My Project

Generated by Doxygen 1.13.2

1 Hierarchical Index 1

1.1 Class Hierarchy 1

2 Class Index 3

2.1 Class List 3

3 File Index 5

3.1 File List 5

4 Class Documentation 7

4.1 Component Class Reference 7

4.2 LogicNetwork Class Reference 7

4.3 LogicNetworkConfigurer Class Reference 8

4.4 Wire Class Reference 9

5 File Documentation 11

5.1 C:/egyetem/nagyHF2/basic_components.h 11

5.2 C:/egyetem/nagyHF2/component.h 11

5.3 C:/egyetem/nagyHF2/logic_network.h 12

5.4 C:/egyetem/nagyHF2/logic_network_configurer.h 12

5.5 C:/egyetem/nagyHF2/memtrace.h 13

5.6 C:/egyetem/nagyHF2/wire.h 15

Index 17

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Component 7
LogicNetwork 7
    LogicNetworkConfigurer 8
Wire 9

Chapter 2

Class Index

2.1 Class List

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

Component 7
LogicNetwork 7
LogicNetworkConfigurer 8
Wire 9

Chapter 3

File Index

3.1 File List

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

C:/egyetem/nagyHF2/basic_components.h 11
C:/egyetem/nagyHF2/component.h 11
C:/egyetem/nagyHF2/logic_network.h 12
C:/egyetem/nagyHF2/logic_network_configurer.h 12
C:/egyetem/nagyHF2/memtrace.h 13
C:/egyetem/nagyHF2/wire.h 15

Chapter 4

Class Documentation

4.1 Component Class Reference

Public Member Functions

- virtual void **update** ()=0
- virtual void **write** (Wire ∗base_address, std::ostream &os=std::cout)=0
- virtual void **debug** ()

Protected Attributes

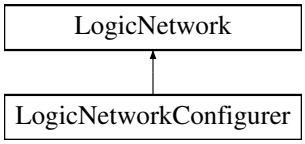
- Wire ∗∗ **inputs**
- Wire ∗ **output**

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

- C:/egyetem/nagyHF2/component.h
- C:/egyetem/nagyHF2/component.cpp

4.2 LogicNetwork Class Reference

Inheritance diagram for LogicNetwork:



Public Member Functions

- **LogicNetwork** (size_t wires_size, std::ostream &os=std::cout)
- void **update** ()
- void **bulk_update** (size_t update_count)
- Wire * **get_wire** (size_t wire_id)
- void **add_component** (Component *component)
- void **debug** ()

Protected Attributes

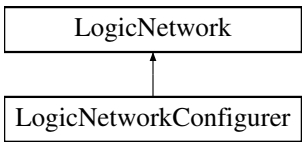
- Wire * **wires**
- size_t **wires_size**
- Component ** **components**
- size_t **components_size**
- std::ostream & **os**

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

- C:/egyetem/nagyHF2/logic_network.h
- C:/egyetem/nagyHF2/logic_network.cpp

4.3 LogicNetworkConfigurer Class Reference

Inheritance diagram for LogicNetworkConfigurer:



Public Member Functions

- **LogicNetworkConfigurer** (size_t wires_size=0, std::ostream &os=std::cout)
- void **read_logic_network** (std::istream &is)
- void **write_logic_network** (std::ostream &os)

Public Member Functions inherited from LogicNetwork

- **LogicNetwork** (size_t wires_size, std::ostream &os=std::cout)
- void **update** ()
- void **bulk_update** (size_t update_count)
- Wire * **get_wire** (size_t wire_id)
- void **add_component** (Component *component)
- void **debug** ()

Additional Inherited Members

Protected Attributes inherited from LogicNetwork

- Wire * **wires**
- size_t **wires_size**
- Component ** **components**
- size_t **components_size**
- std::ostream & **os**

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

- C:/egyetem/nagyHF2/logic_network_configurer.h
- C:/egyetem/nagyHF2/logic_network_configurer.cpp

4.4 Wire Class Reference

Public Member Functions

- void **update** ()
- void **set_signal** (int signal)
- int **get_signal** () const
- void **debug** ()

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

- C:/egyetem/nagyHF2/wire.h
- C:/egyetem/nagyHF2/wire.cpp

Chapter 5

File Documentation

5.1 C:/egyetem/nagyHF2/basic_components.h

```
00001 #ifndef BASIC_COMPONENTS_H
00002 #define BASIC_COMPONENTS_H
00003 #define DEBUG
00004 // alkatalógusból importálja az egyszerű komponenseket
00005 #include "basic_components/AND.h"
00006 #include "basic_components/INP.h"
00007 #include "basic_components/NOT.h"
00008 #include "basic_components/OR.h"
00009 #include "basic_components/PRINT.h"
00010 #include "basic_components/STD_INP.h"
00011 #include "basic_components/XOR.h"
00012 #endif
```

5.2 C:/egyetem/nagyHF2/component.h

```
00001 #ifndef COMPONENT_H
00002 #define DEBUG
00003 #define COMPONENT_H
00004 #include <iostream>
00005
00006 #include "memtrace.h"
00007 #include "wire.h"
00008
00009 // nincs component.cpp?
00010
00011 class Component {
00012     // a gyerekek állítják be, hogy mik legyenek a be és kimenetek
00013     // szerintem így jobb, mintha függvénnyel lenne
00014    protected:
00015     Wire** inputs;
00016     Wire* output;
00017
00018    public:
00019     // NOTE nem konst sehol sem, mert lehessen olyat, aminek van belső állapota
00020     Component() : inputs(nullptr), output(nullptr) {};
00021     // minden leszármazottnak egymástól független, de explicit megkövetelt update függvénye
00022     // ez felel azért, hogy frissítéskor elvégezzék a dolgaikat
00023     virtual void update() = 0;
00024     // ez a fájlba mentés módja, kiírja magát a kapu a megfelelő formátumban
00025     // nem akartam külön osztályokat létrehozni / származtatni őket,
00026     // mert akkor a LogicNetworkConfigurer nem tartalmazhatja a LogicNetwork-ot, mert
00027     // akkor kiírható komponenseket kéne tartalmaznia, nem sima komponenseket, szóval minden
      osztályból csak kettő lenne
00028     // base_address: a kezdőcíme a kábelek tömbjének, hogy ez alapján ki tudják írni hanyas wire-on
      csatlakoznak egy-egy bemeneten
00029     virtual void write(Wire* base_address, std::ostream& os = std::cout) = 0;
00030     virtual ~Component();
00031
00032 #ifdef DEBUG
00033     virtual void debug() {
00034         std::cout « "Default component debug" « std::endl;
00035     };
00036 #endif
00037 };
00038 #endif
```

5.3 C:/egyetem/nagyHF2/logic_network.h

```
00001 #ifndef LOGIC_NETWORK_H
00002 #define LOGIC_NETWORK_H
00003 #include <iostream>
00004
00005 #include "component.h"
00006 #include "memtrace.h"
00007 #include "wire.h"
00008 #define DEBUG
00009 class LogicNetwork {
00010     // lehet privátan kéne? majd átírom, ha igen
00011     // kiíratás miatt
00012     // lehetne mindent függvénybe adni, csak az tök felesleges
00013   protected:
00014     // UML-ben frissíteni, ez wire tömböt tárol, nem Wire*[]-t
00015     // mert asszem végül nem lehet felüldefelni a wire-t?
00016     Wire* wires;
00017     size_t wires_size;
00018     // ide kell a **, mert heterogén kollekció
00019     // ezeket is birtokolja, törli ha törlődik
00020     Component** components;
00021     size_t components_size;
00022     std::ostream& os;
00023
00024     // protecteden, mert nem kell tudnia a felhasználónak, hogy mennyi van a komponensekből, meg hogy
      mik azok
00025     // viszont ki szeretném írni
00026
00027   public:
00028     // létrehozáskor meg kell adni a kábelek számát, hogy rá lehessen kötni a kapukat
00029     LogicNetwork(size_t wires_size, std::ostream& os = std::cout);
00030     // lefrissíti először a wireokat, utána a komponenseket
00031     void update();
00032     // egyszerre több frissítést futtat
00033     void bulk_update(size_t update_count);
00034     // visszaad egy wire-t manuális kötéshez
00035     // ha nincs elég wire, akkor hibát dob
00036     Wire* get_wire(size_t wire_id);
00037     // hozzáad egy komponenst mutatója alapján a hálózathoz
00038     void add_component(Component* component);
00039 #ifdef DEBUG
00040     void debug() {
00041         std::cout « "Logic network" « std::endl
00042                   « "Containts " « wires_size « " wires" « std::endl;
00043         for (size_t i = 0; i < wires_size; i++) {
00044             wires[i].debug();
00045         }
00046         std::cout « "Contains " « components_size « " components" « std::endl;
00047         for (size_t i = 0; i < components_size; i++) {
00048             components[i]->debug();
00049         }
00050     }
00051 #endif
00052     virtual ~LogicNetwork();
00053 };
00054 #endif
```

5.4 C:/egyetem/nagyHF2/logic_network_configurer.h

```
00001 #ifndef LOGIC_NETWORK_CONFIGURER_H
00002 #define LOGIC_NETWORK_CONFIGURER_H
00003
00004 #define DEBUG
00005 #include <iostream>
00006
00007 #include "logic_network.h"
00008 #include "memtrace.h"
00009 class LogicNetworkConfigurer : public LogicNetwork {
00010     // TODO kell mindenféle copy konstruktor, stb
00011
00012   public:
00013     // TODO nem biztos hogy ide kéne a wires_size, meg default argumentumok miatt ilyen sorrendben
00014     // alapvetően 0-ra állítja be a méretet, majd beáll amikor egy txt-ből beolvas
00015     LogicNetworkConfigurer(size_t wires_size = 0, std::ostream& os = std::cout) :
      LogicNetwork(wires_size, os) {}
00016     // TODO olyan függvények, amik ezeket wrappelik és txt-be írnak ki
00017     // beolvassa egy bemeneti streamről a hálózatot
00018     void read_logic_network(std::istream& is);
00019     // kiírja egy kimeneti streamre magát
00020     void write_logic_network(std::ostream& os);
00021 };
```

```
00022
00023 #endif
```

5.5 C:/egyetem/nagyHF2/memtrace.h

```
00001 /*********************************
00002 Memoriaszivargas-detektor
00003 Keszitette: Peregi Tamas, BME IIT, 2011
00004             petamas@iit.bme.hu
00005 Kanari:     Szeberenyi Imre, 2013.,
00006 VS 2012:    Szeberényi Imre, 2015.,
00007 mem_dump:   2016.
00008 inclue-ok:  2017., 2018., 2019., 2021.
00009 *********************************/
00010
00011 #ifndef MEMTRACE_H
00012 #define MEMTRACE_H
00013
00014 #if defined(MEMTRACE)
00015
00016 /*ha definiálva van, akkor a hibakat ebbe a fajlba írja, egyébkent stderr-re*/
00017 /*#define MEMTRACE_ERRFILE MEMTRACE.ERR*/
00018
00019 /*ha definialva van, akkor futas kozben lancolt listat epit. Javasolt a hasznalata*/
00020 #define MEMTRACE_TO_MEMORY
00021
00022 /*ha definialva van, akkor futas kozben fajlba irja a foglalasokat*/
00023 /*ekkor nincs ellenorzes, csak naplozas*/
00024 /*#define MEMTRACE_TO_FILE*/
00025
00026 /*ha definialva van, akkor a megallaskor automatikus riport keszul */
00027 #define MEMTRACE_AUTO
00028
00029 /*ha definialva van, akkor malloc()/calloc()/realloc()/free() kovetve lesz*/
00030 #define MEMTRACE_C
00031
00032 #ifdef MEMTRACE_C
00033     /*ha definialva van, akkor free(NULL) nem okoz hibat*/
00034     #define ALLOW_FREE_NULL
00035 #endif
00036
00037 #ifdef __cplusplus
00038     /*ha definialva van, akkor new/delete/new[]/delete[] kovetve lesz*/
00039     #define MEMTRACE_CPP
00040 #endif
00041
00042 #if defined(__cplusplus) && defined(MEMTRACE_TO_MEMORY)
00043     /*ha definialva van, akkor atexit helyett objektumot hasznal*/
00044     /*ajanlott bekapcsolni*/
00045     #define USE_ATEXIT_OBJECT
00046 #endif
00047
00048 /*****************************************/
00049 /* INNEN NE MODOSITSD                    */
00050 /*****************************************/
00051 #ifdef NO_MEMTRACE_TO_FILE
00052     #undef MEMTRACE_TO_FILE
00053 #endif
00054
00055 #ifdef NO_MEMTRACE_TO_MEMORY
00056     #undef MEMTRACE_TO_MEMORY
00057 #endif
00058
00059 #ifndef MEMTRACE_AUTO
00060     #undef USE_ATEXIT_OBJECT
00061 #endif
00062
00063 #ifdef __cplusplus
00064     #define START_NAMESPACE namespace memtrace {
00065     #define END_NAMESPACE } /*namespace*/
00066     #define TRACEC(func) memtrace::func
00067     #include <new>
00068 #else
00069     #define START_NAMESPACE
00070     #define END_NAMESPACE
00071     #define TRACEC(func) func
00072 #endif
00073
00074 // THROW deklaráció változatai
00075 #if defined(_MSC_VER)
00076   // VS rosszul kezeli az __cplusplus makrot
00077   #if _MSC_VER < 1900
00078     // * nem biztos, hogy jó így *
```

```
00079     #define THROW_BADALLOC
00080     #define THROW_NOTHING
00081   #else
00082     // C++11 vagy újabb
00083     #define THROW_BADALLOC noexcept(false)
00084     #define THROW_NOTHING noexcept
00085   #endif
00086 #else
00087   #if __cplusplus < 201103L
00088     // C++2003 vagy régebbi
00089     #define THROW_BADALLOC throw (std::bad_alloc)
00090     #define THROW_NOTHING throw ()
00091   #else
00092     // C++11 vagy újabb
00093     #define THROW_BADALLOC noexcept(false)
00094     #define THROW_NOTHING noexcept
00095   #endif
00096 #endif
00097
00098 START_NAMESPACE
00099     int allocated_blocks();
00100 END_NAMESPACE
00101
00102 #if defined(MEMTRACE_TO_MEMORY)
00103 START_NAMESPACE
00104         int mem_check(void);
00105 END_NAMESPACE
00106 #endif
00107
00108 #if defined(MEMTRACE_TO_MEMORY) && defined(USE_ATEXIT_OBJECT)
00109 #include <cstdio>
00110 START_NAMESPACE
00111     class atexit_class {
00112         private:
00113             static int counter;
00114             static int err;
00115         public:
00116             atexit_class() {
00117 #if defined(CPORTA) && !defined(CPORTA_NOSETBUF)
00118                 if (counter == 0) {
00119                         setbuf(stdout, 0);
00120                         setbuf(stderr, 0);
00121                 }
00122 #endif
00123             counter++;
00124             }
00125
00126             int check() {
00127                 if(--counter == 0)
00128                     err = mem_check();
00129                 return err;
00130             }
00131
00132             ~atexit_class() {
00133                 check();
00134             }
00135     };
00136
00137 static atexit_class atexit_obj;
00138
00139 END_NAMESPACE
00140 #endif/*MEMTRACE_TO_MEMORY && USE_ATEXIT_OBJECT*/
00141
00142 /*Innentol csak a "normal" include eseten kell, kulonben osszezavarja a mukodest*/
00143 #ifndef FROM_MEMTRACE_CPP
00144 #include <stdlib.h>
00145 #ifdef __cplusplus
00146     #include <iostream>
00147 /* ide gyűjtjük a nemtrace-vel összeakadó headereket, hogy előbb legyenek */
00148
00149     #include <fstream>  // VS 2013 headerjében van deleted definició
00150     #include <sstream>
00151     #include <vector>
00152     #include <list>
00153     #include <map>
00154     #include <algorithm>
00155     #include <functional>
00156     #include <memory>
00157     #include <iomanip>
00158     #include <locale>
00159     #include <typeinfo>
00160     #include <ostream>
00161     #include <stdexcept>
00162     #include <ctime>
00163     #if __cplusplus >= 201103L
00164         #include <iterator>
00165         #include <regex>
```

```
00166     #endif
00167 #endif
00168 #ifdef MEMTRACE_CPP
00169     namespace std {
00170         typedef void (*new_handler)();
00171 }
00172 #endif
00173
00174 #ifdef MEMTRACE_C
00175 START_NAMESPACE
00176     #undef malloc
00177     #define malloc(size) TRACEC(traced_malloc)(size,#size,__LINE__,__FILE__)
00178     void * traced_malloc(size_t size, const char *size_txt, int line, const char * file);
00179
00180     #undef calloc
00181     #define calloc(count,size) TRACEC(traced_calloc)(count, size, #count","#size,__LINE__,__FILE__)
00182     void * traced_calloc(size_t count, size_t size, const char *size_txt, int line, const char *
      file);
00183
00184     #undef free
00185     #define free(p) TRACEC(traced_free)(p, #p,__LINE__,__FILE__)
00186     void traced_free(void * p, const char *size_txt, int line, const char * file);
00187
00188     #undef realloc
00189     #define realloc(old,size) TRACEC(traced_realloc)(old,size,#size,__LINE__,__FILE__)
00190     void * traced_realloc(void * old, size_t size, const char *size_txt, int line, const char * file);
00191
00192     void mem_dump(void const *mem, size_t size, FILE* fp = stdout);
00193
00194
00195 END_NAMESPACE
00196 #endif/*MEMTRACE_C*/
00197
00198 #ifdef MEMTRACE_CPP
00199 START_NAMESPACE
00200     #undef set_new_handler
00201     #define set_new_handler(f) TRACEC(_set_new_handler)(f)
00202     void _set_new_handler(std::new_handler h);
00203
00204     void set_delete_call(int line, const char * file);
00205 END_NAMESPACE
00206
00207 void * operator new(size_t size, int line, const char * file) THROW_BADALLOC;
00208 void * operator new[](size_t size, int line, const char * file) THROW_BADALLOC;
00209 void * operator new(size_t size) THROW_BADALLOC;
00210 void * operator new[](size_t size) THROW_BADALLOC;
00211 void operator delete(void * p)  THROW_NOTHING;
00212 void operator delete[](void * p) THROW_NOTHING;
00213
00214 #if __cplusplus >= 201402L
00215 // sized delete miatt: http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2013/n3536.html
00216 void operator delete(void * p, size_t)  THROW_NOTHING;
00217 void operator delete[](void * p, size_t) THROW_NOTHING;
00218 #endif
00219
00220 /* Visual C++ 2012 miatt kell, mert háklis, hogy nincs megfelelő delete, bár senki sem használja */
00221 void operator delete(void *p, int, const char *) THROW_NOTHING;
00222 void operator delete[](void *p, int, const char *) THROW_NOTHING;
00223
00224
00225 #define new new(__LINE__, __FILE__)
00226 #define delete memtrace::set_delete_call(__LINE__, __FILE__),delete
00227
00228 #ifdef CPORTA
00229 #define system(...)  // system(__VA_ARGS__)
00230 #endif
00231
00232 #endif /*MEMTRACE_CPP*/
00233
00234 #endif /*FROM_MEMTRACE_CPP*/
00235 #else
00236 #pragma message ( "MEMTRACE NOT DEFINED" )
00237 #endif /*MEMTRACE*/
00238
00239 #endif /*MEMTRACE_H*/
```

5.6 C:/egyetem/nagyHF2/wire.h

```
00001 #ifndef WIRE_H
00002 #define WIRE_H
00003 #include <iostream>
00004
00005 #include "memtrace.h"
```

```
00006 #define DEBUG
00007 class Wire {
00008     int input;
00009     int output;
00010 
00011   public:
00012     Wire() : input(0), output(0) {}
00013     // elejéről a végére rakja a jelet
00014     void update();
00015     // beállítja a jelet a bemenetén
00016     void set_signal(int signal);
00017     // visszaadja a kimenetén lévő jelet
00018     int get_signal() const {
00019         return output;
00020     }
00021 #ifdef DEBUG
00022     void debug() {
00023         std::cout « "wire: " « (void*)this « "\n\tinput:  " « input « "\n\toutput: " « output «
      std::endl;
00024     }
00025 #endif
00026 };
00027 
00028 #endif
```

Index

Component, 7

LogicNetwork, 7
LogicNetworkConfigurer, 8

Wire, 9