topology_api

Generated by Doxygen 1.9.3

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 3

Class Documentation

3.1 API Class Reference

Public Member Functions

- API ()

  Default constructor.
- bool readJSON (string fileName)

  Stores a topology in the memory.
- bool writeJSON (string topologyID)

  Writes a topology to a JSON file.
- list< Topology * > * queryTopologies ()

  Query about the topologies stored in the memory.
- bool deleteTopology (string topologyID)

  Delete a toplogy from the memory.
- list< Component * > * queryComponents (string topologyID)

  Query about the components of a topology.
- list< Component * > * queryComponentsWithNetlistNode (string topologyID, string netlistNodeID)

  Query about components of a topology that are connected to a netlist node.
- ~**API** ()

  Destroy the API object by deleting all the topology objects pointed to by the pointers of the saved topologies (in the list) and then clearing the list of the topology pointers.

Private Member Functions

- Topology * getTopologyFromID (string topologyID)

  A utility function that returns a pointer to a topology given its ID.

Private Attributes

- list< Topology * > * **topologies**

  A list the saves the addresses of the topologies that are currently stored in the memory.

3.1.1 Constructor & Destructor Documentation

3.1.1.1 API()

```
API::API ( )
```

Default constructor.

Construct a new API object with an empty list of topology pointers

3.1.2 Member Function Documentation

3.1.2.1 getTopologyFromID()

```
Topology * API::getTopologyFromID (
            string topologyID )  [private]
```

A utility function that returns a pointer to a topology given its ID.

This function is used by many member functions to find a topology with a certain ID and it is a private function because it is considered as non-core functionality for the API

Parameters

topologyID	The ID of required topology to be returned

Returns

The address of the topology object

3.1.2.2 readJSON()

```
bool API::readJSON (
            string fileName )
```

Stores a topology in the memory.

This function reads a topology JSON file which is located in "topology_input_files" directory and store the object in the memory

Parameters

fileName	The name of the topology json file (for example: "topology1.json")

Returns

true if the topology is stored in the memory successfully

false if there is another topology stored in the memory with the same ID of the topology in the input JSON file

3.1.2.3 writeJSON()

```
bool API::writeJSON (
            string topologyID )
```

Writes a topology to a JSON file.

This function writes a topology of given ID to a JSON file, the name of the file is the same as the ID of the topology and the output file is located in "topology_output_directory" directory

Parameters

topologyID	The ID of the topology to be written in the JSON file

Returns

true if the topology is written successfully to the JSON file

false if there is no topology with the given ID stored in the memory

3.1.2.4 queryTopologies()

```
list< Topology * > * API::queryTopologies ( )
```

Query about the topologies stored in the memory.

Returns

A list of addresses of the topologies that are currently stored in the memory

3.1.2.5 deleteTopology()

```
bool API::deleteTopology (
            string topologyID )
```

Delete a toplogy from the memory.

This function deletes a topology of given ID from the memory and delete its address from the list of saved topology addresses

Parameters

topologyID	The ID of the topology to be deleted

Returns

true if the topology is deleted successfully from the memory

false if there is no topology with the given ID stored in the memory

3.1.2.6 queryComponents()

```
list< Component * > * API::queryComponents (
            string topologyID )
```

Query about the components of a topology.

This function searches for a topology of given ID using the private member function 'getTopologyFromID' and returns its components if the topology exists otherwise it returns a null pointer (if there is no topology with the given ID)

Parameters

topologyID	The ID of the topology that contains the required componenets

Returns

A list of Component object addresses (the addresses may be of an 'NMOS' or a 'Resistor' object)

3.1.2.7 queryComponentsWithNetlistNode()

```
list< Component * > * API::queryComponentsWithNetlistNode (
            string topologyID,
            string netlistNodeID )
```

Query about components of a topology that are connected to a netlist node.

This function searches for a topology of given ID using the private member function 'getTopologyFromID' and iterates through its components and filter them according to whether or not the component is connect to the netlist node of given ID and returns a list of the filtered components (the size of the returned list maybe zero if there are no components connected to the given netlist node), the function may return null pointer if there is no topology of the given ID stored in the memory

Parameters

topologyID	The ID of the topology that contains the required componenets
netlistNodeID	The ID of the netlist node that the required components are connected to

Returns

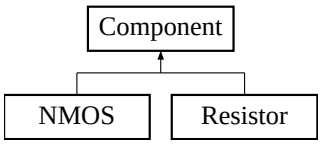
A list of Component object addresses (the addresses may be of an 'NMOS' or a 'Resistor' object)

The documentation for this class was generated from the following files:

- API.h
- API.cpp

3.2 Component Class Reference

Inheritance diagram for Component:



Public Member Functions

- Component (string ID, map< string, string > netlist)

  Creates a new Component object.
- map< string, string > getNetlist ()

  Gets the Netlist of the component.
- virtual void toJson (json *jsonFile)

  The function sets the ID and the netlist of the component in the JSON object (because all the derived classes share those attributes), this function is called by derived classes to set the common attributes.
- virtual void print ()=0

Protected Attributes

- string **ID**

  The ID of the component.
- map< string, string > **netlist**

  The netlist of the component.

3.2.1 Constructor & Destructor Documentation

3.2.1.1 Component()

```
Component::Component (
            string ID,
            map< string, string > netlist )
```

Creates a new Component object.

Construct a new component object and sets its parameters with the sent parametrs

Parameters

ID	The ID of the component
netlist	The netlist of the component

3.2.2 Member Function Documentation

3.2.2.1 getNetlist()

```
map< string, string > Component::getNetlist ( )
```

Gets the Netlist of the component.

Returns

The netlist of the component

3.2.2.2 toJson()

```
void Component::toJson (
            json * jsonFile )  [virtual]
```

The function sets the ID and the netlist of the component in the JSON object (because all the derived classes share those attributes), this function is called by derived classes to set the common attributes.

This function converts the component to a JSON to be used in the API, it is overriden by the derived classes (Resistor & NMOS) because they have different attributes to be converted to JSON object

Parameters

jsonFile	A pointer to the JSON object that is used to convert the component to JSON object

Reimplemented in NMOS, and Resistor.

3.2.2.3 print()

```
virtual void Component::print ( )  [pure virtual]
```

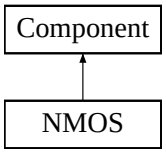
Implemented in NMOS, and Resistor.

The documentation for this class was generated from the following files:

- component.h
- component.cpp

3.3 NMOS Class Reference

Inheritance diagram for NMOS:



Public Member Functions

- NMOS (string ID, map< string, float > ml, map< string, string > netlist)

  Creates a new NMOS object.
- virtual void print ()

  Prints the attributes of the NMOS object in the console window.
- virtual void toJson (json *jsonFile)

  This function sets the m(l) of the NMOS object and calls Component::toJson(jsonFile) to convert the other parameters to a JSON object.
- map< string, string > getNetlist ()

  Gets the Netlist of the component.

Protected Attributes

- string **ID**

  The ID of the component.
- map< string, string > **netlist**

  The netlist of the component.

Private Attributes

- map< string, float > **ml**

  The m(l) of the NMOS transistor.

3.3.1 Constructor & Destructor Documentation

3.3.1.1 NMOS()

```
NMOS::NMOS (
            string ID,
            map< string, float > ml,
            map< string, string > netlist )
```

Creates a new NMOS object.

Construct a new NMOS object and sets its parameters and the Component parameters with the sent parameters

Parameters

ID	The ID of the NMOS component
ml	The m(l) of the NMOS component
netlist	The netlist of the NMOS componet

3.3.2 Member Function Documentation

3.3.2.1 print()

```
void NMOS::print ( )  [virtual]
```

Prints the attributes of the NMOS object in the console window.

Implements Component.

3.3.2.2 toJson()

```
void NMOS::toJson (
            json * jsonFile )  [virtual]
```

This function sets the m(l) of the NMOS object and calls Component::toJson(jsonFile) to convert the other parameters to a JSON object.

Parameters

jsonFile	A pointer to the JSON object that is used to convert the component to JSON object

Reimplemented from Component.

3.3.2.3 getNetlist()

```
map< string, string > Component::getNetlist ( )  [inherited]
```

Gets the Netlist of the component.

Returns

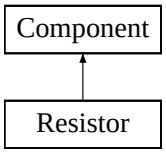
The netlist of the component

The documentation for this class was generated from the following files:

- NMOS.h
- NMOS.cpp

3.4 Resistor Class Reference

Inheritance diagram for Resistor:



Public Member Functions

- Resistor (string ID, map< string, int > resistance, map< string, string > netlist)

  Creates a new Resistor object.
- virtual void print ()

  Prints the attributes of the Resistor object in the console window.
- virtual void toJson (json *jsonFile)

  This function sets the resistance of the Resistance object and calls Component::toJson(jsonFile) to convert the other parameters to a JSON object.
- map< string, string > getNetlist ()

  Gets the Netlist of the component.

Protected Attributes

- string **ID**

  The ID of the component.
- map< string, string > **netlist**

  The netlist of the component.

Private Attributes

- map< string, int > **resistance**

  The reistance of the Resistor.

3.4.1 Constructor & Destructor Documentation

3.4.1.1 Resistor()

```
Resistor::Resistor (
            string ID,
            map< string, int > resistance,
            map< string, string > netlist )
```

Creates a new Resistor object.

Construct a new Resistor object and sets its parameters and the Component parameters with the sent parameters

Parameters

ID	The ID of the Reistor component
resistance	The resistance of the Resistor component
netlist	The netlist of the resistor component

3.4.2 Member Function Documentation

3.4.2.1 print()

```
void Resistor::print ( )  [virtual]
```

Prints the attributes of the Resistor object in the console window.

Implements Component.

3.4.2.2 toJson()

```
void Resistor::toJson (
            json * jsonFile )  [virtual]
```

This function sets the resistance of the Resistance object and calls Component::toJson(jsonFile) to convert the other parameters to a JSON object.

Parameters

jsonFile	A pointer to the JSON object that is used to convert the component to JSON object

Reimplemented from Component.

3.4.2.3 getNetlist()

```
map< string, string > Component::getNetlist ( )  [inherited]
```

Gets the Netlist of the component.

Returns

The netlist of the component

The documentation for this class was generated from the following files:

- resistor.h
- resistor.cpp

3.5 Topology Class Reference

Public Member Functions

- Topology (string ID, list< Component * > *components)

  Create a new Topology object.
- void **print** ()

  Prints the ID of the topology and the components by iterating through the list of components and calling Component::print() (a virtual function)
- string getID ()

  Gets the ID of the topology.
- list< Component * > * getComponents ()

  Get the components of the topology.
- ~**Topology** ()

  Destroy the Topology object by deleting all the Component objects pointed to by the pointers of the saved components (in the list) and then clearing the list of the component pointers.

Private Attributes

- string **ID**

  The ID of the topology.
- list< Component * > * **components**

  A list the saves the addresses of the components that are contained in the topology.

3.5.1 Constructor & Destructor Documentation

3.5.1.1 Topology()

```
Topology::Topology (
            string ID,
            list< Component * > * components )
```

Create a new Topology object.

Construct a new Topology object and sets its parameters with the sent parameters

Parameters

ID	The ID of the topology
components	A list of component pointers to create a topology

3.5.2 Member Function Documentation

3.5.2.1 getID()

```
string Topology::getID ( )
```

Gets the ID of the topology.

Returns

ID The ID of the topology

3.5.2.2 getComponents()

```
list< Component * > * Topology::getComponents ( )
```

Get the components of the topology.

Returns

A list of component pointers

The documentation for this class was generated from the following files:

- topology.h
- topology.cpp