TUC

Generated by Doxygen 1.9.6

Chapter 1

Class Index

1.1 Class List

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

Chapter 2

File Index

2.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 3

Class Documentation

3.1 Gate Struct Reference

Structure used for simulating a single gate.

`#include <gate.hpp>`

Public Attributes

- GateType **type** = GateType::UNDEFINED

  Gate type.
- std::vector< size_t > **inputs**

  Inputs (other nodes that the gate needs to generate output)
- GateValue **value** = GateValue::UNDEFINED

  Output value.

3.1.1 Detailed Description

Structure used for simulating a single gate.

The documentation for this struct was generated from the following file:

- gate.hpp

3.2 Input Struct Reference

Structure used for reading inputs/writing outputs.

`#include <parser_input.hpp>`

Public Attributes

- size_t **nodeID**

  ID of the node.
- bool **value**

  Binary representation of the state of the node.

3.2.1 Detailed Description

Structure used for reading inputs/writing outputs.

The documentation for this struct was generated from the following file:

- parser_input.hpp

3.3 ParsedGate Struct Reference

Contains information about a specific gate/input (intermediate struct between text and struct Gate)

`#include <parser_circuit.hpp>`

Public Attributes

- size_t **lineNumber**

  Number of line that the gate was read from.
- GateType **type** = GateType::UNDEFINED

  Gate type (defined in gate.hpp)
- std::vector< size_t > **nodes**

  Inputs and outputs for normal gates or input/output nodes when the type is GateType::INPUT or GateType::OUTPUT.

3.3.1 Detailed Description

Contains information about a specific gate/input (intermediate struct between text and struct Gate)

The documentation for this struct was generated from the following file:

- parser_circuit.hpp

Chapter 4

File Documentation

4.1 gate.hpp File Reference

Structures, enums and functions related to the gate representation.

```
#include <vector>
#include <cstddef>
#include <map>
```

Classes

- struct Gate

  Structure used for simulating a single gate.

Enumerations

- enum class GateType {
  UNDEFINED , **INPUT** , **OUTPUT** , **AND** ,
  NAND , **OR** , **NOR** , **XOR** ,
  XNOR , **NEG** }

  Enumerator containing valid gate types.
- enum class GateValue { **ZERO** = 0 , **ONE** = 1 , **UNDEFINED** }

  Enumerator containing valid values that a gate can have.

Functions

- GateValue get_gate_value (Gate &gate, std::map< size_t, Gate > &nodes)

  Recursively get the value of the gate.

4.1.1 Detailed Description

Structures, enums and functions related to the gate representation.

4.1.2 Function Documentation

4.1.2.1 get_gate_value()

```
GateValue get_gate_value (
            Gate & gate,
            std::map< size_t, Gate > & nodes )
```

Recursively get the value of the gate.

Parameters

in,out	*gate*	Current gate
in,out	*nodes*	Map of all nodes in the circuit

Returns

- GateValue::ZERO or GateValue::ONE on success
- GateValue::UNDEFINED on failure

4.2 gate.hpp

Go to the documentation of this file.

```
00001
00007 #pragma once
00008
00009 #include <vector>
00010 #include <cstddef>
00011 #include <map>
00012
00016 enum class GateType
00017 {
00018     UNDEFINED,
00019     INPUT,
00020     OUTPUT,
00021     AND,
00022     NAND,
00023     OR,
00024     NOR,
00025     XOR,
00026     XNOR,
00027     NEG,
00028 };
00029
00033 enum class GateValue
00034 {
00035     ZERO = 0,
00036     ONE = 1,
00037     UNDEFINED,
00038 };
00039
00043 struct Gate
00044 {
00046     GateType type = GateType::UNDEFINED;
00048     std::vector<size_t> inputs;
00050     GateValue value = GateValue::UNDEFINED;
00051 };
00052
00064 GateValue get_gate_value(Gate& gate, std::map<size_t, Gate>& nodes);
```

4.3 output_writer.hpp File Reference

Header containing function used to write the output to a specific stream.

```
#include "parser_input.hpp"
```

Functions

- bool output_write (std::ostream &stream, const std::vector< InputLine > &input, const std::vector< Output↩ Line > &output)

  Write the processed data to a stream.

4.3.1 Detailed Description

Header containing function used to write the output to a specific stream.

4.3.2 Function Documentation

4.3.2.1 output_write()

```
bool output_write (
            std::ostream & stream,
            const std::vector< InputLine > & input,
            const std::vector< OutputLine > & output )
```

Write the processed data to a stream.

Parameters

in	*stream*	Stream where the data will be saved
in	*input*	Vector containing lines with input states
in	*output*	Vector containing lines with output states

Returns

- true on success
- false on failure

4.4 output_writer.hpp

Go to the documentation of this file.

```
00001
```

```
00007 #pragma once
00008
00009 #include "parser_input.hpp"
00010
00024 bool output_write(std::ostream& stream, const std::vector<InputLine>& input, const
      std::vector<OutputLine>& output);
```

4.5 parser_circuit.hpp File Reference

Functions and structures related to parsing the circuit file.

```
#include <cstddef>
#include <vector>
#include <string>
#include "gate.hpp"
```

Classes

- struct ParsedGate

  Contains information about a specific gate/input (intermediate struct between text and struct Gate)

Functions

- std::vector< ParsedGate > get_circuit (const std::string &filename)

  Extract nodes from the circuit file.

4.5.1 Detailed Description

Functions and structures related to parsing the circuit file.

4.5.2 Function Documentation

4.5.2.1 get_circuit()

```
std::vector< ParsedGate > get_circuit (
            const std::string & filename )
```

Extract nodes from the circuit file.

Parameters

in	*filename*	Path to the file containing the circuit

Returns

Vector of unprocessed logic gates, inputs and outputs

4.6 parser_circuit.hpp

Go to the documentation of this file.

```
00001
00007 #pragma once
00008
00009 #include <cstddef>
00010 #include <vector>
00011 #include <string>
00012 #include "gate.hpp"
00013
00017 struct ParsedGate
00018 {
00020     size_t lineNumber;
00022     GateType type = GateType::UNDEFINED;
00024     std::vector<size_t> nodes;
00025 };
00026
00034 std::vector<ParsedGate> get_circuit(const std::string& filename);
```

4.7 parser_input.hpp File Reference

Functions and structures related to parsing the input file.

```
#include <cstddef>
#include <vector>
#include <string>
```

Classes

- struct Input

  Structure used for reading inputs/writing outputs.

Typedefs

- typedef Input **Output**
- typedef std::vector< Input > **InputLine**
- typedef std::vector< Output > **OutputLine**

Functions

- std::vector< InputLine > get_inputs (const std::string &filename)

  Parse given file.

4.7.1 Detailed Description

Functions and structures related to parsing the input file.

4.7.2 Function Documentation

4.7.2.1 get_inputs()

```
std::vector< InputLine > get_inputs (
            const std::string & filename )
```

Parse given file.

Parameters

in	*filename*	Path to the file containing inputs

Returns

Vector of input lines

4.8 parser_input.hpp

Go to the documentation of this file.

```
00001
00007 #pragma once
00008
00009 #include <cstddef>
00010 #include <vector>
00011 #include <string>
00012
00016 struct Input
00017 {
00019     size_t nodeID;
00021     bool value;
00022 };
00023
00024 typedef Input Output;
00025 typedef std::vector<Input> InputLine;
00026 typedef std::vector<Output> OutputLine;
00027
00035 std::vector<InputLine> get_inputs(const std::string& filename);
```

4.9 simulator.cpp File Reference

Parse command-line arguments, intialize structures and run the simulation.

```
#include <cstdlib>
#include <ios>
#include <iostream>
#include <fstream>
#include <stdexcept>
#include <vector>
#include "gate.hpp"
#include "parser_circuit.hpp"
#include "parser_input.hpp"
#include "simulator.hpp"
#include "output_writer.hpp"
```

Typedefs

- typedef std::vector< size_t > **OutputIDs**

Functions

- void **print_help** (const std::string &program_name, std::ostream &stream)

  Display help message.
- std::pair< std::map< size_t, Gate >, OutputIDs > prepare_simulation (const std::vector< ParsedGate > &parsed_gates)

  Convert initially parsed vector of gates to a map containing these gates.
- OutputLine simulate_circuit (std::map< size_t, Gate > nodes, const InputLine &input, const std::vector< size_t > &outputs)

  Resolve the outputs in the circuit.
- std::string **get_optarg** (int &index, int argc, char *argv[])

  Helper function used for extracting option argument from the argument vector.
- int **simulate** (int argc, char *argv[])

  The main function of the program, where everything happens.

4.9.1 Detailed Description

Parse command-line arguments, intialize structures and run the simulation.

4.9.2 Function Documentation

4.9.2.1 prepare_simulation()

```
std::pair< std::map< size_t, Gate >, OutputIDs > prepare_simulation (
            const std::vector< ParsedGate > & parsed_gates )
```

Convert initially parsed vector of gates to a map containing these gates.

Parameters

in	*parsed_gates*	Vector of unprocessed gates

Returns

- first: Map with the whole circuit
- second: Vector with IDs of output nodes

4.9.2.2 simulate_circuit()

```
OutputLine simulate_circuit (
            std::map< size_t, Gate > nodes,
            const InputLine & input,
            const std::vector< size_t > & outputs )
```

Resolve the outputs in the circuit.

Parameters

in	*nodes*	Map of all logic gates
in	*input*	Values of the input nodes
in	*outputs*	Vector containing IDs of output nodes

Returns

- success: Non-empty vector of output values
- failure: Empty vector

4.10 simulator.hpp File Reference

Header containing the main function of the program.

Functions

- int **simulate** (int argc, char ∗argv[ ])

  The main function of the program, where everything happens.

4.10.1 Detailed Description

Header containing the main function of the program.

4.11 simulator.hpp

Go to the documentation of this file.

```
00001
00007 #pragma once
00008
00012 int simulate(int argc, char* argv[]);
```

Index