates a set of op-amp topologies fulfilling a given set of specifications. Three different op-amp types are supported: fully-differential, complementary and single-output. The method is in detail described in [6, 8].

1.6. Topology Library Generation

The topology library generation method generates all topologies currently supported by the synthesis method within **acst**. These are currently 3912 topologies including single-output op-amps, symmetrical op-amps, and fully-differential op-amps.

2. Compiling and Installing acst

To be able to compile **acst**, following additional libraries must be provided:

- Boost (`https://www.boost.org/`)
- RapidXml (`http://rapidxml.sourceforge.net/`)
- GeCode (`https://www.gecode.org/`)

If not already available on your computer system. Please download the respective libraries from the provided websites and install them (see Appendix D for further information).

If not provided on your system, you also need the gnu/g++ compiler collection able to handle C++-17 and CMake. They can be obtained with following command:

```
$ sudo apt install build-essential
```

To compile acst, please follow the steps below.

1. Download **acst** from git and safe it into a proper directory "{*filepath*}".

2. Open {*filepath*}`/acst/CMakeLists.txt` and adjust the file path for the inclusion of the RapidXml-library and Boost-library:

   ```
   ...

   ###
   # Find RapidXml
   ###
   list(APPEND CMAKE_INCLUDE_PATH
            {file path to rapidxml directory}
   )
   find_package(RapidXml REQUIRED)
   include_directories(SYSTEM ${RAPIDXML_INCLUDE_DIR})


   ###
   # Find Boost library
   ###
   set(BOOST_ROOT {file path to boost directory})

   ...
   ```

   An example of an adequate `CMakeLists.txt` file is given below:

   ```
   ...

   ###
   # Find RapidXml
   ###
   list(APPEND CMAKE_INCLUDE_PATH
           /usr/local/public/include/rapidxml/
   )
   find_package(RapidXml REQUIRED)
   include_directories(SYSTEM ${RAPIDXML_INCLUDE_DIR})


   ###
   # Find Boost library
   ###
   set(BOOST_ROOT /usr/include/boost_1_48_0)

   ...
   ```

3. Open {*filepath*}`/acst/cmake/modules/FindGecode.cmake` and adjust the path for the inclusion of the GeCode-library:

   ```
   if(NOT GECODE_FEATURES)
           set(GECODE_FEATURES "driver" "flatzinc" "gist" "graph" "int" "iter"
   ```

```
                "kernel" "minimodel" "scheduling" "search" "set" "support")
endif()

set(GECODE_INCLUDE  {file path to the gecode directory})

if(GECODE_INCLUDE STREQUAL "GECODE_INCLUDE-NOTFOUND")

...
```

An example of an adequate `FindGecode.cmake` file is given below:

```
if(NOT GECODE_FEATURES)
        set(GECODE_FEATURES "driver" "flatzinc" "gist" "graph" "int" "iter"
                "kernel" "minimodel" "scheduling" "search" "set" "support")
endif()

set(GECODE_INCLUDE  usr/local/public/include/gecode-release-6.2.0)

if(GECODE_INCLUDE STREQUAL "GECODE_INCLUDE-NOTFOUND")

...
```

4. Open $\{filepath\}$`/acst/Control/script/acst.sh` and adjust the file path to the GeCode-library:

```
#!/bin/bash

export LD_LIBRARY_PATH="{file path to the gecode directory}:${LD_LIBRARY_PATH}"
export OA_UNSUPPORTED_PLAT=linux_rhel40
${CMAKE_RUNTIME_OUTPUT_DIRECTORY}/acst $@
```

An example of an adequate `acst.sh` file is given below:

```
#!/bin/bash

export LD_LIBRARY_PATH="usr/local/public/include/gecode-release-6.2.0:${LD_LIBRARY
_PATH}"
export OA_UNSUPPORTED_PLAT=linux_rhel40
${CMAKE_RUNTIME_OUTPUT_DIRECTORY}/acst $@
```

5. Use following Linux commands to compile **acst**:

```
$ cd {file path}/acst    #enters the acst directory
$ mkdir build             #creates a new build folder within acst
$ cd build                #enters the new build folder within acst
$ cmake ..                #runs CMake to build generate a buildsystem
$ cd ..                   # returns to the acst folder
$ make -C build           # builds acst into the build-folder
```

Please note to run **acst**, `acst.sh` must be made executable. This can be obtained with the following command in the `{file path}/acst/build/bin/` directory:

```
$ chmod a+x acst.sh
```

3. Commands

acst can be best used using command-scripts. Examples of command-scripts are given in `acst/InputFileExamples/` ordered according to the different methods. To make a new script executable, use the command `chmod a+x {newScript.sh}`. Detailed explanations of the different commands that can be used to control **acst** are given below.

3.1. General Commands

General commands are not method specific. These are:

```
--log-level-console [DEBUG/TRACE/OFF]
```

`--log-level-console` specifies what is outputted on the console, debug messages [`DEBUG`], nothing [`OFF`], etc.

```
--analysis [structrec/rulegen/partitioning/automaticsizing/synthesis/toplibgen]
```

`--analysis` specifies what method is performed, structure recognition [`structrec`], rule generation [`rulegen`], partitioning [`partitioning`], sizing [`automaticsizing`], synthesis [`synthesis`] or topology library generation [`toplibgen`].

3.2. Method Specific Commands

Additionally to the general commands, also commands specified for the different functionalities of the tool are needed.

3.2.1. Structure Recognition

Following additional commands are defined for structure recognition:

```
--circuit-netlist [relative file path to the circuit netlist to be analysis]
```

The circuit netlist contains the devices of the circuit and their interconnections. It must have the HSpice-file-format. An example of a circuit netlist that can be analyzed is given in App. A.1.

```
--device-types-file [relative file path to the device types file]
```

The device-types-file contains information of the devices supported by **acst**. An example is shown in App. A.4.

```
--hspice-mapping-file [relative file path to the HSpice mapping file]
```

As the circuit netlist is written in HSpice-file-format, the corresponding mapping must be provided in a HSpice-mapping-file. An example is shown in App. A.2.

```
--hspice-supplynet-file [relative file path to the supply net file]
```

The supply-net-file defines the supply nets in the circuit. They must be named as global nets in the circuit-netlist-file. An example is shown in App. A.3.

```
--xml-structrec-library-file [abosolut file path to the library used for recognition]
```

The library-file defines the basic structures recognizable by **acst**. This library is integrated in **acst**. The file path is: `{file path to acst}/acst/StructRec/xml/AnalogLibrary.xml`. An example of the library-file is shown in App. A.7.

```
--output-file [relative file path to the output file(file is created by program)]
```

The output-file contains all basic structures recognized by **acst** in the circuit netlist. An example is given in App. B.1.

In the following, an example of a command script is shown. The example is provided in `acst/InputFileExamples/StructureRecognition/command.sh`.

```
{absolute file path to acst}/acst/build/bin/acst.sh --circuit-netlist input.ckt
--device-types-file deviceTypes.xcat --hspice-mapping-file HSpiceMapping.xcat
--hspice-supplynet-file supplyNets.xcat --analysis structrec --output-file output.xml
--log-level-console DEBUG --xml-structrec-library-file
{absolute file path to acst}/acst/StructRec/xml/AnalogLibrary.xml
```

To run the script, adjust the absolute file path to **acst** in the script.

3.2.2. Rule Generation

Following additional commands are defined for the rule generation method:

```
--circuit-netlist [relative file path to the circuit netlist containing the circuit for
                                which recognition new rules are created]
```

The circuit netlist contains the devices of the circuit and their interconnections. It must have the HSpice-file-format. An example of a circuit netlist for which new rules can be created is given in in App. A.1.

```
--device-types-file [relative file path to the device types file]
```

The device-types-file contains information of the devices supported by **acst**. An example is shown in App. A.4.

```
--hspice-mapping-file [relative file path to the HSpice mapping file]
```

As the circuit netlist is written in HSpice-file-format, the corresponding mapping must be provided in a HSpice-mapping-file. An example is shown in App. A.2.

```
--hspice-supplynet-file [relative file path to the supply net file]
```

The supply-net-file defines the supply nets in the circuit. They must be named as global nets in the circuit netlist file. An example is shown in App. A.3.

```
--xml-structrec-library-file [abosolut file path to the library used for recognition]
```

The library-file defines the basic set of structures that is used for rule generation. It must at least contain the array library (`acst/StructRec/xml/Array/ArrayLibrary.xml`). Also other structures can be added to build the basis structures of the new established rules.

```
--structure-name [name of the structure for which new rules are created]
```

With the structure name, a new library is created containing the old libraries for recognition as well as the new library created for the structure. Also a folder is created containing the rules for the structure as well as its substructures. Examples are provided in App. B.2.

In the following, an example of a command script is shown. The example is provided in `acst/InputFileExamples/RuleGeneration/command.sh`.

```
{absolute file path to acst}/acst/build/bin/acst.sh --log-level-console DEBUG  --analysis
rulegen --circuit-netlist cascodedSymmetricalCMOSOTA.hspice --xml-structrec-library-file
{absolute file path to acst}/acst/InputFileExamples/RuleGeneration/Library.xml
--hspice-mapping-file HSpiceMapping.xcat --hspice-supplynet-file supplyNets.xcat
--device-types-file deviceTypes.xcat --structure-name SymmetricalCascodeOpAmp
```

To run the script, adjust the absolute file path to **acst** in the script.

3.2.3. Partitioning

Following additional commands are defined for the partitioning method:

```
--circuit-netlist [relative file path to the circuit netlist to be analysis]
```

The circuit netlist contains the devices of the circuit and their interconnections. It must have the HSpice-file-format. An example of a circuit netlist that can be analyzed is given in App. A.1.

```
--device-types-file [relative file path to the device types file]
```

The device-types-file contains information of the devices supported by **acst**. An example is shown in App. A.4.

```
--hspice-mapping-file [relative file path to the HSpice mapping file]
```

As the circuit netlist is written in HSpice-file-format, the corresponding mapping must be provided in a HSpice-mapping-file. An example is shown in App. A.2.

```
--hspice-supplynet-file [relative file path to the supply net file]
```

The supply-net-file defines the supply nets in the circuit. They must be named as global nets in the circuit-netlist-file. An example is shown in App. A.3.

```
--xml-structrec-library-file [abosolut file path to the library used for recognition]
```

The library-file defines the basic structures recognizable by **acst**. This library is integrated in **acst**. The file path is: {`file path to acst`}`/acst/StructRec/xml/AnalogLibrary.xml`. An example of the library-file is shown in App. A.7.

```
--output-file [relative file path to the output file(file is created by program)]
```

The output-file contains all functional blocks recognized by **acst** in the circuit netlist. This are, e.g., the amplification stages and their subblocks. An example is given in App. B.3.

In the following, an example of a command script is shown. The example is provided in `acst/InputFileExamples/Partitioning/command.sh`.

```
{absolute file path to acst}/acst/build/bin/acst.sh --log-level-console DEBUG --output-file
cascodedSymmetricalCMOSOTA.xml --circuit-netlist cascodedSymmetricalCMOSOTA.hspice
--device-types-file deviceTypes.xcat --hspice-mapping-file HSpiceMapping.xcat --analysis
partitioning --hspice-supplynet-file supplyNets.xcat --xml-structrec-library-file
{absolute file path to acst}/acst/StructRec/xml/AnalogLibrary.xml
```

To run the script, adjust the absolute file path to **acst** in the script.

3.2.4. Automatic Sizing

Following additional commands are defined for automatic sizing:

```
--circuit-netlist [relative file path to the circuit netlist to be analysis]
```

The circuit netlist contains the devices of the circuit to be sized and their interconnections. It must have the HSpice-file-format. An example of a circuit netlist that can be sized is given in App. A.1.

```
--device-types-file [relative file path to the device types file]
```

The device-types-file contains information of the devices supported by **acst**. An example is shown in App. A.4.

```
--hspice-mapping-file [relative file path to the HSpice mapping file]
```

As the circuit netlist is written in HSpice-file-format, the corresponding mapping must be provided in a HSpice-mapping-file. An example is shown in App. A.2.

```
--hspice-supplynet-file [relative file path to the supply net file]
```

The supply-net-file defines the supply nets in the circuit. They must be named as global nets in the circuit-netlist-file. An example is shown in App. A.3.

```
--xml-structrec-library-file [abosolute file path to the library used for recognition]
```

The library-file defines the basic structures recognizable by **acst**. This library is integrated in **acst**. The file path is: {`file path to acst`}`/acst/StructRec/xml/AnalogLibrary.xml`. An example of the library-file is shown in App. A.7.

```
--xml-technologie-file [relative file path to the technology file]
```

The technology-file contains all parameters of the manufacturing process needed in the sizing algorithm. An example how it must look like and which parameters are needed is given in App. A.5.

```
--xml-circuit-information-file [relative file path to the circuit information file]
```

The circuit-information-file contains all parameters and specifications to the circuit performance. An example how it must look like and which parameters and specifications are needed is given in App. A.6.

```
--transistor-model [SHM/EKV]
```

`--transistor-model` specifies which transistor model is used by the sizing algorithm, Shichmann-Hodge model [`SHM`], EKV-model (currently not supported) [`EKV`].

```
--scaling [0.1mum/1mum]
```

`--scaling` specifies with which discretization the device sizes are calculated, with an accuracy of 0.1 mum [`0.1mum`], or with an accuracy of 1 mum [`1mum`].

```
--runtime [secs]
```

`--runtime` names the time in which the solver has to find solutions for a circuit. Its is provided in secs.

```
--output-file [relative file path to the output file(file is created by program)]
```

The output-file contains all device sizes calculated by the program, the estimated performance values and the voltage and current values for the circuit in the dc-operation point. An example is given in App. B.4.

In the following, an example of a command script is shown. The example is provided in `acst/InputFileExamples/AutomaticSizing/command.sh`.

```
{absolute file path to acst}/acst/build/bin/acst.sh --log-level-console DEBUG --analysis
automaticsizing --circuit-netlist cascodedSymmetricalCMOSOTA.hspice --output-file
cascodedSymmetricalCMOSOTA.xml --hspice-mapping-file HSpiceMapping.xcat --scaling 0.1mum
--transistor-model SHM --device-types-file deviceTypes.xcat --xml-structrec-library-file
{absolute file path to acst}/acst/StructRec/xml/AnalogLibrary.xml --hspice-supplynet-file
supplyNets.xcat --xml-technologie-file TechnologieFile.xml --runtime 5
--xml-circuit-information-file CircuitParameterAndSpecifications.xml
```

To run the script, adjust the absolute file path to **acst** in the script.

3.2.5. Synthesis

Two types of circuit synthesis can be performed. A synthesis featuring a topology library with around 4000 topologies and a synthesis featuring a smaller topology library with 36 topologies. While in the synthesis process with 4000 topologies, the topologies are dynamically generated, the synthesis with only 36 topologies has a fixed HSpice library.

Which library is used is defined by following command:

```
--use-hspice-library [true/false]
```

When `--use-hspice-library` is set to `true` the smaller HSpice-library is used otherwise the large library is used.

Other additional command are:

```
--device-types-file [relative file path to the device types file]
```

The device-types-file contains information of the devices supported by **acst**. An example is shown in App. A.4.

```
--xml-structrec-library-file [abosolute file path to the library used for recognition]
```

The library-file defines the basic structures recognizable by **acst**. This library is integrated in **acst**. The file path is: {`file path to acst`}`/acst/StructRec/xml/AnalogLibrary.xml`. An example of the library-file is shown in App. A.7.

```
--xml-technologie-file [relative file path to the technology file]
```

The technology-file contains all parameters of the manufacturing process needed in the sizing algorithm. An example how it must look like and which parameters are needed is given in App. A.5.

```
--xml-circuit-information-file [relative file path to the circuit information file]
```

The circuit-information-file contains all circuit parameters and specifications to the circuit performance. An example how it must look like and which parameters and specifications are needed is given in App. A.8.

```
--transistor-model [SHM/EKV]
```

`--transistor-model` specifies which transistor model is used by the sizing algorithm, Shichmann-Hodge model [`SHM`], EKV-model (currently not supported) [`EKV`].

```
--scaling [0.1mum/1mum]
```

`--scaling` specifies with which discretization the device sizes are calculated, with an accuracy of 0.1 mum [`0.1mum`], or with an accuracy of 1 mum [`1mum`].

```
--HSPICE-netlist-dir [path where the files of the circuits fulfilling the specification
                      are written to]
```

For each circuit fulfilling the specifications, a file is created containing the circuit netlist ready to be read in by Cadence [9]. Also, the file lists the expected performance values. An example of such a file is provided in App. B.5.

When the hspice circuit library is used additional command are needed:

```
--xml-circuit-library-file [absolute file path to the HSpice circuit library]
```

The circuit-library-file defines the topologies used in the synthesis algorithm. This library is integrated in **acst**. The file path is: {`file path to acst`}`/acst/Synthesis/hspice/Library.xml`.

An example of the library-file is shown in App. A.7.

```
--hspice-mapping-file [relative file path to the HSpice mapping file]
```

As the library is written in HSpice-file-format, the corresponding mapping must be provided in a HSpice-mapping-file. An example is shown in App. A.2.

```
--hspice-supplynet-file [relative file path to the supply net file]
```

The supply-net-file defines the supply nets in the circuit. They must be named as global nets in the circuit-netlist-file. An example is shown in App. A.3.

In the following, an example of a command script is shown using the large circuit library. The example is provided in `acst/InputFileExamples/Synthesis/command.sh`.

```
{absolute file path to acst}/acst/build/bin/acst.sh --log-level-console DEBUG --analysis
synthesis --device-types-file deviceTypes.xcat --xml-structrec-library-file
 {absolute file path to acst}/acst/StructRec/xml/AnalogLibrary.xml   --scaling 1mum
--transistor-model SHM --xml-circuit-information-file CircuitSpecifications.xml
--xml-technologie-file TechnologieFile.xml --HSPICE-netlist-dir
{absolute file path to acst}/acst/InputFileExamples/Synthesis/HspiceNetlist
```

To run the script, adjust the absolute file path to **acst** in the script.

The following command script is an example for a script using the small HSpice circuit library. The example is provided in `acst/InputFileExamples/SynthesisSmallLibrary/command.sh`.

```
{absolute file path to acst}/acst/build/bin/acst.sh --log-level-console DEBUG --analysis
synthesis --use-hspice-library true --xml-technologie-file TechnologieFile.xml
--xml-circuit-information-file CircuitSpecifications.xml --xml-structrec-library-file
{absolute file path to acst}/acst/StructRec/xml/AnalogLibrary.xml  --transistor-model SHM
--xml-circuit-library-file {absolute file path to acst}/acst/Synthesis/hspice/Library.xml
--hspice-mapping-file HSpiceMapping.xcat --hspice-supplynet-file supplyNets.xcat
--device-types-file deviceTypes.xcat  --scaling 1mum --HSPICE-netlist-dir
{absolute file path to acst}/acst/InputFileExamples/SynthesisSmallLibrary/Netlist
```

To run the script, adjust the absolute file path to **acst** in the script.

3.2.6. Topology Library Generation

Following additional commands are defined for the topology library generation method:

```
--HSPICE-netlist-dir [path where the files of the library are written to]
```

For each circuit in the library a HSpice-file is created containing the circuit netlist. An example of such a file is provided in App. B.6.

Adding the structure library to the command with

```
--xml-structrec-library-file [abosolute file path to the library used for recognition]
```

and the device-type-file with

```
--device-types-file [relative file path to the device types file]
```

leads to a generation of circuit files with labeled transistors. To each transistor, its function in the circuit is written (Fig. B.17)

In the following, an example of a command script is shown. The example is provided in `acst/InputFileExamples/TopologyLibraryGeneration/command.sh`.

```
{absolute file path to acst}/acst/build/bin/acst.sh --log-level-console DEBUG
--analysis toplibgen --HSPICE-netlist-dir
{absolute file path to acst}/acst/InputFileExamples/TopologyLibraryGeneration/Netlists
```

To run the script, adjust the absolute file path to **acst** in the script.

To generate circuit files with transistor labels, following command script can be used:

```
{absolute file path to acst}/acst/build/bin/acst.sh --log-level-console DEBUG
--analysis toplibgen --device-types-file deviceTypes.xcat --xml-structrec-library-file
{absolute file path to acst}/acst/StructRec/xml/AnalogLibrary.xml --HSPICE-netlist-dir
{absolute file path to acst}/acst/InputFileExamples/TopologyLibraryGeneration/Netlists
```

To run the script, adjust the absolute file path to **acst** in the script.

4. Tests

For structure recognition, rule generation and partitioning automatic tests are provided. They can be started in `acst/build/` with the command `ctest`. The results are written to `acst/build/Testing/`.

5. Examples

Examples for each method are provided in `acst/InputFileExamples/`.

A. Example Input Files

In the following, examples of input files to **acst** are provided.

A.1. Circuit Netlist

The circuit netlist must always be written in HSpice-format. To obtain an error-free recognition of the functional blocks in op-amps, the supply nets of the circuit must be marked as global nets.

```
** Design library name: CascodeSymmetricalCMOSOTA
** Design cell name: cascodeSymmetricalCMOSOTA
** Design view name: schematic
.GLOBAL vdd! gnd!

.TEMP 25.0
.OPTION
+    ARTIST=2
+    INGOLD=2
+    PARHIER=LOCAL
+    PSF=2

** Library name: CascodeSymmetricalCMOSOTA
** Cell name: cascodeSymmetricalCMOSOTA
** View name: schematic
m17 net30 ibias gnd! gnd! nmos
m8 ibias ibias gnd! gnd! nmos
m7 net20 net20 gnd! gnd! nmos
m6 net26 net20 net48 net48 nmos
m5 net48 net26 gnd! gnd! nmos
m4 net40 inp net36 net36 nmos
m3 net29 inn net36 net36 nmos
m2 net42 net26 gnd! gnd! nmos
m1 out net20 net42 net42 nmos
m0 net36 ibias gnd! gnd! nmos
m16 net30 net30 vdd! vdd! pmos
m15 net20 net30 vdd! vdd! pmos
m14 net26 net30 net27 net27 pmos
m13 net40 net40 vdd! vdd! pmos
m12 net27 net29 vdd! vdd! pmos
m11 net29 net29 vdd! vdd! pmos
m10 net44 net40 vdd! vdd! pmos
m9 out net30 net44 net44 pmos
cl out gnd!
.END
```

Figure A.1.: Example of a circuit netlist file (`acst/InputFileExamples/AutomaticSizing/cascodedSymmetricalCMOSOTA.hspice`)

Rules for the interpretation of the HSpice-file are set in the HSpice-mapping-file (Sec. A.2). The file defines the device naming convention, and the number of pins for specified devices. Also model types are defined. The supply nets defined as global net must be listed in the supply-net-file.

A.2. HSpice Mapping File

The HSpice-mapping-file defines the interpretation of the circuit-netlist-file. The file defines the device naming convention, and the number of pins for specified devices. Also model types are defined.

```
<deviceLineMapper>
        <deviceLineMapping identifier ="q">
                <deviceTypeName>Bipolar</deviceTypeName>
                <deviceIdentifier position="0"/>
                <pins>
                        <pin pinType="Collector" position="1"/>
                        <pin pinType="Base" position="2"/>
                        <pin pinType="Emitter" position="3"/>
                </pins>
                <modelName position="5">
                        <model name="pnp" techType="p"/>
                        <model name="npn" techType="n"/>
                </modelName>
        </deviceLineMapping>
        <deviceLineMapping identifier ="m">
                <deviceTypeName>Mosfet</deviceTypeName>
                <deviceIdentifier position="0"/>
                <pins>
                        <pin pinType="Drain" position="1"/>
                        <pin pinType="Gate" position="2"/>
                        <pin pinType="Source" position="3"/>
                        <pin pinType="Bulk" position="4"/>
                </pins>
                <modelName position="5">
                        <model name="pmos" techType="p"/>
                        <model name ="pmos4" techType="p"/>
                        <model name ="pmos24" techType="p"/>
                        <model name="nmos" techType="n"/>
                        <model name="nmos4" techType="n"/>
                        <model name="nmos24" techType="n"/>
                </modelName>
        </deviceLineMapping>
        <deviceLineMapping identifier ="c">
                <deviceTypeName>Capacitor</deviceTypeName>
                <deviceIdentifier position="0"/>
                <pins>
                        <pin pinType="Plus" position="1"/>
                        <pin pinType="Minus" position="2"/>
                </pins>
                <techType>undefined</techType>
        </deviceLineMapping>
        <deviceLineMapping identifier ="r">
                <deviceTypeName>Resistor</deviceTypeName>
                <deviceIdentifier position="0"/>
                <pins>
                        <pin pinType="Plus" position="1"/>
                        <pin pinType="Minus" position="2"/>
                </pins>
                <techType>undefined</techType>
        </deviceLineMapping>
        <deviceLineMapping identifier ="l">
                <deviceTypeName>Inductor</deviceTypeName>
                <deviceIdentifier position="0"/>
                <pins>
                        <pin pinType="Plus" position="1"/>
                        <pin pinType="Minus" position="2"/>
                </pins>
                <techType>undefined</techType>
        </deviceLineMapping>
        <deviceLineMapping identifier ="d">
                <deviceTypeName>Diode</deviceTypeName>
                <deviceIdentifier position="0"/>
                <pins>
                        <pin pinType="Anode" position="1"/>
                        <pin pinType="Cathode" position="2"/>
                </pins>
                <techType>undefined</techType>
        </deviceLineMapping>
</deviceLineMapping>
```

Figure A.2.: Example of a HSpice-mapping-file (`acst/InputFileExamples/Automatic Sizing/HSpiceMapping.xcat`)

A.3. Supply Net File

The supply-net-file list all supply nets supported as global nets. Two different voltage level can be specified.

```
GND_1 "gnd!"
GND_1 "vss!"
GND_1 "vssb!"
GND_1 "vssp!"
GND_1 "vssb"
GND_1 "vssp"
VDD_1 "vddLow!"
VDD_2 "vdd!"
VDD_2 "vddp!"
VDD_2 "vdd"
VDD_2 "vddp"
VDD_1 "Vdd"
```

Figure A.3.: Example of a supply net file (`acst/InputFileExamples/AutomaticSizing/supplyNets.xcat`)

A.4. Device Types File

The device-types-file defines the devices supported by **acst** and their parameters, as pin names and doping types.

```
<deviceTypes>
        <deviceType name = "Mosfet">
                <techTypes>
                        <techType>n</techType>
                        <techType>p</techType>
                </techTypes>
                <pinTypes>
                        <pinType>Drain</pinType>
                        <pinType>Gate</pinType>
                        <pinType>Source</pinType>
                        <pinType optional = "true" autoConnection="Source">Bulk</pinType
                           >
                </pinTypes>
        </deviceType>
        <deviceType name = "Bipolar">
                <techTypes>
                        <techType>n</techType>
                        <techType>p</techType>
                </techTypes>
                <pinTypes>
                        <pinType>Emitter</pinType>
                        <pinType>Base</pinType>
                        <pinType>Collector</pinType>
                </pinTypes>
        </deviceType>
        <deviceType name = "Resistor">
                <techTypes>
                        <techType>undefined</techType>
                </techTypes>
                <pinTypes>
                        <pinType>Plus</pinType>
                        <pinType>Minus</pinType>
                </pinTypes>
        </deviceType>
        <deviceType name = "Capacitor">
                <techTypes>
                        <techType>undefined</techType>
                </techTypes>
                <pinTypes>
                        <pinType>Plus</pinType>
                        <pinType>Minus</pinType>
                </pinTypes>
        </deviceType>
        <deviceType name = "Inductor">
                <techTypes>
                        <techType>undefined</techType>
                </techTypes>
                <pinTypes>
                        <pinType>Plus</pinType>
                        <pinType>Minus</pinType>
                </pinTypes>
        </deviceType>
        <deviceType name = "Diode">
                <techTypes>
                        <techType>undefined</techType>
                </techTypes>
                <pinTypes>
                        <pinType>Anode</pinType>
                        <pinType>Cathode</pinType>
                </pinTypes>
        </deviceType>
</deviceTypes>
```

Figure A.4.: Example of a device-types-file (`acst/InputFileExamples/AutomaticSizing/deviceTypes.xcat`)

A.5. Technology File

The technology-file contains the parameters of the process used for the manufacturing of the circuit. It is needed for the automatic sizing method and synthesis method.

```
<general>
   <thermalVoltage Vt = "0.026"></thermalVoltage><!--- [V] -->
</general>
<pmos>
   <thresholdVoltage vth="-0.564"></thresholdVoltage><!--- [V] -->
   <mobilityOxideCapacity muCox="0.00003574"></mobilityOxideCapacity><!--- [A/(V^2)] -->
   <earlyVoltage earlyVoltage="2.86"></earlyVoltage><!--- [V/mum] -->
   <overlapCapacity Cgdov = "0.000000000666" ></overlapCapacity><!--- [F/m] -->
   <gateOxideCapacity Cox = "0.006058"></gateOxideCapacity><!--- [F/m2] --->
   <zeroBiasBulkJunctionCapacitance Cj = "0.001894"></zeroBiasBulkJunctionCapacitance><!
    --- [F/m^2] -->
   <zeroBiasSidewallBulkJunctionCapacitance Cjsw = "0.0000000003626"></zeroBiasSidewall
    BulkJunctionCapacitance><!--- [F/m] -->
   <bulkJunctionContactPotential pb = "0.99"></bulkJunctionContactPotential><!--- [V]
    -->
   <lateralDiffusionLength Ldiff= "0.0000009968"></lateralDiffusionLength><!-- [mum] -->
   <slopeFactor n="1.31"></slopeFactor>
   <channelLengthCoefficientStrongInversion lamda="0.029"></channelLengthCoefficientStro
    ngInversion>
   <channelLengthCoefficientWeakInversion lamda="0.074"></channelLengthCoefficientWeakIn
    version>
   <minArea Amin="10"></minArea><!--- [mum^2] -->
   <minLength Lmin="1"></minLength><!--- [mum] --><!--- Is used as integer in the p
    rogram -->
   <minWidth Wmin="1"></minWidth><!--- [mum] --><!--- Is used as integer in the program
    -->
</pmos>
<nmos>
   <thresholdVoltage vth="0.405"></thresholdVoltage><!--- [V] -->
   <mobilityOxideCapacity muCox="0.0001693"></mobilityOxideCapacity><!--- [A/(V^2)] -->
   <earlyVoltage earlyVoltage="4.4"></earlyVoltage><!--- [V/mum] -->
   <overlapCapacity Cgdov = "0.000000000620" ></overlapCapacity><!--- [F/m] -->
   <gateOxideCapacity Cox = "0.006058"></gateOxideCapacity><!--- [F/m2] --->
   <zeroBiasBulkJunctionCapacitance Cj = "0.001812"></zeroBiasBulkJunctionCapacitance><!
   --- [F/m^2] -->
   <zeroBiasSidewallBulkJunctionCapacitance Cjsw = "0.0000000005341"></zeroBiasSidewallB
    ulkJunctionCapacitance><!--- [F/m] -->
   <bulkJunctionContactPotential pb = "0.5"></bulkJunctionContactPotential><!--- [V] -->
   <lateralDiffusionLength Ldiff= "0.00003162"></lateralDiffusionLength><!-- [mum] -->
   <slopeFactor n="1.75"></slopeFactor>
   <channelLengthCoefficientStrongInversion lamda="0.024"></channelLengthCoefficientStro
    ngInversion>
   <channelLengthCoefficientWeakInversion lamda="0.07"></channelLengthCoefficientWeakInv
    ersion>
   <minArea Amin="10"></minArea><!--- [mum] -->
   <minLength Lmin="1"></minLength><!--- [mum] --><!--- Is used as integer in the progra
    m -->
   <minWidth Wmin="1"></minWidth><!--- [mum] --><!--- Is used as integer in the program
    -->
</nmos>
```

Figure A.5.: Example of a technology-file (`acst/InputFileExamples/AutomaticSizing/TechnologyFile.xml`)

A.6. Circuit Parameters and Specifications

The circuit-parameters-and-specifications-file contains all information about the circuit needed for circuit sizing. It is needed for the automatic sizing method.

```
<CircuitParameter>
   <LoadCapacities>
      <LoadCapacity>
         <Value>20</Value><!--- [pF] -->
         <DeviceName>cl</DeviceName>
      </LoadCapacity>
   </LoadCapacities>
   <SupplyVoltagePin>
      <SupplyVoltage Vdd="5"></SupplyVoltage><!--- [V] -->
      <NetName>vdd!</NetName>
   </SupplyVoltagePin>
   <GroundPin>
      <GroundVoltage Gnd="0"></GroundVoltage><!--- [V] -->
      <NetName>gnd!</NetName>
   </GroundPin>
   <CurrentBiasPin>
     <BiasCurrent Ibias="10"></BiasCurrent><!--- [uA] -->
     <NetName>ibias</NetName>
   </CurrentBiasPin>
   <InputPinMinus>
      <InputVoltage Vin="2.5"></InputVoltage><!--- [V] -->
      <NetName>inn</NetName>
   </InputPinMinus>
   <InputPinPlus>
      <InputVoltage Vin= "2.5"></InputVoltage>
      <NetName>inp</NetName>
   </InputPinPlus>
   <OutputPin>
      <NetName>out</NetName>
   </OutputPin>
</CircuitParameter>
<Specifications>
   <minimumGain A="80"></minimumGain>
   <minimumTransientFrequency ft="2.75"></minimumTransientFrequency><!--- [MHz] -->
   <maximumSlewRate SR="3.5"></maximumSlewRate><!--- [V/(uS)] -->
   <minimumCMRR CMRR="70"></minimumCMRR><!--- [dB] -->
   <minimumPosPSRR posPSRR="80"></minimumPosPSRR><!--- [dB] -->
   <minimumNegPSRR negPSRR="80"></minimumNegPSRR><!--- [dB] -->
   <OutputVoltageSwing Voutmax="3" Voutmin="1"></OutputVoltageSwing><!--- [V] -->
   <CommonModeInputVoltage Vcmmin="-0.5" Vcmmax="0.5" ></CommonModeInputVoltage><!--- [V
    ] -->
   <GateOverDriveVoltage Vover="0.13"></GateOverDriveVoltage><!--- [V] --->
   <maximumPowerConsumption P ="10"></maximumPowerConsumption><!--- [mW] -->
   <maximumArea Area = "15000"></maximumArea><!--- [mum^2]--->
   <phaseMargin PM = "60"></phaseMargin><!-- [ns] -->
</Specifications>
```

Figure A.6.: Example of a circuit-parameter-and-specifications-file (`acst/InputFileExamples/AutomaticSizing/CircuitParameterAndSpecifications.xml`)

A.7. Library File

For many methods the integrated structure library within **acst** is used. It has following file path: `acst/StructRec/xml/AnalogLibrary.xml`. It names the file path to other libraries containing the information needed for the recognition of specific structures.

```
<xml version="1.0" encoding="utf-8">
<library>
   <arrayLibraries>
      <arrayLibraryFile>Array/ArrayLibrary.xml</arrayLibraryFile>
   </arrayLibraries>
   <pairLibraries>
      <pairLibraryFile>Analog/AnalogLibrary.xml</pairLibraryFile>
   </pairLibraries>
</library>
```

Figure A.7.: Integrated library-file within **acst** (`acst/StructRec/xml/AnalogLibrary.xml`)

For the generation of new recognition rules for structures, it may be useful to use a specific library file for that structure. An example for such a file is given below:

```
<xml version="1.0" encoding="utf-8">
<library>
   <arrayLibraries>
       <arrayLibraryFile>../../StructRec/xml/Array/ArrayLibrary.xml</arrayLibraryFile>
   </arrayLibraries>
   <pairLibraries>
   </pairLibraries>
</library>
```

Figure A.8.: New library-file for the rule generation of a cascode symmetrical op-amp (`acst/InputFileExamples/RuleGeneration/Library.xml`)

For device array recognition (array libraries) still the basic library within **acst** is used. A relative path points to that library. For pair structures as current mirrors or differential pairs no library is used. However, with a relative path, also the basic building block library within **acst** could be integrated.

A.8. Circuit Specifications

The circuit-specifications-file contains all needed specifications for a circuit synthesis. It is needed for the synthesis method.

```
<Specifications>
   <complementary>no</complementary><!-- [yes/no] -->
   <fullyDifferential>yes</fullyDifferential><!-- [yes/no] -->
   <BiasCurrent Ibias = "10"></BiasCurrent><!--- [muA] -->
   <LoadCapacity Cl = "20"></LoadCapacity><!--- [pF] -->
   <SupplyVoltage Vdd="5"></SupplyVoltage><!--- [V] -->
   <GroundVoltage Gnd="0"></GroundVoltage><!--- [V] -->
   <InputVoltage Vin="2.5"></InputVoltage><!--- [V] -->
   <minimumGain A="80"></minimumGain>
   <minimumTransientFrequency ft="2.5"></minimumTransientFrequency><!--- [MHz] -->
   <maximumSlewRate SR="3.5"></maximumSlewRate><!--- [V/(uS)] -->
   <minimumCMRR CMRR="70"></minimumCMRR><!--- [dB] -->
   <minimumPosPSRR posPSRR="0"></minimumPosPSRR><!--- [dB] -->
   <minimumNegPSRR negPSRR="0"></minimumNegPSRR><!--- [dB] -->
   <OutputVoltageSwing Voutmax="3" Voutmin="2"></OutputVoltageSwing><!--- [V] -->
   <CommonModeInputVoltage Vcmmin="-0.5" Vcmmax="0.5" ></CommonModeInputVoltage><!--- [V
   ] -->
   <OffsetError Vmin="-4" Vmax="4"></OffsetError><!--- [mV] -->
   <GateOverDriveVoltage Vover="0.15"></GateOverDriveVoltage><!--- [V] --->
   <maximumPowerConsumption P ="15"></maximumPowerConsumption><!--- [mW] -->
   <maximumArea Area = "15000"></maximumArea><!--- [mum^2]--->
   <settlingTime ts = "450"></settlingTime><!-- [ns] -->
   <phaseMargin PM = "60"></phaseMargin><!-- [] -->
</Specifications>
```

Figure A.9.: Example of a circuit-specifications-file (`acst/InputFileExamples/Synthesis/CircuitSpecifications.xml`)

If the small HSpice library (Sec. 3.2.5) is used for synthesis, the attributes `<complementary>` and `<fullyDifferential>` are not allowed as only single-output op-amps are supported.

B. Output File Examples

For the different methods within **acst**, different output files are generated. In the following, for each method, examples of output files are presented.

B.1. Structure Recognition

The output file of the structure recognition method is a XML-file containing the recognized structures. They are ordered hierarchically. The top most structures are named at the beginning.

```xml
<xcat_results>
   <date day="17" month="5" year="2021" hour="17" minute="58" second="22"/>
   <structure_recognition_results>
      <structure name="MosfetCascodeCurrentMirror[1]" techType="p" instance="/">
         <pins>
            <pin name="Inner1" net="/n3"/>
            <pin name="Inner2" net="/n4"/>
            <pin name="Input" net="/n7"/>
            <pin name="Output" net="/out"/>
            <pin name="Source" net="/vdd!"/>
         </pins>
         <structure name="MosfetDiodeStack[1]" techType="p" instance="/">
            <pins>
               <pin name="Drain" net="/n7"/>
               <pin name="Inner" net="/n3"/>
               <pin name="Source" net="/vdd!"/>
            </pins>
            <structure name="MosfetDiodeArray[6]" techType="p" instance="/">
               <pins>
                  <pin name="Bulk" net="/vdd!"/>
                  <pin name="Drain" net="/n7"/>
                  <pin name="Source" net="/n3"/>
               </pins>
               <devices>
                  <device name="/mp7" deviceType="Mosfet" techType="p" instance="/"/>
               </devices>
            </structure>
            <structure name="MosfetDiodeArray[4]" techType="p" instance="/">
               <pins>
                  <pin name="Bulk" net="/vdd!"/>
                  <pin name="Drain" net="/n3"/>
                  <pin name="Source" net="/vdd!"/>
               </pins>
               <devices>
                  <device name="/mp3" deviceType="Mosfet" techType="p" instance="/"/>
               </devices>
            </structure>
         </structure>
         <structure name="MosfetCascodePair[5]" techType="p" instance="/">
            <pins>
               <pin name="Drain" net="/out"/>
               <pin name="Gate1" net="/n7"/>
               <pin name="Gate2" net="/n3"/>
               <pin name="Inner" net="/n4"/>
               <pin name="Source" net="/vdd!"/>
            </pins>
            <structure name="MosfetNormalArray[10]" techType="p" instance="/">
               <pins>
                  <pin name="Bulk" net="/vdd!"/>
                  <pin name="Drain" net="/out"/>
                  <pin name="Gate" net="/n7"/>
                  <pin name="Source" net="/n4"/>
               </pins>
               <devices>
                  <device name="/mp8" deviceType="Mosfet" techType="p" instance="/"/>
               </devices>
            </structure>
            <structure name="MosfetNormalArray[8]" techType="p" instance="/">
               <pins>
                  <pin name="Bulk" net="/vdd!"/>
                  <pin name="Drain" net="/n4"/>
                  <pin name="Gate" net="/n3"/>
                  <pin name="Source" net="/vdd!"/>
               </pins>
               <devices>
                  <device name="/mp4" deviceType="Mosfet" techType="p" instance="/"/>
               </devices>
            </structure>
         </structure>
      </structure>
```

Figure B.1.: Example of an output file of the structure recognition method [Part 1] (`acst/InputFileExamples/StructureRecognition/output.xml`)

```
<structure name="MosfetCascodeCurrentMirror[2]" techType="n" instance="/">
   <pins>
      <pin name="Inner1" net="/n5"/>
      <pin name="Inner2" net="/n6"/>
      <pin name="Input" net="/n7"/>
      <pin name="Output" net="/out"/>
      <pin name="Source" net="/gnd!"/>
   </pins>
   <structure name="MosfetDiodeStack[2]" techType="n" instance="/">
      <pins>
         <pin name="Drain" net="/n7"/>
         <pin name="Inner" net="/n5"/>
         <pin name="Source" net="/gnd!"/>
      </pins>
      <structure name="MosfetDiodeArray[1]" techType="n" instance="/">
         <pins>
            <pin name="Bulk" net="/gnd!"/>
            <pin name="Drain" net="/n7"/>
            <pin name="Source" net="/n5"/>
         </pins>
         <devices>
            <device name="/mn7" deviceType="Mosfet" techType="n" instance="/"/>
         </devices>
      </structure>
      <structure name="MosfetDiodeArray[3]" techType="n" instance="/">
         <pins>
            <pin name="Bulk" net="/gnd!"/>
            <pin name="Drain" net="/n5"/>
            <pin name="Source" net="/gnd!"/>
         </pins>
         <devices>
            <device name="/mn3" deviceType="Mosfet" techType="n" instance="/"/>
         </devices>
      </structure>
   </structure>
   <structure name="MosfetCascodePair[6]" techType="n" instance="/">
     <pins>
        <pin name="Drain" net="/out"/>
        <pin name="Gate1" net="/n7"/>
        <pin name="Gate2" net="/n5"/>
        <pin name="Inner" net="/n6"/>
        <pin name="Source" net="/gnd!"/>
     </pins>
     <structure name="MosfetNormalArray[5]" techType="n" instance="/">
        <pins>
           <pin name="Bulk" net="/gnd!"/>
           <pin name="Drain" net="/out"/>
           <pin name="Gate" net="/n7"/>
           <pin name="Source" net="/n6"/>
        </pins>
        <devices>
           <device name="/mn8" deviceType="Mosfet" techType="n" instance="/"/>
        </devices>
     </structure>
     <structure name="MosfetNormalArray[3]" techType="n" instance="/">
       <pins>
          <pin name="Bulk" net="/gnd!"/>
          <pin name="Drain" net="/n6"/>
          <pin name="Gate" net="/n5"/>
          <pin name="Source" net="/gnd!"/>
       </pins>
       <devices>
           <device name="/mn4" deviceType="Mosfet" techType="n" instance="/"/>
       </devices>           </structure>
   </structure>
</structure>
```

Figure B.2.: Example of an output file of the structure recognition method [Part 2] (`acst/InputFileExamples/StructureRecognition/output.xml`)

```
      <structure name="MosfetDifferentialPair[1]" techType="n" instance="/">
         <pins>
            <pin name="Input1" net="/ip"/>
            <pin name="Input2" net="/in"/>
            <pin name="Output1" net="/n3"/>
            <pin name="Output2" net="/n4"/>
            <pin name="Source" net="/n1"/>
         </pins>
         <structure name="MosfetNormalArray[2]" techType="n" instance="/">
            <pins>
               <pin name="Bulk" net="/gnd!"/>
               <pin name="Drain" net="/n3"/>
               <pin name="Gate" net="/ip"/>
               <pin name="Source" net="/n1"/>
            </pins>
            <devices>
               <device name="/mn5" deviceType="Mosfet" techType="n" instance="/"/>
            </devices>
         </structure>
         <structure name="MosfetNormalArray[4]" techType="n" instance="/">
            <pins>
               <pin name="Bulk" net="/gnd!"/>
               <pin name="Drain" net="/n4"/>
               <pin name="Gate" net="/in"/>
               <pin name="Source" net="/n1"/>
            </pins>
            <devices>
               <device name="/mn6" deviceType="Mosfet" techType="n" instance="/"/>
            </devices>
         </structure>
      </structure>
      <structure name="MosfetDifferentialPair[2]" techType="p" instance="/">
         <pins>
            <pin name="Input1" net="/in"/>
            <pin name="Input2" net="/ip"/>
            <pin name="Output1" net="/n6"/>
            <pin name="Output2" net="/n5"/>
            <pin name="Source" net="/n2"/>
         </pins>
         <structure name="MosfetNormalArray[7]" techType="p" instance="/">
            <pins>
               <pin name="Bulk" net="/vdd!"/>
               <pin name="Drain" net="/n6"/>
               <pin name="Gate" net="/in"/>
               <pin name="Source" net="/n2"/>
            </pins>
            <devices>
               <device name="/mp6" deviceType="Mosfet" techType="p" instance="/"/>
            </devices>
         </structure>
         <structure name="MosfetNormalArray[9]" techType="p" instance="/">
            <pins>
               <pin name="Bulk" net="/vdd!"/>
               <pin name="Drain" net="/n5"/>
               <pin name="Gate" net="/ip"/>
               <pin name="Source" net="/n2"/>
            </pins>
            <devices>
               <device name="/mp5" deviceType="Mosfet" techType="p" instance="/"/>
            </devices>
         </structure>
      </structure>
```

Figure B.3.: Example of an output file of the structure recognition method [Part 3] (`acst/InputFileExamples/StructureRecognition/output.xml`)

```
      <structure name="MosfetSimpleCurrentMirror[3]" techType="n" instance="/">
         <pins>
            <pin name="Input" net="/n8"/>
            <pin name="Output" net="/n1"/>
            <pin name="Source" net="/gnd!"/>
         </pins>
         <structure name="MosfetDiodeArray[2]" techType="n" instance="/">
            <pins>
               <pin name="Bulk" net="/gnd!"/>
               <pin name="Drain" net="/n8"/>
               <pin name="Source" net="/gnd!"/>
            </pins>
            <devices>
               <device name="/mn1" deviceType="Mosfet" techType="n" instance="/"/>
            </devices>
         </structure>
         <structure name="MosfetNormalArray[1]" techType="n" instance="/">
            <pins>
               <pin name="Bulk" net="/gnd!"/>
               <pin name="Drain" net="/n1"/>
               <pin name="Gate" net="/n8"/>
               <pin name="Source" net="/gnd!"/>
            </pins>
            <devices>
               <device name="/mn2" deviceType="Mosfet" techType="n" instance="/"/>
            </devices>
         </structure>
      </structure>
      <structure name="MosfetSimpleCurrentMirror[4]" techType="p" instance="/">
         <pins>
            <pin name="Input" net="/n9"/>
            <pin name="Output" net="/n2"/>
            <pin name="Source" net="/vdd!"/>
         </pins>
         <structure name="MosfetDiodeArray[5]" techType="p" instance="/">
            <pins>
               <pin name="Bulk" net="/vdd!"/>
               <pin name="Drain" net="/n9"/>
               <pin name="Source" net="/vdd!"/>
            </pins>
            <devices>
               <device name="/mp1" deviceType="Mosfet" techType="p" instance="/"/>
            </devices>
         </structure>
         <structure name="MosfetNormalArray[6]" techType="p" instance="/">
            <pins>
               <pin name="Bulk" net="/vdd!"/>
               <pin name="Drain" net="/n2"/>
               <pin name="Gate" net="/n9"/>
               <pin name="Source" net="/vdd!"/>
            </pins>
            <devices>
               <device name="/mp2" deviceType="Mosfet" techType="p" instance="/"/>
            </devices>
         </structure>
      </structure>
   </structure_recognition_results>
</xcat_results>
```

Figure B.4.: Example of an output file of the structure recognition method [Part 4] (`acst/InputFileExamples/StructureRecognition/output.xml`)

B.2. Rule Generation

The rule generation method generates a new structure recognition library containing the rules for the recognition of basic structures as well as the rules used to recognize the new structure `<structure-name>`. The library consists of a basic file containing the file paths to more detailed descriptions and a folder, named after the new structure (`<structure-name>`), containing the files needed to recognize the new structures.

```
<library>
   <arrayLibraries>
      <arrayLibraryFile>../../StructRec/xml/Array/ArrayLibrary.xml</arrayLibraryFile>
   </arrayLibraries>
   <pairLibraries>
      <pairLibraryFile>../../StructRec/xml/Analog/AnalogLibrary.xml</pairLibraryFile>
      <pairLibraryFile>SymmetricalCascodeOpAmp/SymmetricalCascodeOpAmpLibrary.xml</pairL
       ibraryFile>
   </pairLibraries>
</library>
```

Figure B.5.: Example of a library-file generated by rule generation (`acst/InputFileExamples/RuleGeneration/SymmetricalCascodeOpAmpLibrary.xml`)

In the folder `<structure-name>`, another file is written containing more information about the structure.

```
<xml version="1.0" encoding="utf-8">
<pairLibrary>
   <pairLibraryItemFiles>
      <pairLibraryItemFile>Items/SymmetricalCascodeOpAmp2.xml</pairLibraryItemFile>
      <pairLibraryItemFile>Items/SymmetricalCascodeOpAmp3.xml</pairLibraryItemFile>
      <pairLibraryItemFile>Items/SymmetricalCascodeOpAmp4.xml</pairLibraryItemFile>
      <pairLibraryItemFile>Items/SymmetricalCascodeOpAmp5.xml</pairLibraryItemFile>
      <pairLibraryItemFile>Items/SymmetricalCascodeOpAmp1.xml</pairLibraryItemFile>
      <pairLibraryItemFile>Items/SymmetricalCascodeOpAmp7.xml</pairLibraryItemFile>
      <pairLibraryItemFile>Items/SymmetricalCascodeOpAmp8.xml</pairLibraryItemFile>
      <pairLibraryItemFile>Items/SymmetricalCascodeOpAmp6.xml</pairLibraryItemFile>
      <pairLibraryItemFile>Items/SymmetricalCascodeOpAmp9.xml</pairLibraryItemFile>
      <pairLibraryItemFile>Items/SymmetricalCascodeOpAmp10.xml</pairLibraryItemFile>
   </pairLibraryItemFiles>
   <hierarchyLevels>
      <hierarchyLevel level="2">
         <pairLibraryItem persistence="1">SymmetricalCascodeOpAmp2</pairLibraryItem>
         <pairLibraryItem persistence="1">SymmetricalCascodeOpAmp3</pairLibraryItem>
         <pairLibraryItem persistence="2">SymmetricalCascodeOpAmp4</pairLibraryItem>
         <pairLibraryItem persistence="1">SymmetricalCascodeOpAmp5</pairLibraryItem>
      </hierarchyLevel>
      <hierarchyLevel level="3">
         <pairLibraryItem persistence="1">SymmetricalCascodeOpAmp1</pairLibraryItem>
         <pairLibraryItem persistence="2">SymmetricalCascodeOpAmp7</pairLibraryItem>
         <pairLibraryItem persistence="3">SymmetricalCascodeOpAmp8</pairLibraryItem>
      </hierarchyLevel>
      <hierarchyLevel level="4">
         <pairLibraryItem persistence="1">SymmetricalCascodeOpAmp6</pairLibraryItem>
      </hierarchyLevel>
      <hierarchyLevel level="5">
         <pairLibraryItem persistence="1">SymmetricalCascodeOpAmp9</pairLibraryItem>
      </hierarchyLevel>
      <hierarchyLevel level="6">
         <pairLibraryItem>SymmetricalCascodeOpAmp10</pairLibraryItem>
      </hierarchyLevel>
   </hierarchyLevels>
   <dominanceRelations/>
</pairLibrary>
```

Figure B.6.: File generated by rule generation containing information of the different substructures of the structure for which new recognition rules are generated (`acst/InputFileExamples/RuleGeneration/SymmetricalCascodeOpAmp/SymmetricalCascodeOpAmpLibrary.xml`)

The items-folder contains files with the rules to recognize the structure in a bigger circuit. `SymmetricalCascodeOpAmp10` is the top level structure being the circuit itself. The substructures are only valid if this structure is recognized.

B.3. Partitioning

The output file of partitioning is a xml-file containing the recognized functional blocks. They are ordered hierarchically. The top most structures are named at the beginning.

```
<xcat_results>
   <date day="16" month="3" year="2021" hour="16" minute="36" second="46"/>
   <PartioningResults>
      <TransconductaneParts>
         <gmPart type="firstStage">
            <structure name="MosfetDifferentialPair[1]" techType="n" instance="/">
               <pins>
                  <pin name="Input1" net="/inn"/>
                  <pin name="Input2" net="/inp"/>
                  <pin name="Output1" net="/net29"/>
                  <pin name="Output2" net="/net40"/>
                  <pin name="Source" net="/net36"/>
               </pins>
               <structure name="MosfetNormalArray[2]" techType="n" instance="/">
                  <pins>
                     <pin name="Bulk" net="/net36"/>
                     <pin name="Drain" net="/net29"/>
                     <pin name="Gate" net="/inn"/>
                     <pin name="Source" net="/net36"/>
                  </pins>
                  <devices>
                     <device name="/m3" deviceType="Mosfet" techType="n" instance="/"/>
                  </devices>
               </structure>
               <structure name="MosfetNormalArray[5]" techType="n" instance="/">
                  <pins>
                     <pin name="Bulk" net="/net36"/>
                     <pin name="Drain" net="/net40"/>
                     <pin name="Gate" net="/inp"/>
                     <pin name="Source" net="/net36"/>
                  </pins>
                  <devices>
                     <device name="/m4" deviceType="Mosfet" techType="n" instance="/"/>
                  </devices>
               </structure>
            </structure>
         </gmPart>
         <gmPart type="primarySecondStage">
            <structure name="MosfetCascodePair[5]" techType="p" instance="/">
               <pins>
                  <pin name="Drain" net="/out"/>
                  <pin name="Gate1" net="/net30"/>
                  <pin name="Gate2" net="/net40"/>
                  <pin name="Inner" net="/net44"/>
                  <pin name="Source" net="/vdd!"/>
               </pins>
               <structure name="MosfetNormalArray[13]" techType="p" instance="/">
                  <pins>
                     <pin name="Bulk" net="/net44"/>
                     <pin name="Drain" net="/out"/>
                     <pin name="Gate" net="/net30"/>
                     <pin name="Source" net="/net44"/>
                  </pins>
                  <devices>
                     <device name="/m9" deviceType="Mosfet" techType="p" instance="/"/>
                 </devices>
               </structure>
               <structure name="MosfetNormalArray[12]" techType="p" instance="/">
                  <pins>
                     <pin name="Bulk" net="/vdd!"/>
                     <pin name="Drain" net="/net44"/>
                     <pin name="Gate" net="/net40"/>
                     <pin name="Source" net="/vdd!"/>
                  </pins>
                  <devices>
                     <device name="/m10" deviceType="Mosfet" techType="p" instance="/"/>
                  </devices>
               </structure>
            </structure>
         </gmPart>
```

Figure B.7.: Example of an output file of the partitioning method [Part 1] (`acst/InputFileExamples/Partitioning/cascodedSymmetricalCMOSOTA.xml`)

```xml
      <gmPart type="secondarySecondStage">
         <structure name="MosfetNormalArray[11]" techType="p" instance="/">
            <pins>
               <pin name="Bulk" net="/vdd!"/>
               <pin name="Drain" net="/net27"/>
               <pin name="Gate" net="/net29"/>
               <pin name="Source" net="/vdd!"/>
            </pins>
            <devices>
               <device name="/m12" deviceType="Mosfet" techType="p" instance="/"/>
            </devices>
         </structure>
         <structure name="MosfetNormalArray[10]" techType="p" instance="/">
            <pins>
               <pin name="Bulk" net="/net27"/>
               <pin name="Drain" net="/net26"/>
               <pin name="Gate" net="/net30"/>
               <pin name="Source" net="/net27"/>
            </pins>
            <devices>
               <device name="/m14" deviceType="Mosfet" techType="p" instance="/"/>
            </devices>
         </structure>
      </gmPart>
   </TransconductaneParts>
   <BiasParts>
      <biasPart>
         <structure name="MosfetNormalArray[4]" techType="n" instance="/">
            <pins>
               <pin name="Bulk" net="/gnd!"/>
               <pin name="Drain" net="/net36"/>
               <pin name="Gate" net="/ibias"/>
               <pin name="Source" net="/gnd!"/>
            </pins>
            <devices>
               <device name="/m0" deviceType="Mosfet" techType="n" instance="/"/>
            </devices>
         </structure>
      </biasPart>
      <biasPart>
         <structure name="MosfetCascodePair[4]" techType="n" instance="/">
            <pins>
               <pin name="Drain" net="/out"/>
               <pin name="Gate1" net="/net20"/>
               <pin name="Gate2" net="/net26"/>
               <pin name="Inner" net="/net42"/>
               <pin name="Source" net="/gnd!"/>
            </pins>
            <structure name="MosfetNormalArray[8]" techType="n" instance="/">
               <pins>
                  <pin name="Bulk" net="/net42"/>
                  <pin name="Drain" net="/out"/>
                  <pin name="Gate" net="/net20"/>
                  <pin name="Source" net="/net42"/>
               </pins>
               <devices>
                  <device name="/m1" deviceType="Mosfet" techType="n" instance="/"/>
               </devices>
            </structure>
            <structure name="MosfetNormalArray[6]" techType="n" instance="/">
               <pins>
                  <pin name="Bulk" net="/gnd!"/>
                  <pin name="Drain" net="/net42"/>
                  <pin name="Gate" net="/net26"/>
                  <pin name="Source" net="/gnd!"/>
               </pins>
               <devices>
                  <device name="/m2" deviceType="Mosfet" techType="n" instance="/"/>
               </devices>
            </structure>
         </structure>
      </biasPart>
```

Figure B.8.: Example of an output file of the partitioning method [Part 2] (`acst/InputFileExamples/Partitioning/cascodedSymmetricalCMOSOTA.xml`)

```
         <biasPart>
            <structure name="MosfetVoltageReference2[1]" techType="n" instance="/">
               <pins>
                  <pin name="Drain" net="/net26"/>
                  <pin name="Gate" net="/net20"/>
                  <pin name="Inner" net="/net48"/>
                  <pin name="Source" net="/gnd!"/>
               </pins>
               <structure name="MosfetNormalArray[1]" techType="n" instance="/">
                          <pins>
                     <pin name="Bulk" net="/net48"/>
                     <pin name="Drain" net="/net26"/>
                     <pin name="Gate" net="/net20"/>
                     <pin name="Source" net="/net48"/>
                  </pins>
                  <devices>
                     <device name="/m6" deviceType="Mosfet" techType="n" instance="/"/>
                          </devices>
               </structure>
               <structure name="MosfetNormalArray[7]" techType="n" instance="/">
                  <pins>
                     <pin name="Bulk" net="/gnd!"/>
                     <pin name="Drain" net="/net48"/>
                     <pin name="Gate" net="/net26"/>
                     <pin name="Source" net="/gnd!"/>
                  </pins>
                  <devices>
                     <device name="/m5" deviceType="Mosfet" techType="n" instance="/"/>
                  </devices>
               </structure>
            </structure>
         </biasPart>
         <biasPart>
            <structure name="MosfetDiodeArray[1]" techType="n" instance="/">
               <pins>
                  <pin name="Bulk" net="/gnd!"/>
                  <pin name="Drain" net="/ibias"/>
                  <pin name="Source" net="/gnd!"/>
               </pins>
               <devices>
                  <device name="/m8" deviceType="Mosfet" techType="n" instance="/"/>
               </devices>
            </structure>
         </biasPart>
         <biasPart>
            <structure name="MosfetNormalArray[3]" techType="n" instance="/">
               <pins>
                  <pin name="Bulk" net="/gnd!"/>
                  <pin name="Drain" net="/net30"/>
                  <pin name="Gate" net="/ibias"/>
                  <pin name="Source" net="/gnd!"/>
               </pins>
               <devices>
                  <device name="/m17" deviceType="Mosfet" techType="n" instance="/"/>
               </devices>
            </structure>
          </biasPart>
```

Figure B.9.: Example of an output file of the partitioning method [Part 3] (`acst/InputFileExamples/Partitioning/cascodedSymmetricalCMOSOTA.xml`)

```
      <biasPart>
         <structure name="MosfetDiodeArray[4]" techType="p" instance="/">
            <pins>
               <pin name="Bulk" net="/vdd!"/>
               <pin name="Drain" net="/net30"/>
               <pin name="Source" net="/vdd!"/>
            </pins>
            <devices>
               <device name="/m16" deviceType="Mosfet" techType="p" instance="/"/>
            </devices>
         </structure>
      </biasPart>
     <biasPart>
         <structure name="MosfetNormalArray[9]" techType="p" instance="/">
            <pins>
               <pin name="Bulk" net="/vdd!"/>
               <pin name="Drain" net="/net20"/>
               <pin name="Gate" net="/net30"/>
               <pin name="Source" net="/vdd!"/>
            </pins>
            <devices>
               <device name="/m15" deviceType="Mosfet" techType="p" instance="/"/>
            </devices>
         </structure>
      </biasPart>
      <biasPart>
         <structure name="MosfetDiodeArray[2]" techType="n" instance="/">
            <pins>
               <pin name="Bulk" net="/gnd!"/>
               <pin name="Drain" net="/net20"/>
               <pin name="Source" net="/gnd!"/>
            </pins>
            <devices>
               <device name="/m7" deviceType="Mosfet" techType="n" instance="/"/>
            </devices>
         </structure>
      </biasPart>
   </BiasParts>
   <LoadParts>
      <loadPart>
         <structure name="MosfetDiodeArray[3]" techType="p" instance="/">
            <pins>
                <pin name="Bulk" net="/vdd!"/>
                <pin name="Drain" net="/net29"/>
                <pin name="Source" net="/vdd!"/>
            </pins>
            <devices>
               <device name="/m11" deviceType="Mosfet" techType="p" instance="/"/>
            </devices>
         </structure>
         <structure name="MosfetDiodeArray[5]" techType="p" instance="/">
            <pins>
               <pin name="Bulk" net="/vdd!"/>
               <pin name="Drain" net="/net40"/>
               <pin name="Source" net="/vdd!"/>
            </pins>
            <devices>
               <device name="/m13" deviceType="Mosfet" techType="p" instance="/"/>
            </devices>
         </structure>
      </loadPart>
   </LoadParts>
```

Figure B.10.: Example of an output file of the partitioning method [Part 4] (`acst/InputFileExamples/Partitioning/cascodedSymmetricalCMOSOTA.xml`)

```
      <CapacitanceParts>
         <capacitance type="load">
            <structure name="CapacitorArray[1]" techType="undefined" instance="/">
               <pins>
                  <pin name="Minus" net="/gnd!"/>
                  <pin name="Plus" net="/out"/>
               </pins>
               <devices>
                  <device name="/cl" deviceType="Capacitor" techType="undefined">
               </devices>
            </structure>
         </capacitance>
      </CapacitanceParts>
   </PartioningResults>
</xcat_results>
```

Figure B.11.: Example of an output file of the partitioning method [Part 5] (`acst/InputFileExamples/Partitioning/cascodedSymmetricalCMOSOTA.xml`)

B.4. Automatic Sizing

The automatic sizing method outputs a file containing the expected performance values, the currents and net voltage potentials at the dc-operation point, and the device dimensions calculated by the sizing algorithm.

```
<xcat_results>
   <date day="14" month="6" year="2021" hour="18" minute="5" second="8"/>
   <automatic_sizing-results>
      <ExpectedPerformance>
         <Gain unit="dB">97</Gain>
         <Power unit="m_W">5.40901</Power>
         <Area unit="(mu_m)^2">8942</Area>
         <TransitFrequency unit="M_Hz">28.6421</TransitFrequency>
         <TransitFrequencyWithErrorFactor unit="M_Hz">28.6423</TransitFrequencyWithError
          Factor>
         <SlewRate unit="V/mum_s">24.1242</SlewRate>
         <PhaseMargin unit="degree">64.7443</PhaseMargin>
         <CMRR unit="dB">143</CMRR>
         <negPSRR unit="dB">62</negPSRR>
         <posPSRR unit="dB">61</posPSRR>
         <MaximumOutputVoltage unit="V">4.31001</MaximumOutputVoltage>
         <MinimumOutputVoltage unit="V">0.75</MinimumOutputVoltage>
         <maxCommonModeInputVoltage unit="V">4.38001</maxCommonModeInputVoltage>
         <minCommonModeInputVoltage unit="V">0.720001</minCommonModeInputVoltage>
      </ExpectedPerformance>
      <Voltages unit="V">
         <Net name="/gnd!">0</Net>
         <Net name="/ibias">0.592001</Net>
         <Net name="/inn">2.5</Net>
         <Net name="/inp">2.5</Net>
         <Net name="/net20">1.15501</Net>
         <Net name="/net26">0.778001</Net>
         <Net name="/net27">4.47001</Net>
         <Net name="/net29">3.97001</Net>
         <Net name="/net30">3.68601</Net>
         <Net name="/net36">1.96301</Net>
         <Net name="/net40">3.97001</Net>
         <Net name="/net42">0.373001</Net>
         <Net name="/net44">4.47001</Net>
         <Net name="/net48">0.373001</Net>
         <Net name="/out">2.5</Net>
         <Net name="/vdd!">5</Net>
      </Voltages>
      <Currents unit="mu_A">
         <Component name="/m0">390.582</Component>
         <Component name="/m1">242.104</Component>
         <Component name="/m10">-242.102</Component>
         <Component name="/m11">-195.29</Component>
         <Component name="/m12">-242.102</Component>
         <Component name="/m13">-195.29</Component>
         <Component name="/m14">-242.103</Component>
         <Component name="/m15">-98.5739</Component>
         <Component name="/m16">-98.5379</Component>
         <Component name="/m17">98.5371</Component>
         <Component name="/m2">242.103</Component>
         <Component name="/m3">195.291</Component>
         <Component name="/m4">195.291</Component>
         <Component name="/m5">242.103</Component>
         <Component name="/m6">242.104</Component>
         <Component name="/m7">98.5731</Component>
         <Component name="/m8">9.99901</Component>
         <Component name="/m9">-242.103</Component>
      </Currents>
```

Figure B.12.: Example of an output file of the automatic sizing method [Part 1] (`acst/InputFileExamples/AutomaticSizing/cascodedSymmetricalCMOSOTA.xml`)

```
      <Dimensions>
         <Transistors>
            <Transistor name="/m8">
               <Width unit="mu_m">15.1001</Width>
               <Length unit="mu_m">4.5</Length>
            </Transistor>
            <Transistor name="/m7">
               <Width unit="mu_m">5.80001</Width>
               <Length unit="mu_m">2.80001</Length>
            </Transistor>
            <Transistor name="/m11">
               <Width unit="mu_m">86.1001</Width>
               <Length unit="mu_m">1.70001</Length>
            </Transistor>
            <Transistor name="/m16">
               <Width unit="mu_m">9.90001</Width>
               <Length unit="mu_m">1</Length>
            </Transistor>
            <Transistor name="/m13">
               <Width unit="mu_m">86.1001</Width>
               <Length unit="mu_m">1.70001</Length>
            </Transistor>
            <Transistor name="/m6">
               <Width unit="mu_m">56</Width>
               <Length unit="mu_m">2.80001</Length>
            </Transistor>
            <Transistor name="/m3">
               <Width unit="mu_m">490.901</Width>
               <Length unit="mu_m">3.70001</Length>
            </Transistor>
            <Transistor name="/m17">
               <Width unit="mu_m">148.301</Width>
               <Length unit="mu_m">4.5</Length>
            </Transistor>
            <Transistor name="/m0">
               <Width unit="mu_m">599.701</Width>
               <Length unit="mu_m">4.5</Length>
            </Transistor>
            <Transistor name="/m4">
               <Width unit="mu_m">490.901</Width>
               <Length unit="mu_m">3.70001</Length>
            </Transistor>
            <Transistor name="/m2">
               <Width unit="mu_m">57.1001</Width>
               <Length unit="mu_m">2.80001</Length>
            </Transistor>
            <Transistor name="/m5">
               <Width unit="mu_m">57.1001</Width>
               <Length unit="mu_m">2.80001</Length>
            </Transistor>
            <Transistor name="/m1">
               <Width unit="mu_m">56.1001</Width>
               <Length unit="mu_m">2.80001</Length>
            </Transistor>
            <Transistor name="/m15">
               <Width unit="mu_m">9.90001</Width>
               <Length unit="mu_m">1</Length>
            </Transistor>
            <Transistor name="/m14">
               <Width unit="mu_m">278.201</Width>
               <Length unit="mu_m">1</Length>
            </Transistor>
```

Figure B.13.: Example of an output file of the automatic sizing method [Part 2] (`acst/InputFileExamples/AutomaticSizing/cascodedSymmetricalCMOSOTA.xml`)

```
            <Transistor name="/m12">
               <Width unit="mu_m">105</Width>
               <Length unit="mu_m">1.70001</Length>
            </Transistor>
            <Transistor name="/m10">
               <Width unit="mu_m">105</Width>
               <Length unit="mu_m">1.70001</Length>
            </Transistor>
            <Transistor name="/m9">
               <Width unit="mu_m">278.201</Width>
               <Length unit="mu_m">1</Length>
            </Transistor>
         </Transistors>
         <Capacitors>
            <Capacitor name="/cl">
               <Value unit="p_F">20</Value>
            </Capacitor>
         </Capacitors>
      </Dimensions>
   </automatic_sizing-results>
</xcat_results>
```

Figure B.14.: Example of an output file of the automatic sizing method [Part 3] (`acst/InputFileExamples/AutomaticSizing/cascodedSymmetricalCMOSOTA.xml`)

B.5. Synthesis

In the following, an example file of a circuit is shown outputted by the synthesis process.

```
** Name: two_stage_single_output_op_amp_1_1

.MACRO two_stage_single_output_op_amp_1_1 ibias in1 in2 out sourceNmos sourcePmos
m1 FirstStageYout1 FirstStageYout1 sourceNmos sourceNmos nmos4 L=4e-6 W=53e-6
m2 ibias ibias sourcePmos sourcePmos pmos4 L=4e-6 W=4e-6
m3 out outFirstStage sourceNmos sourceNmos nmos4 L=2e-6 W=95e-6
m4 outFirstStage FirstStageYout1 sourceNmos sourceNmos nmos4 L=4e-6 W=53e-6
m5 out ibias sourcePmos sourcePmos pmos4 L=4e-6 W=36e-6
m6 outFirstStage in2 FirstStageYsourceTransconductance FirstStageYsourceTransconductance
    pmos4 L=4e-6 W=12e-6
m7 FirstStageYout1 in1 FirstStageYsourceTransconductance FirstStageYsourceTransconductan
    ce pmos4 L=4e-6 W=12e-6
m8 FirstStageYsourceTransconductance ibias sourcePmos sourcePmos pmos4 L=4e-6 W=20e-6
Capacitor1 out sourceNmos 20e-12
Capacitor2 outFirstStage out 4.5e-12
.EOM two_stage_single_output_op_amp_1_1

** Expected Performance Values:
** Gain: 83 dB
** Power consumption: 0.809001 mW
** Area: 950 (mu_m)^2
** Transit frequency: 2.59701 MHz
** Transit frequency with error factor: 2.58136 MHz
** Slew rate: 3.71006 V/mu_s
** Phase margin: 63.5984°
** CMRR: 88 dB
** negPSRR: 90 dB
** posPSRR: 196 dB
** VoutMax: 4.25 V
** VoutMin: 0.150001 V
** VcmMax: 3 V
** VcmMin: -0.00999999 V


** Expected Currents:
** DiodeTransistorNmos: 2.53831e+07 muA
** NormalTransistorNmos: 2.53831e+07 muA
** NormalTransistorPmos: -5.07669e+07 muA
** NormalTransistorPmos: -2.53839e+07 muA
** NormalTransistorPmos: -2.53839e+07 muA
** NormalTransistorNmos: 9.09981e+07 muA
** NormalTransistorPmos: -9.09989e+07 muA
** DiodeTransistorPmos: -9.99899e+06 muA


** Expected Voltages:
** ibias: 3.68601  V
** in1: 2.5  V
** in2: 2.5  V
** out: 2.5  V
** outFirstStage: 0.555001  V
** sourceNmos: 0  V
** sourcePmos: 5  V
** out1: 0.555001  V
** sourceTransconductance: 3.74901  V


.END
```

Figure B.15.: Example of a circuit netlist file created by the synthesis algorithm (`acst/InputFileExamples/Synthesis/HspiceNetlist/two_stage_single_output_op_amp_1_1.ckt`)

B.6. Topology Library Generation

In the following, an example file of a circuit is shown outputted by the topology library generation method.

```
.suckt  two_stage_single_output_op_amp_1_12 ibias in1 in2 out sourceNmos sourcePmos
c1 outFirstStage out
m1 outVoltageBiasXXpXX1 outSourceVoltageBiasXXnXX1 sourceNmos sourceNmos nmos
m2 inputVoltageBiasXXpXX2 outSourceVoltageBiasXXnXX1 sourceNmos sourceNmos nmos
m3 FirstStageYout1 FirstStageYout1 sourceNmos sourceNmos nmos
m4 outFirstStage FirstStageYout1 sourceNmos sourceNmos nmos
m5 FirstStageYsourceTransconductance inputVoltageBiasXXpXX2 sourcePmos sourcePmos pmos
m6 FirstStageYout1 in1 FirstStageYsourceTransconductance FirstStageYsourceTransconductan
    ce pmos
m7 outFirstStage in2 FirstStageYsourceTransconductance FirstStageYsourceTransconductance
     pmos
c2 out sourceNmos
m8 out ibias outSourceVoltageBiasXXnXX1 outSourceVoltageBiasXXnXX1 nmos
m9 outSourceVoltageBiasXXnXX1 outSourceVoltageBiasXXnXX1 sourceNmos sourceNmos nmos
m10 out outVoltageBiasXXpXX1 SecondStageYinnerTransconductance SecondStageYinnerTranscon
    ductance pmos
m11 SecondStageYinnerTransconductance outFirstStage sourcePmos sourcePmos pmos
m12 ibias ibias VoltageBiasXXnXX1Yinner VoltageBiasXXnXX1Yinner nmos
m13 VoltageBiasXXnXX1Yinner outSourceVoltageBiasXXnXX1 sourceNmos sourceNmos nmos
m14 outVoltageBiasXXpXX1 outVoltageBiasXXpXX1 sourcePmos sourcePmos pmos
m15 inputVoltageBiasXXpXX2 inputVoltageBiasXXpXX2 sourcePmos sourcePmos pmos
.end two_stage_single_output_op_amp_1_12
```

Figure B.16.: Example of a circuit file created by the topology library generation method (`acst/InputFileExamples/TopologyLibraryGeneration/NetlistsWithoutLabels/SingleOutputOpAmps/two_stage_single_output_op_amp1_12.ckt`)

If the structure library (App. A.7) is provided in the commands as well as the device-types-file (App. A.4), following file is outputted (Sec. 3.2.5):

```
.suckt  two_stage_single_output_op_amp_1_12 ibias in1 in2 out sourceNmos sourcePmos
c_SingleOutput_Compensation_Capacitor_1 outFirstStage out
m_SingleOutput_MainBias_1 outVoltageBiasXXpXX1 outSourceVoltageBiasXXnXX1 sourceNmos sou
     rceNmos nmos
m_SingleOutput_MainBias_2 inputVoltageBiasXXpXX2 outSourceVoltageBiasXXnXX1 sourceNmos s
     ourceNmos nmos
m_SingleOutput_FirstStage_Load_3 FirstStageYout1 FirstStageYout1 sourceNmos sourceNmos n
     mos
m_SingleOutput_FirstStage_Load_4 outFirstStage FirstStageYout1 sourceNmos sourceNmos nmo
     s
m_SingleOutput_FirstStage_StageBias_5 FirstStageYsourceTransconductance inputVoltageBias
     XXpXX2 sourcePmos sourcePmos pmos
m_SingleOutput_FirstStage_Transconductor_6 FirstStageYout1 in1 FirstStageYsourceTranscon
     ductance FirstStageYsourceTransconductance pmos
m_SingleOutput_FirstStage_Transconductor_7 outFirstStage in2 FirstStageYsourceTranscondu
     ctance FirstStageYsourceTransconductance pmos
c_SingleOutput_Load_Capacitor_2 out sourceNmos
m_SingleOutput_SecondStage1_StageBias_8 out ibias outSourceVoltageBiasXXnXX1 outSourceVo
     ltageBiasXXnXX1 nmos
m_SingleOutput_SecondStage1_StageBias_9 outSourceVoltageBiasXXnXX1 outSourceVoltageBiasX
     XnXX1 sourceNmos sourceNmos nmos
m_SingleOutput_SecondStage1_Transconductor_10 out outVoltageBiasXXpXX1 SecondStageYinner
     Transconductance SecondStageYinnerTransconductance pmos
m_SingleOutput_SecondStage1_Transconductor_11 SecondStageYinnerTransconductance outFirst
     Stage sourcePmos sourcePmos pmos
m_SingleOutput_MainBias_12 ibias ibias VoltageBiasXXnXX1Yinner VoltageBiasXXnXX1Yinner n
     mos
m_SingleOutput_MainBias_13 VoltageBiasXXnXX1Yinner outSourceVoltageBiasXXnXX1 sourceNmos
      sourceNmos nmos
m_SingleOutput_SecondStage1_StageBias_14 outVoltageBiasXXpXX1 outVoltageBiasXXpXX1 sourc
     ePmos sourcePmos pmos
m_SingleOutput_MainBias_15 inputVoltageBiasXXpXX2 inputVoltageBiasXXpXX2 sourcePmos sour
     cePmos pmos
.end two_stage_single_output_op_amp_1_12
```

Figure B.17.: Example of a circuit file created by the topology library generation method (`acst/InputFileExamples/TopologyLibraryGeneration/NetlistsWithLabels/SingleOutputOpAmps/two_stage_single_output_op_amp1_12.ckt`)

C. Examples of Supported Op-Amps

In the following, examples are shown of supported op-amp topologies.

C.1. Symmetrical Op-Amps

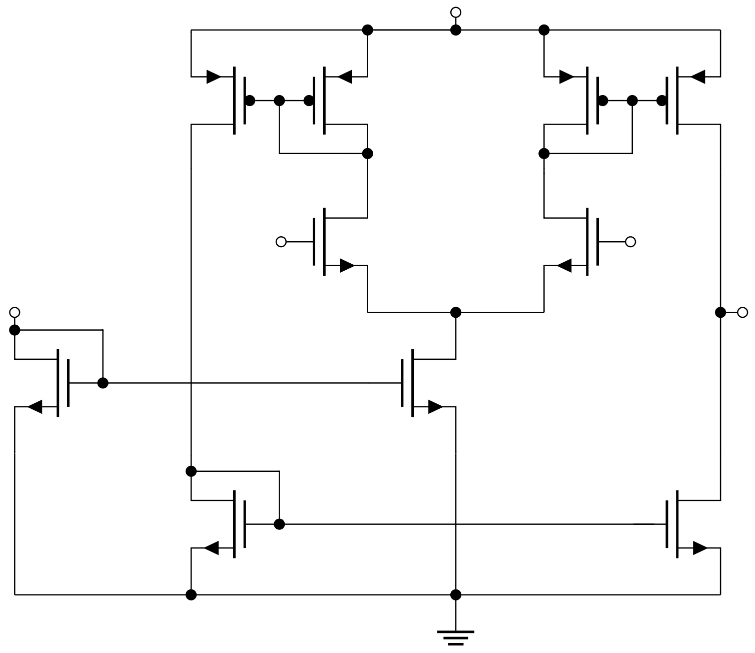

Figure C.1.: Symmetrical op-amp

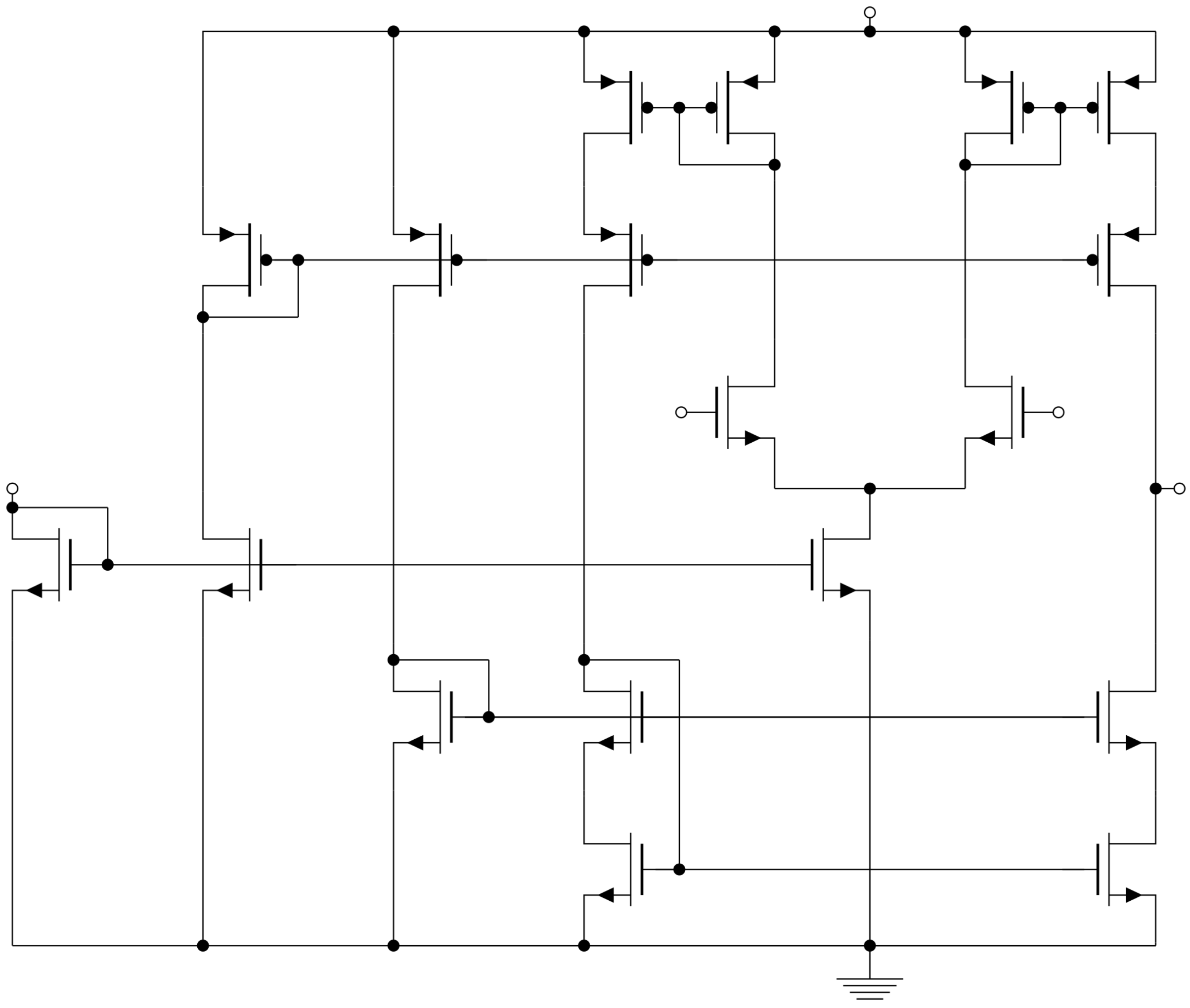

Figure C.2.: Cascode symmetrical op-amp

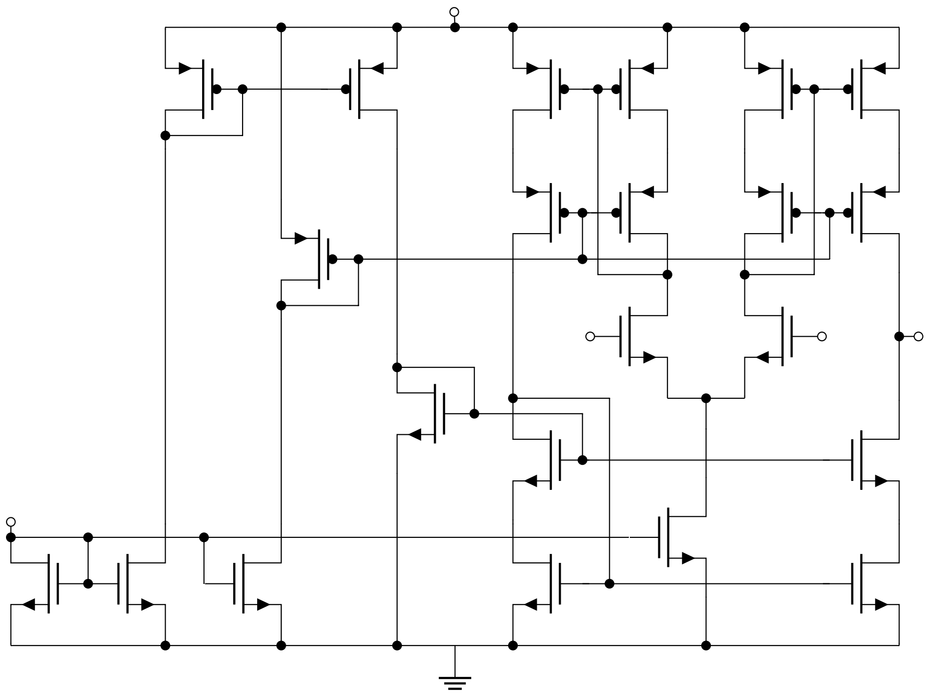
Figure C.3.: Cascode symmetrical op-amp with cascode first and second stage

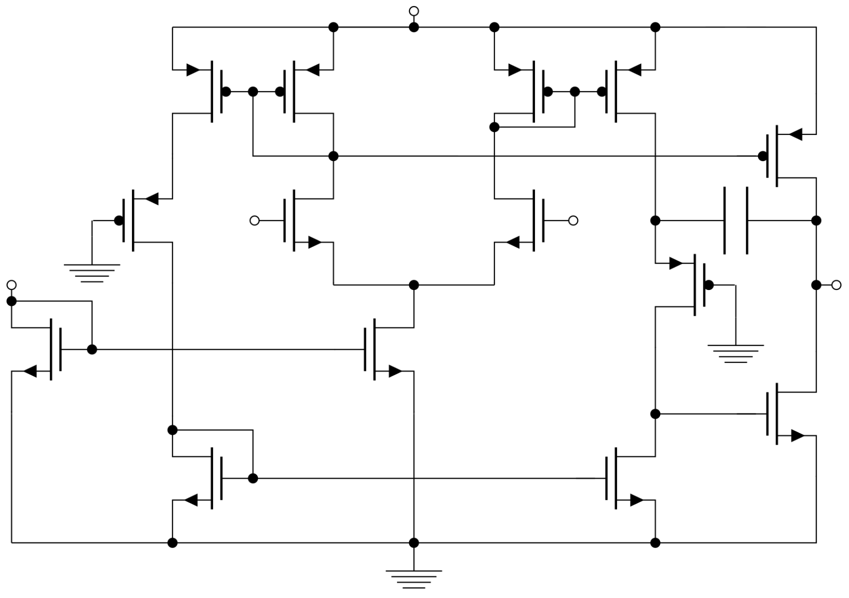
Figure C.4.: Symmetrical op-amp with high PSRR [10]

C.2. Single-Output Op-Amps

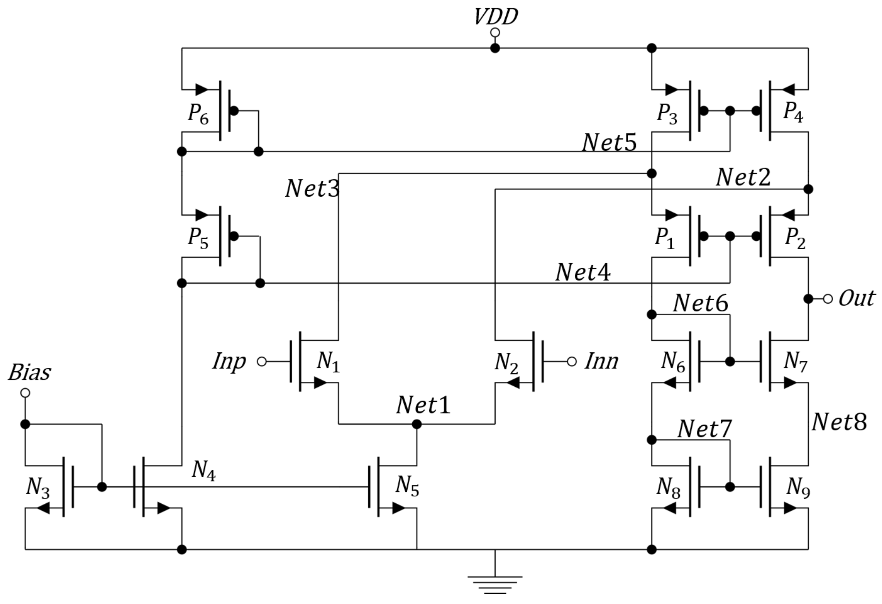

Figure C.5.: Folded-cascode op-amp with nmos differential stage

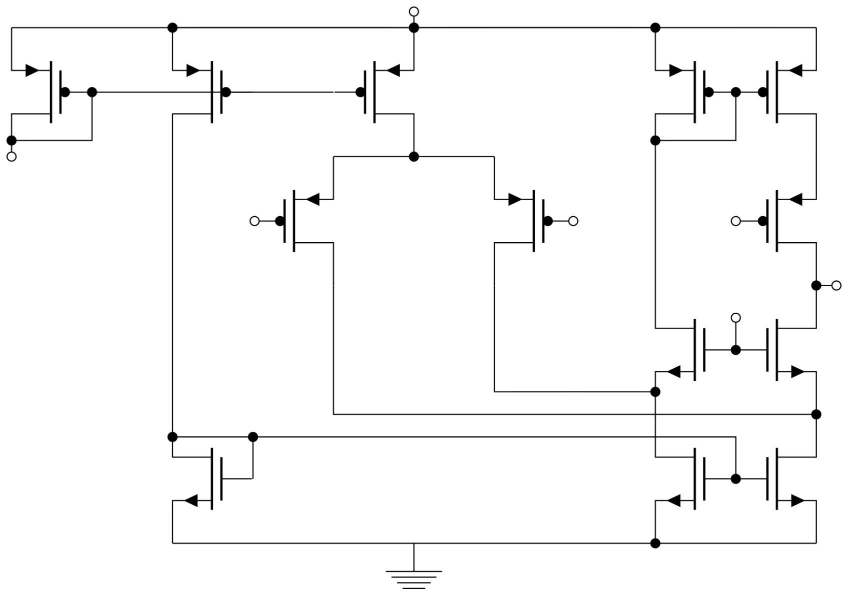

Figure C.6.: Folded-cascode op-amp with pmos differential stage

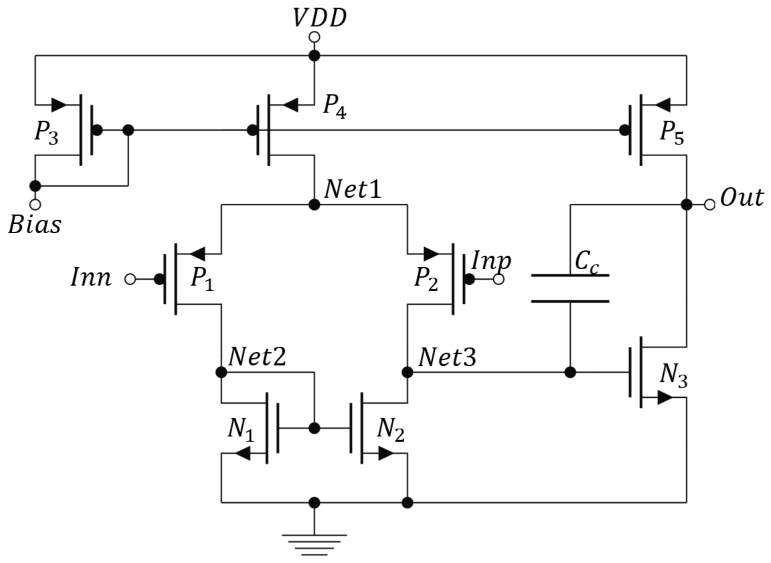

Figure C.7.: Miller op-amp

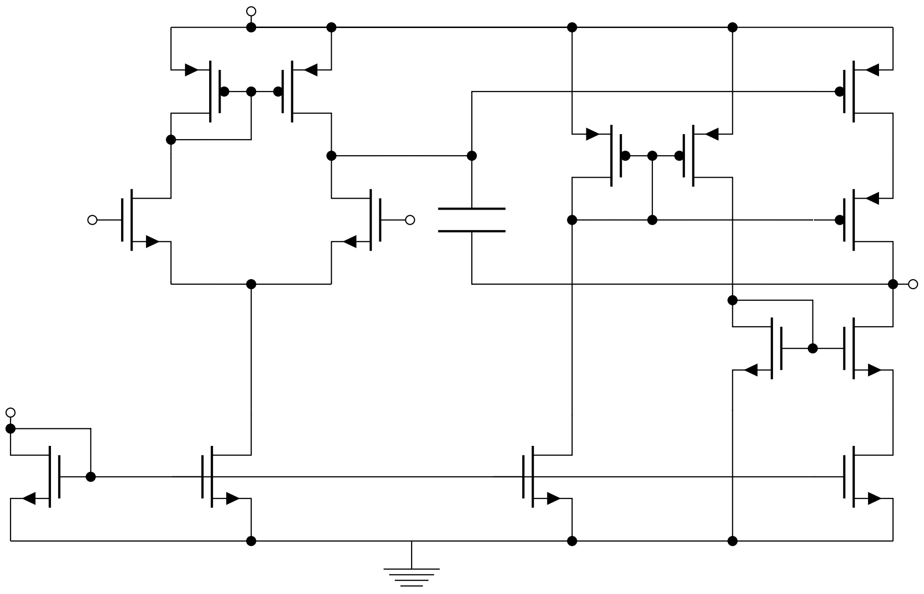

Figure C.8.: Two-stage op-amp with cascode second stage

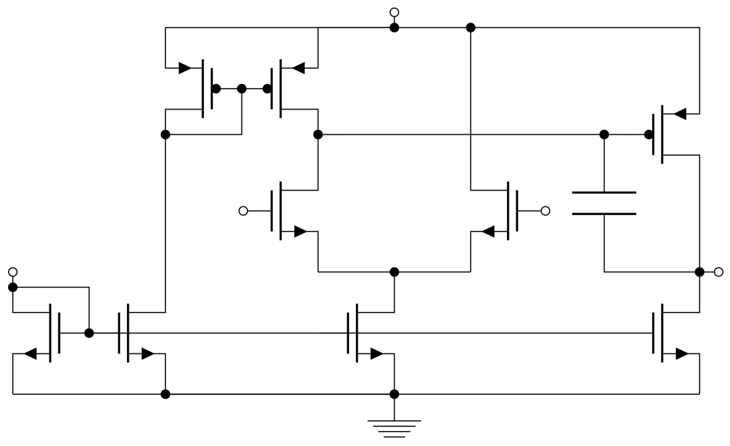

Figure C.9.: Two-stage op-amp with one load transistor [11]

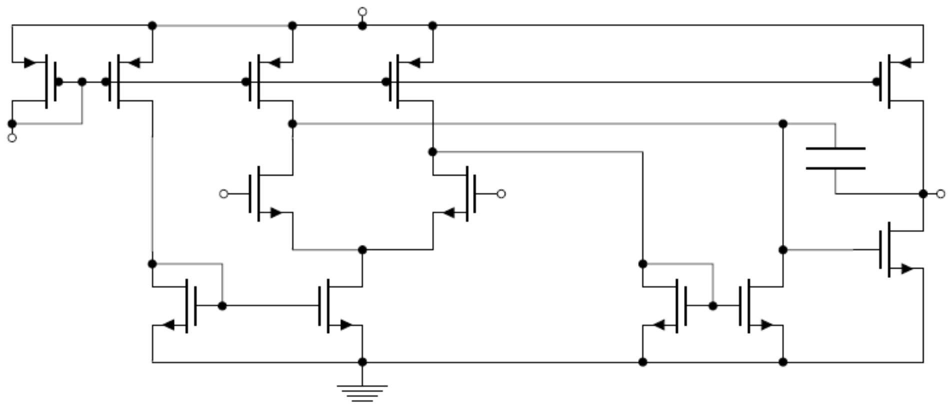

Figure C.10.: Low-power op-amp [11]

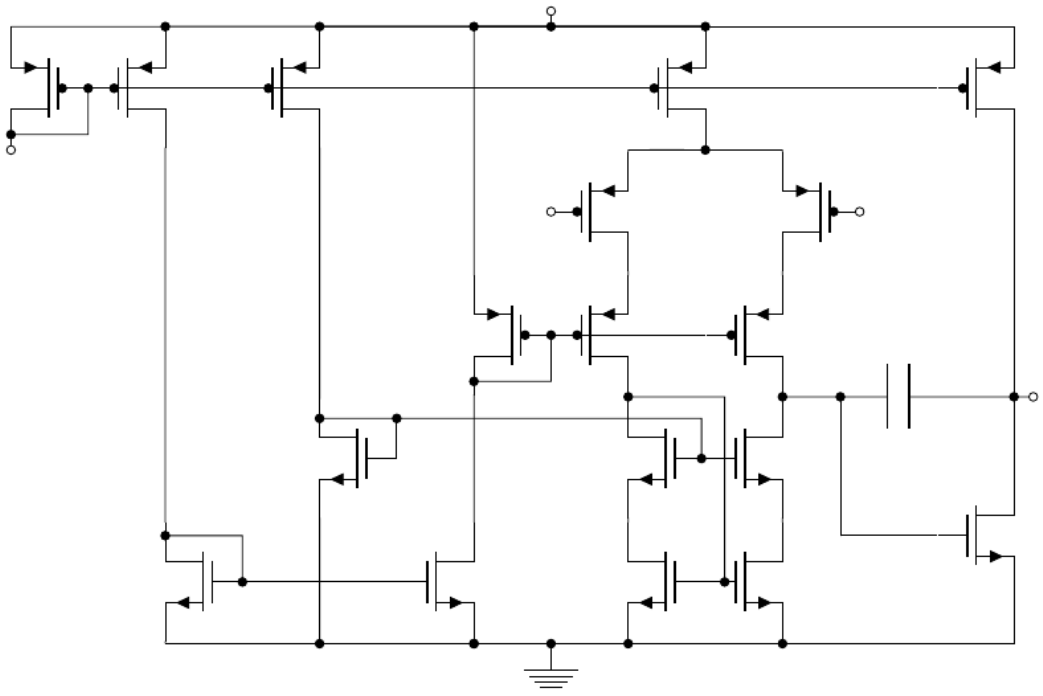

Figure C.11.: Telescopic op-amp

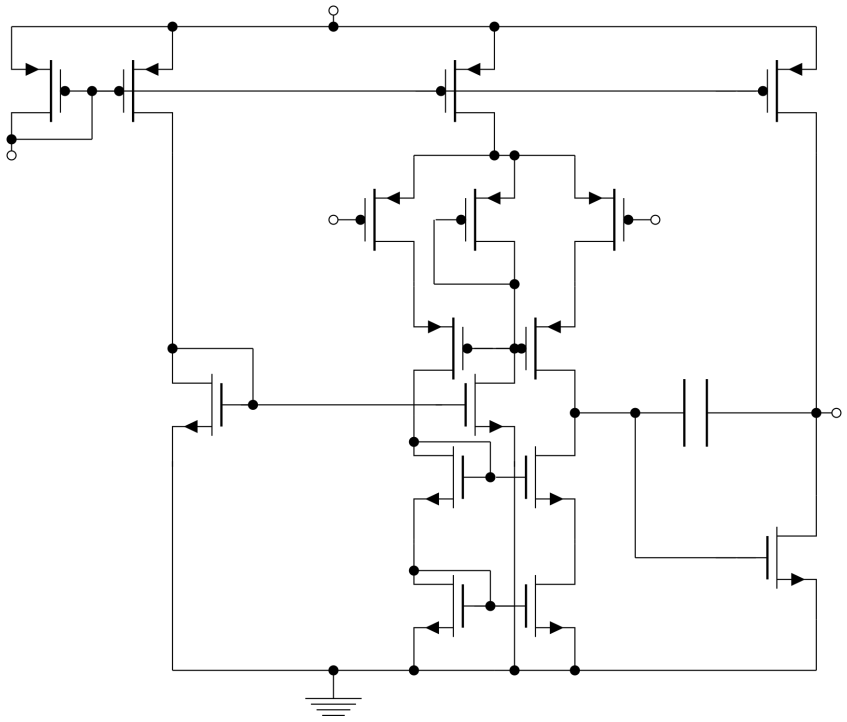

Figure C.12.: Telescopic op-amp with different bias circuit

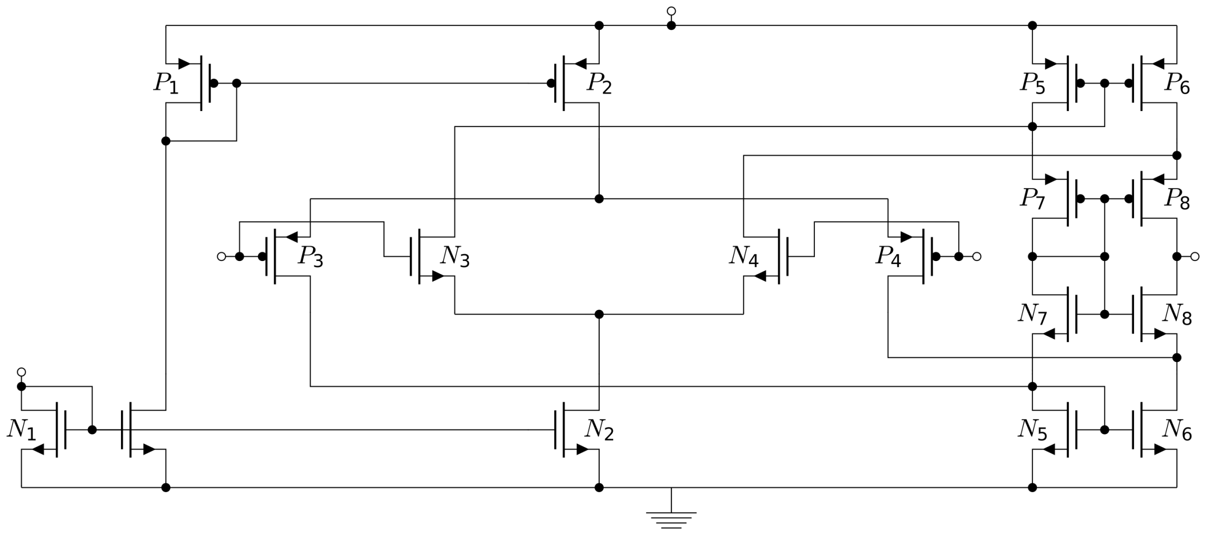

Figure C.13.: Complementary op-amp

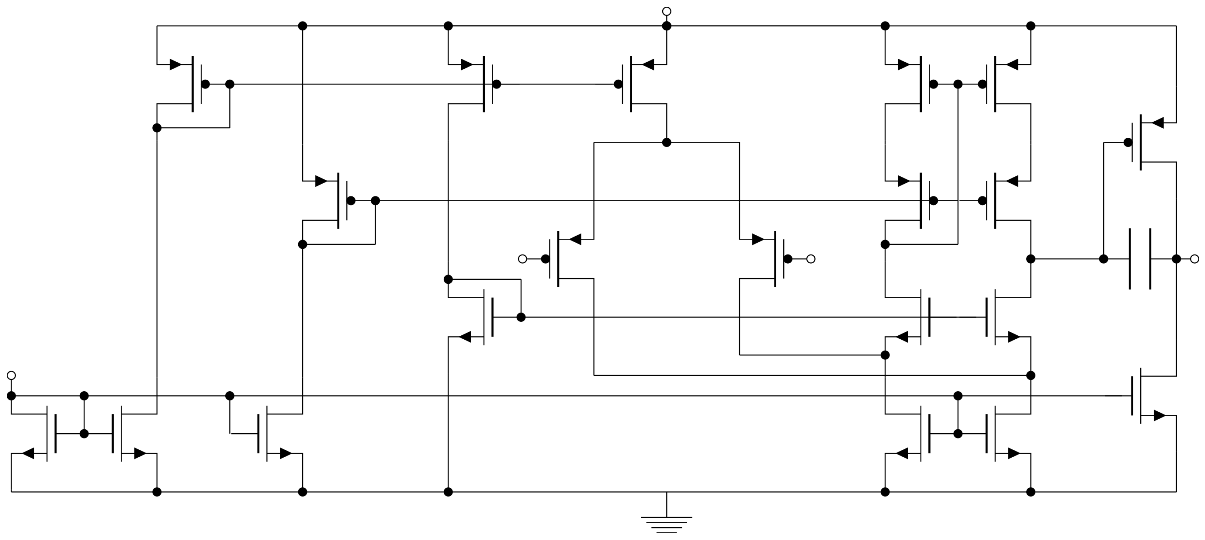

Figure C.14.: Two-stage folded-cascode op-amp

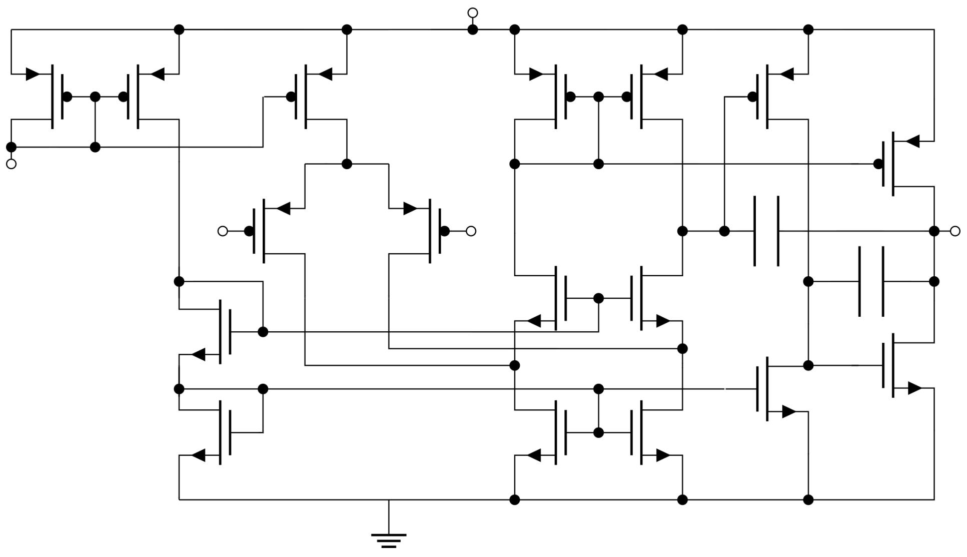

Figure C.15.: Three-stage op-amp

C.3. Fully-differential Op-Amps

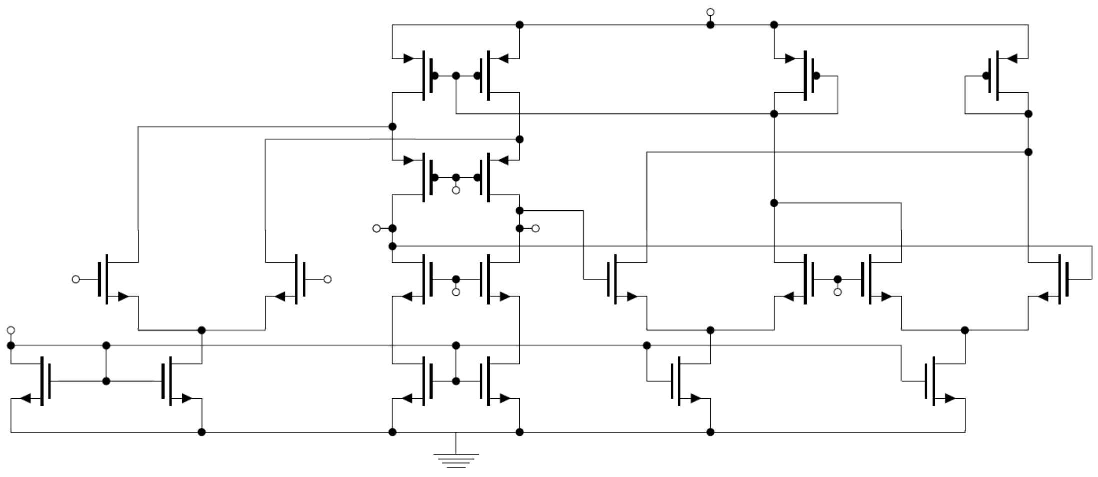

Figure C.16.: Folded-cascode op-amp with CMFB stage

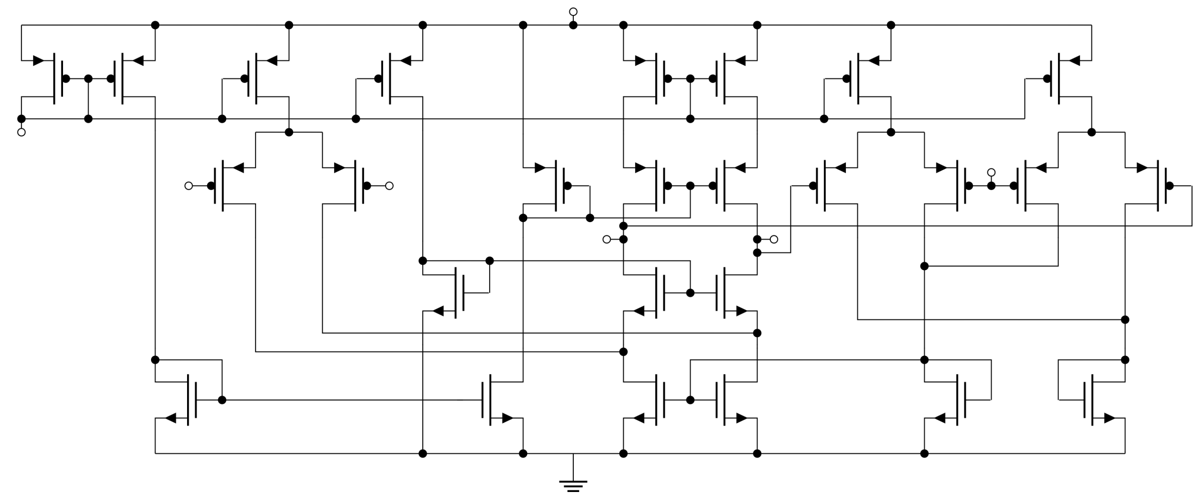
Figure C.17.: Folded-cascode op-amp with CMFB stage PMOS variant

D. Installation Guide for Supplementary Libraries Needed by acst

The following installation guide does not claim to be complete and without errors. Please refer to the websites of the individual libraries for further information.

D.1. Installation of Boost

On most systems, Boost is already installed. You can check that with:

```
$ apt list --installed libboost-all-dev
```

Or look at:

```
/usr/local/lib/cmake/
```

If it is not installed, you can download boost at `https://www.boost.org/` and install it with the following command in the directory of Boost:

```
$ ./bootstrap
$ ./b2
$ sudo ./b2 install
```

D.2. Installation of RapidXml

You can download RapidXml at `http://rapidxml.sourceforge.net/`. No installation is needed. However, there is a bug in `rapidxml_print.hpp` if clang is used as compiler. You must add following lines to `rapidxml_print.hpp` as bug fix:

```
///////////////////////////////////////////////////////////////////////////
// Internal printing operations

//Predeclarations for bugfix
template<class OutIt, class Ch>
inline OutIt print_children(OutIt out, const xml_node<Ch> *node, int flags, int indent);

template<class OutIt, class Ch>
inline OutIt print_attributes(OutIt out, const xml_node<Ch> *node, int flags);

template<class OutIt, class Ch>
inline OutIt print_data_node(OutIt out, const xml_node<Ch> *node, int flags, int
indent);

template<class OutIt, class Ch>
inline OutIt print_cdata_node(OutIt out, const xml_node<Ch> *node, int flags, int
indent);

template<class OutIt, class Ch>
inline OutIt print_element_node(OutIt out, const xml_node<Ch> *node, int flags, int
indent);

template<class OutIt, class Ch>
inline OutIt print_declaration_node(OutIt out, const xml_node<Ch> *node, int flags, int
indent);

template<class OutIt, class Ch>
inline OutIt print_comment_node(OutIt out, const xml_node<Ch> *node, int flags, int
indent);

template<class OutIt, class Ch>
inline OutIt print_doctype_node(OutIt out, const xml_node<Ch> *node, int flags, int
indent);

template<class OutIt, class Ch>
inline OutIt print_pi_node(OutIt out, const xml_node<Ch> *node, int flags, int indent);
//Test for bugfix

// Print node
template<class OutIt, class Ch>
inline OutIt print_node(OutIt out, const xml_node<Ch> *node, int flags, int indent)
```

D.3. Installation of GeCode

For the correct installation of GeCode (`https://www.gecode.org/`) the GMP library (`https://gmplib.org/`) and the MPFR (`https://www.mpfr.org/`) are needed, as they provide the needed support for trigonometrical and transcendental constraints.

On some systems, GMP and MPFR are preinstalled. You can check that by running the command `./configure` in the GeCode directory. Look at the output and check if gmp.h and mpfr.h was found.

If they are not installed, download these libraries from the website and install them by using the commands:

```
$ ./configure
$ make
$ sudo make install
```

in their directory.

GMP depends on M4. If not already installed, you can install it with the following commands:

```
$ sudo apt-get update
$ sudo apt-get install m4
```

GeCode must be downloaded from the GeCode web page (`https://www.gecode.org/`). It is installed using the commands:

```
$ ./configure
$ make
```

in the GeCode directory.

Bibliography

[1] T. Massier, H. Graeb, and U. Schlichtmann, “The Sizing Rules Method for CMOS and Bipolar Analog Integrated Circuit Synthesis,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2008.

[2] M. Neuner, I. Abel, and H. Graeb, “Library-free structure recognition for analog circuits,” in *Design, Automation Test in Europe Conference Exhibition (DATE*, 2021.

[3] I. Abel, M. Neuner, and H. Graeb, “A Functional Block Decomposition Method for Automatic Op-Amp Design,” (Dec, 2020). [Online]. Available: https://arxiv.org/abs/2012.09051

[4] ——, “A hierarchical performance equation library for basic op-amp design,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pp. 1–1, 2021.

[5] I. Abel and H. Graeb, “Structural Synthesis of Operational Amplifiers Based on Functional Block Modeling,” in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2020.

[6] I. Abel and H. Graeb, “FUBOCO: Structure Synthesis of Basic Op-Amps by FUnctional BlOck COmposition,” 2021, . [Online]. Available: http://arxiv.org/abs/2101.07517

[7] T. Massier, “On the structural analysis of cmos and bipolar analog integrated circuits,” Ph.D. dissertation, Technische Universität München, 2010.

[8] I. Abel, C. Kowalsky, and H. Graeb, “A fast Structural Synthesis Algorithm for Op-Amps based on Multi-Threading Strategies,” in *International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD)*, 2021.

[9] *Cadence*, www.cadence.com, Cadence Design Systems, 2018. [Online]. Available: www.cadence.com

[10] K. R. Laker and W. M. C. Sansen, *Design of analog integrated circuits and systems.* McGraw-Hill, 1994.

[11] D. R. H. Phillip E. Allan, *CMOS Analog Circuit Design.* Oxford University Press, 2012.