Reference Manual

Generated by Doxygen 1.5.8

Thu Sep 17 00:08:12 2009

Contents

1 Guidance for STAMP program 1

1.1 Introduction 1

1.2 How to Compile and Install 1

1.3 Hint 2

1.4 How to Expand the Program 2

1.5 Contact 2

2 Todo List 3

3 Class Index 5

3.1 Class Hierarchy 5

4 Class Index 7

4.1 Class List 7

5 File Index 9

5.1 File List 9

6 Class Documentation 11

6.1 Capacitor Class Reference 11

6.1.1 Detailed Description 11

6.1.2 Constructor & Destructor Documentation 12

6.1.2.1 Capacitor 12

6.1.3 Member Function Documentation 12

6.1.3.1 stamp 12

6.2 Device Class Reference 13

6.2.1 Detailed Description 14

6.2.2 Member Function Documentation 14

6.2.2.1 name 14

6.2.2.2 nnode 14

6.2.2.3 pnode 15
6.2.2.4 setName 15
6.2.2.5 setNnode 15
6.2.2.6 setPnode 15
6.2.2.7 setValue 15
6.2.2.8 stamp 16
6.2.2.9 value 16
6.3 Inductor Class Reference 17
6.3.1 Detailed Description 17
6.3.2 Constructor & Destructor Documentation 17
6.3.2.1 Inductor 18
6.3.3 Member Function Documentation 18
6.3.3.1 auxNode 18
6.3.3.2 setAux 18
6.3.3.3 stamp 18
6.4 Isrc Class Reference 19
6.4.1 Detailed Description 19
6.4.2 Constructor & Destructor Documentation 19
6.4.2.1 Isrc 19
6.4.3 Member Function Documentation 19
6.4.3.1 stamp 20
6.5 Mat< T > Class Template Reference 21
6.5.1 Detailed Description 21
6.5.2 Constructor & Destructor Documentation 21
6.5.2.1 Mat 22
6.5.2.2 ∼Mat 22
6.5.3 Member Function Documentation 22
6.5.3.1 column 22
6.5.3.2 row 22
6.6 Mutual Class Reference 23
6.6.1 Detailed Description 24
6.6.2 Constructor & Destructor Documentation 24
6.6.2.1 Mutual 24
6.6.3 Member Function Documentation 24
6.6.3.1 auxNegNode 24
6.6.3.2 auxPosNode 24

6.6.3.3 ind1 24
6.6.3.4 ind2 25
6.6.3.5 setAuxNeg 25
6.6.3.6 setAuxPos 25
6.6.3.7 setInd1 25
6.6.3.8 setInd2 25
6.6.3.9 stamp 26
6.7 Resistor Class Reference 27
6.7.1 Detailed Description 27
6.7.2 Constructor & Destructor Documentation 27
6.7.2.1 Resistor 27
6.7.3 Member Function Documentation 27
6.7.3.1 stamp 28
6.8 Stamp Class Reference 29
6.8.1 Detailed Description 29
6.8.2 Constructor & Destructor Documentation 29
6.8.2.1 ∼Stamp 29
6.8.3 Member Function Documentation 30
6.8.3.1 output 30
6.8.3.2 parse 30
6.8.3.3 setup 30
6.9 Vsrc Class Reference 32
6.9.1 Detailed Description 32
6.9.2 Constructor & Destructor Documentation 32
6.9.2.1 Vsrc 33
6.9.3 Member Function Documentation 33
6.9.3.1 auxNode 33
6.9.3.2 setAux 33
6.9.3.3 stamp 33

7 File Documentation 35
7.1 /home/yangfan/project1/project1/stamp-1.0/src/cap.cpp File Reference 35
7.1.1 Detailed Description 35
7.2 /home/yangfan/project1/project1/stamp-1.0/src/cap.h File Reference 36
7.2.1 Detailed Description 36
7.3 /home/yangfan/project1/project1/stamp-1.0/src/device.h File Reference 37
7.3.1 Detailed Description 37

7.4 /home/yangfan/project1/project1/stamp-1.0/src/ind.cpp File Reference 38
7.4.1 Detailed Description 38
7.5 /home/yangfan/project1/project1/stamp-1.0/src/ind.h File Reference 39
7.5.1 Detailed Description 39
7.6 /home/yangfan/project1/project1/stamp-1.0/src/isrc.cpp File Reference 40
7.6.1 Detailed Description 40
7.7 /home/yangfan/project1/project1/stamp-1.0/src/isrc.h File Reference 41
7.7.1 Detailed Description 41
7.8 /home/yangfan/project1/project1/stamp-1.0/src/main.cpp File Reference 42
7.8.1 Detailed Description 42
7.8.2 Function Documentation 42
7.8.2.1 main 42
7.9 /home/yangfan/project1/project1/stamp-1.0/src/mat.h File Reference 43
7.9.1 Detailed Description 43
7.10 /home/yangfan/project1/project1/stamp-1.0/src/mut.cpp File Reference 44
7.10.1 Detailed Description 44
7.11 /home/yangfan/project1/project1/stamp-1.0/src/mut.h File Reference 45
7.11.1 Detailed Description 45
7.12 /home/yangfan/project1/project1/stamp-1.0/src/res.cpp File Reference 46
7.12.1 Detailed Description 46
7.13 /home/yangfan/project1/project1/stamp-1.0/src/res.h File Reference 47
7.13.1 Detailed Description 47
7.14 /home/yangfan/project1/project1/stamp-1.0/src/stamp.cpp File Reference 48
7.14.1 Detailed Description 48
7.15 /home/yangfan/project1/project1/stamp-1.0/src/stamp.h File Reference 49
7.15.1 Detailed Description 49
7.16 /home/yangfan/project1/project1/stamp-1.0/src/util.cpp File Reference 50
7.16.1 Detailed Description 50
7.16.2 Function Documentation 50
7.16.2.1 capitalize 50
7.16.2.2 to_double 51
7.16.2.3 tokenizer 51
7.17 /home/yangfan/project1/project1/stamp-1.0/src/util.h File Reference 52
7.17.1 Detailed Description 52
7.17.2 Function Documentation 52
7.17.2.1 capitalize 52

7.17.2.2 to_double 53
7.17.2.3 tokenizer 53
7.18 /home/yangfan/project1/project1/stamp-1.0/src/vsrc.cpp File Reference 54
7.18.1 Detailed Description 54
7.19 /home/yangfan/project1/project1/stamp-1.0/src/vsrc.h File Reference 55
7.19.1 Detailed Description 55

Chapter 1

Guidance for STAMP program

1.1 Introduction

This docmument aims at helping you to build a STAMP program. This doc is embbed in a skeleton program show some main steps recommended in builiding such program. The recommended stamp flow contains

1. Parsing the SPICE netlist.
2. Stamping the elements from each device to the system matrices.
3. Output the matrix to a binary file.

Please note that the skeleton program is very simple but intuitive. Just follow the todo items to fill up and expand the functions, and you will have your own STAMP program. Good luck.

1.2 How to Compile and Install

This program is developed under Linux and maintianed by Autotools. However, it can also be compiled in win32 enviroment. To build this program in Linux, type the following commands in the shell:

```
./configure --enable-debug=yes
 make
```

Your can drop the *–enable-debug=yes* argument if you don't want to debug the program After building the program, you will have a binary executable file in *src*. Then You can run it by typing

```
./stamp netlist.sp mat.out
```

For windows user, please follow the following instructions:

1. Download a VC++6.0 with SP6 from `http://10.64.130.17:82/` (search with keyword VC++)

2. Go to directory win32/stamp, open the VC++ project file stamp.dsw
3. Build the project in VC++6.0

1.3 Hint

- Make good use of this document. Especially, check the TODO list.
- Discuss with your colleagues. But *don't copy!*
- Read the SPICE manual or some C++ references.

1.4 How to Expand the Program

For Linux user, when you add some new source or header files to this project, you have to update the *Makefile.am* and reconfig the Autotools. First, you have to add the names of the source files to *Makefile.am* in *src* directory and then type the following commands in the shell

```
automake --add-missing
autoreconf
```

This will generate a new *configure* file in the project root. Then using the build commands introduced above to re-build the program. For windows user, just use VC++ to add the source/header files to your VC++ project.

1.5 Contact

If you're confused or have any problems. Please contact me. My email is `yangfan@fudan.edu.cn`. You can also come to Room 319 at working hours to find me. Thanks.

Chapter 2

Todo List

Member Capacitor::stamp(Matrix &C, Matrix &G, Matrix &B) You have to fills in each stamp function.

Member Inductor::stamp(Matrix &C, Matrix &G, Matrix &B) You have to fills in each stamp function.

Member Isrc::stamp(Matrix &C, Matrix &G, Matrix &B) You have to fills in each stamp function.

Member Mat::Mat(int nrow, int ncol) Allocate the memory with given size of the matrix

Member Mat::∼Mat() release the allocated memory

Member Mutual::stamp(Matrix &C, Matrix &G, Matrix &B) You have to fills in each stamp function.

Member Resistor::stamp(Matrix &C, Matrix &G, Matrix &B) You have to fills in each stamp function.

Member Stamp::output(char ∗filename) Please fill in this function.

Member Stamp::parse(char ∗filename) Currently, only limited devices cards for SPICE are considered. You can Further develop this fucntion to accomodate more SPICE elements or cards such as subcircuit or controlled sources.

Member Stamp::setup() = Some devices such as mutual inductance depends on other devices. You need to take care of this situation.

- We also have to enumerate the probe list to build the L^T matrix.
- SPICE file support multi-line command with a '+' at the beginning of the line. Please consider how to support this.

Member Vsrc::stamp(Matrix &C, Matrix &G, Matrix &B) You have to fills in each stamp function.

Chapter 3

Class Index

3.1 Class Hierarchy

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

- Device 13
  - Capacitor 11
  - Inductor 17
  - Isrc 19
  - Mutual 23
  - Resistor 27
  - Vsrc 32
- Mat< T > 21
- Stamp 29

Chapter 4

Class Index

4.1 Class List

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

Capacitor (The capacitor) 11
Device (The base class for various devices) 13
Inductor (The inductor) 17
Isrc (The current source) 19
Mat< T > (Matrix class) 21
Mutual (The mutual inductance between two inductors) 23
Resistor (The resistor) 27
Stamp (Stamp engine) 29
Vsrc (The voltage source) 32

Chapter 5

File Index

5.1 File List

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

/home/yangfan/project1/project1/stamp-1.0/src/cap.cpp (Implementation of Capacitor) 35
/home/yangfan/project1/project1/stamp-1.0/src/cap.h (Header of the Capacitor class) 36
/home/yangfan/project1/project1/stamp-1.0/src/device.h (The header of Device class) 37
/home/yangfan/project1/project1/stamp-1.0/src/ind.cpp (Implementation of inductor) 38
/home/yangfan/project1/project1/stamp-1.0/src/ind.h (Header of the Inductor class) 39
/home/yangfan/project1/project1/stamp-1.0/src/isrc.cpp (Implementation of current source) . . . 40
/home/yangfan/project1/project1/stamp-1.0/src/isrc.h (Header of the Current source class) . . . 41
/home/yangfan/project1/project1/stamp-1.0/src/main.cpp (This is the main function of STAMP) 42
/home/yangfan/project1/project1/stamp-1.0/src/mat.h (Header of the Matrix class) 43
/home/yangfan/project1/project1/stamp-1.0/src/mut.cpp (Implementation of mutual inductance) 44
/home/yangfan/project1/project1/stamp-1.0/src/mut.h (Header of the mutual inductance) 45
/home/yangfan/project1/project1/stamp-1.0/src/res.cpp (Implementation of Resistor) 46
/home/yangfan/project1/project1/stamp-1.0/src/res.h (Header of the Resistor class) 47
/home/yangfan/project1/project1/stamp-1.0/src/stamp.cpp (Implementation of the Stamp class) . 48
/home/yangfan/project1/project1/stamp-1.0/src/stamp.h (Header file for Stamp class) 49
/home/yangfan/project1/project1/stamp-1.0/src/util.cpp (Implementation of the utility functions) 50
/home/yangfan/project1/project1/stamp-1.0/src/util.h (Header of the utility funtions) 52
/home/yangfan/project1/project1/stamp-1.0/src/vsrc.cpp (Implementation of voltage source) . . 54
/home/yangfan/project1/project1/stamp-1.0/src/vsrc.h (Header of voltage source) 55

Chapter 6

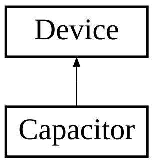
Class Documentation

6.1 Capacitor Class Reference

The capacitor.

`#include <cap.h>`

Inheritance diagram for Capacitor::



Public Member Functions

- Capacitor (const string &name)

  Constructor.

- virtual void stamp (Matrix &C, Matrix &G, Matrix &B)

  stamping function of the capacitor

- virtual ~Capacitor ()

  Destructor.

6.1.1 Detailed Description

The capacitor.

A Capacitor instance corrsponds to the "c1 n1 n2 3p" card in the SPICE netlist It records the information of the capacitor.

6.1.2 Constructor & Destructor Documentation

6.1.2.1 Capacitor::Capacitor (const string & *name*) `[inline]`

Constructor.

Parameters:

name name of the capacitor

6.1.3 Member Function Documentation

6.1.3.1 void Capacitor::stamp (Matrix & *C*, Matrix & *G*, Matrix & *B*) `[virtual]`

stamping function of the capacitor

Parameters:

C system matrix C

G system matrix G

B system matrix B

Todo

You have to fills in each stamp function.

Implements Device.

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

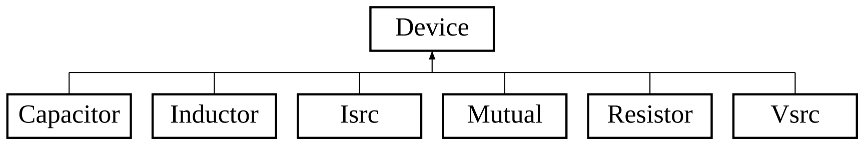
- /home/yangfan/project1/project1/stamp-1.0/src/cap.h
- /home/yangfan/project1/project1/stamp-1.0/src/cap.cpp

6.2 Device Class Reference

The base class for various devices.

`#include <device.h>`

Inheritance diagram for Device::



Public Member Functions

- Device (const string &name)

  Constructor.

- virtual void stamp (Matrix &C, Matrix &G, Matrix &B)=0

  pure virtual fucntion for stamping

- virtual ~Device ()

  destructor

- void setPnode (int p)

  Set the positive node number.

- int pnode () const

  Get the positive node number.

- void setNnode (int p)

  Set the negative node number.

- int nnode () const

  Get the negative node number.

- void setValue (double d)

  Set the value of this device.

- double value () const

  Get the value of this device.

- void setName (const string &n)

  Set the name of this device.

- string name () const

  Get the name of this device.

Protected Attributes

- string _name

  device name

- int _pnode

  positive node number

- int _nnode

  negative node number

- double _value

  value

6.2.1 Detailed Description

The base class for various devices.

This is an abstract base class for all the devices in the circuit. For a device it must have a name, have some pins and some values. Most important of all, it must have a *stamp* function which shows its contribution to the system matrix. Different devices such as resistors and capacitors have different stamping functions. So they will all overwrites the *stamp* pure virtual function here.

If you want to add some new devices to this engine, please inherit from Device. And make sure to implement the *stamp* function.

6.2.2 Member Function Documentation

6.2.2.1 string Device::name () const `[inline]`

Get the name of this device.

Returns:

name of this device

6.2.2.2 int Device::nnode () const `[inline]`

Get the negative node number.

Returns:

negative node number of this device

6.2.2.3 int Device::pnode () const `[inline]`

Get the positive node number.

Returns:

positive node number of this device

6.2.2.4 void Device::setName (const string & *n*) `[inline]`

Set the name of this device.

Parameters:

n device name

6.2.2.5 void Device::setNnode (int *p*) `[inline]`

Set the negative node number.

Parameters:

p node number

6.2.2.6 void Device::setPnode (int *p*) `[inline]`

Set the positive node number.

Parameters:

p node number

6.2.2.7 void Device::setValue (double *d*) `[inline]`

Set the value of this device.

Parameters:

d value

The value could be capacitance, conductance or mutual inductance coefficient and etc. Store what you want.

6.2.2.8 virtual void Device::stamp (Matrix & ***C***, Matrix & ***G***, Matrix & ***B***) `[pure virtual]`

pure virtual fucntion for stamping

Parameters:

C system matrix C

G system matrix G

B system matrix B

Implemented in Capacitor, Inductor, Isrc, Mutual, Resistor, and Vsrc.

6.2.2.9 double Device::value () const `[inline]`

Get the value of this device.

Returns:

value

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

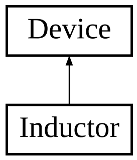
- /home/yangfan/project1/project1/stamp-1.0/src/device.h

6.3 Inductor Class Reference

The inductor.

`#include <ind.h>`

Inheritance diagram for Inductor::



Public Member Functions

- Inductor (const string &name)

  Constructor.

- virtual void stamp (Matrix &C, Matrix &G, Matrix &B)

  stamping function of the inductor

- virtual ∼Inductor ()

  Destructor.

- int auxNode () const

  Get auxiliary node.

- void setAux (int s)

  set auxiliary node number

6.3.1 Detailed Description

The inductor.

A Inductor instance corrsponds to the "l1 n1 n2 3n" card in the SPICE netlist It records the information of the inductor. In MNA formualtion, an inductor will introduce a new state variable to the equation, i.e. its current. So we added an auxiliary node *_aux_node* to store this informatin.

6.3.2 Constructor & Destructor Documentation

6.3.2.1 Inductor::Inductor (const string & *name*) `[inline]`

Constructor.

Parameters:

name name of the inductor

6.3.3 Member Function Documentation

6.3.3.1 int Inductor::auxNode () const `[inline]`

Get auxiliary node.

Returns:

auxiliary node (current through the inductor)

6.3.3.2 void Inductor::setAux (int *s*) `[inline]`

set auxiliary node number

Parameters:

s auxiliary node number

6.3.3.3 void Inductor::stamp (Matrix & *C*, Matrix & *G*, Matrix & *B*) `[virtual]`

stamping function of the inductor

Parameters:

C system matrix C

G system matrix G

B system matrix B

Todo

You have to fills in each stamp function.

Implements Device.

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

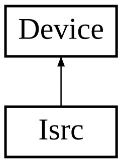
- /home/yangfan/project1/project1/stamp-1.0/src/ind.h
- /home/yangfan/project1/project1/stamp-1.0/src/ind.cpp

6.4 Isrc Class Reference

The current source.

`#include <isrc.h>`

Inheritance diagram for Isrc::



Public Member Functions

- Isrc (const string &name)

  Constructor.
- virtual void stamp (Matrix &C, Matrix &G, Matrix &B)

  stamping function of the current source
- virtual ∼Isrc ()

  Destructor.

6.4.1 Detailed Description

The current source.

An Isrc instance corrsponds to the "i1 n1 n2 1" card in the SPICE netlist It records the information of the current source.

6.4.2 Constructor & Destructor Documentation

6.4.2.1 Isrc::Isrc (const string & *name*) `[inline]`

Constructor.

Parameters:

name name of the current source

6.4.3 Member Function Documentation

6.4.3.1 void Isrc::stamp (Matrix & *C*, Matrix & *G*, Matrix & *B*) `[virtual]`

stamping function of the current source

Parameters:

- ***C*** system matrix C
- ***G*** system matrix G
- ***B*** system matrix B

Todo

You have to fills in each stamp function.

Implements Device.

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

- /home/yangfan/project1/project1/stamp-1.0/src/isrc.h
- /home/yangfan/project1/project1/stamp-1.0/src/isrc.cpp

6.5 Mat< T > Class Template Reference

the matrix class

`#include <mat.h>`

Public Member Functions

- Mat (int nrow, int ncol)

  Constructor.

- ∼Mat ()

  Destructor.

- int row () const

  return rows

- int column () const

  return columns

6.5.1 Detailed Description

template<class T> class Mat< T >

the matrix class

This is an empty class. You have to design you own Matrix class by adding necessary data and public functions.

To facilliate the stamping process, the matrix class needs to satisfy the following condition:

- Dynamic allocation of the memory according to the size of the circuit.
- Expandable. Each time we stamp an element, the entries of the matrix may grow.
- Flexible and efficient. Most of the entries in the matrix would be zero and we don't need to store them.

My advice is: Using orthogonal list data structure.

Currently, this class is designed as a C++ template. If you're not familliar with the concept of template. Just define a non-template class and do the same thing.

6.5.2 Constructor & Destructor Documentation

6.5.2.1 template< class T > Mat< T >::Mat (int *nrow*, int *ncol*) `[inline]`

Constructor.

Parameters:

> ***nrow*** rows
> ***ncol*** columns

Todo

> Allocate the memory with given size of the matrix

6.5.2.2 template< class T > Mat< T >::~Mat () `[inline]`

Destructor.

Todo

> release the allocated memory

6.5.3 Member Function Documentation

6.5.3.1 template< class T > int Mat< T >::column () const `[inline]`

return columns

Returns:

> columns

6.5.3.2 template< class T > int Mat< T >::row () const `[inline]`

return rows

Returns:

> rows

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

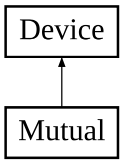
- /home/yangfan/project1/project1/stamp-1.0/src/mat.h

6.6 Mutual Class Reference

The mutual inductance between two inductors.

`#include <mut.h>`

Inheritance diagram for Mutual::



Public Member Functions

- Mutual (const string &name)

  Constructor.

- virtual void stamp (Matrix &C, Matrix &G, Matrix &B)

  stamping function of the mutual inductance

- virtual ∼Mutual ()

  Destructor.

- int auxPosNode () const

  return the auxiliary current node of L1

- void setAuxPos (int s)

  set the auxiliary current node of L1

- int auxNegNode () const

  return the auxiliary current node of L2

- void setAuxNeg (int s)

  set the auxiliary current node of L2

- string ind1 () const

  return the name of L1

- void setInd1 (const string &l)

  set the name of L1

- string ind2 () const

  return the name of L2

- void setInd2 (const string &l)

  set the name of L2

6.6.1 Detailed Description

The mutual inductance between two inductors.

An Mutual instance corrsponds to the "k1 l1 l2 0.5" card in the SPICE netlist It records the information of the mutual inductance. Differnt from other devices in the SPICE netlist, the mutual inductance doesn't have the concept of pins. Instead, it records the name of two inductors which will also appear in the the netlist. However, it is not sure that the two inductors will appear before or after the mutual indectance card. so we have to record the name of these two inductors. Additionally, we also have to record currents through the inductors as auxiliary vairiables.

6.6.2 Constructor & Destructor Documentation

6.6.2.1 Mutual::Mutual (const string & *name*) `[inline]`

Constructor.

Parameters:

name name of the mutual inductance

6.6.3 Member Function Documentation

6.6.3.1 int Mutual::auxNegNode () const `[inline]`

return the auxiliary current node of L2

Returns:

auxiliary current node of L2

6.6.3.2 int Mutual::auxPosNode () const `[inline]`

return the auxiliary current node of L1

Returns:

auxiliary current node of L1

6.6.3.3 string Mutual::ind1 () const `[inline]`

return the name of L1

Returns:

name of inductor L1

6.6.3.4 string Mutual::ind2 () const `[inline]`

return the name of L2

Returns:

name of inductor L2

6.6.3.5 void Mutual::setAuxNeg (int *s*) `[inline]`

set the auxiliary current node of L2

Parameters:

s auxiliary current node of L2

6.6.3.6 void Mutual::setAuxPos (int *s*) `[inline]`

set the auxiliary current node of L1

Parameters:

s auxiliary current node of L1

6.6.3.7 void Mutual::setInd1 (const string & *l*) `[inline]`

set the name of L1

Parameters:

l name of inductor L1

6.6.3.8 void Mutual::setInd2 (const string & *l*) `[inline]`

set the name of L2

Parameters:

l name of inductor L2

6.6.3.9 void Mutual::stamp (Matrix & *C*, Matrix & *G*, Matrix & *B*) `[virtual]`

stamping function of the mutual inductance

Parameters:

- ***C*** system matrix C
- ***G*** system matrix G
- ***B*** system matrix B

Todo

You have to fills in each stamp function.

Implements Device.

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

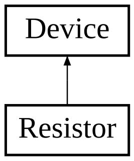
- /home/yangfan/project1/project1/stamp-1.0/src/mut.h
- /home/yangfan/project1/project1/stamp-1.0/src/mut.cpp

6.7 Resistor Class Reference

The resistor.

`#include <res.h>`

Inheritance diagram for Resistor::



Public Member Functions

- Resistor (const string &name)

  Constructor.

- virtual void stamp (Matrix &C, Matrix &G, Matrix &B)

  stamping function of the resistor

- virtual ~Resistor ()

  Destructor.

6.7.1 Detailed Description

The resistor.

A Resisitor instance corrsponds to the "r1 n1 n2 3k" card in the SPICE netlist It records the information of the resisitor.

6.7.2 Constructor & Destructor Documentation

6.7.2.1 Resistor::Resistor (const string & *name*) `[inline]`

Constructor.

Parameters:

name name of the resistor

6.7.3 Member Function Documentation

6.7.3.1 void Resistor::stamp (Matrix & *C*, Matrix & *G*, Matrix & *B*) `[virtual]`

stamping function of the resistor

Parameters:

C system matrix C

G system matrix G

B system matrix B

Todo

You have to fills in each stamp function.

Implements Device.

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

- /home/yangfan/project1/project1/stamp-1.0/src/res.h
- /home/yangfan/project1/project1/stamp-1.0/src/res.cpp

6.8 Stamp Class Reference

Stamp engine.

`#include <stamp.h>`

Public Types

- typedef std::vector< Device ∗ > **devList**
- typedef std::map< std::string, int > **nodeList**

Public Member Functions

- Stamp ()

  Constructor.

- ∼Stamp ()

  Destructor.

- void parse (char ∗filename)

  Parsing the SPICE netlist.

- void output (char ∗filename)

  Output the system matrix to disk.

- void setup ()

  Initialize the system matrix and carry out the stamping.

6.8.1 Detailed Description

Stamp engine.

The class Stamp is actually the syamp engine. It contains necessary data structures for completing the work.

6.8.2 Constructor & Destructor Documentation

6.8.2.1 Stamp::∼Stamp () `[inline]`

Destructor.

It destroys the system matrix and release the allocated memory.

6.8.3 Member Function Documentation

6.8.3.1 void Stamp::output (char ∗ *filename*)

Output the system matrix to disk.

Parameters:

filename output file name

This function will write the system matrix to the disk. Binary file format is preferred over the ASCII one. Sparse matrix structure is preferred over the dense one.

Todo

Please fill in this function.

6.8.3.2 void Stamp::parse (char ∗ *filename*)

Parsing the SPICE netlist.

Parameters:

filename SPICE filename

This is the first step of STAMP. It reads in the SPICE file and extracts device and node information from it. The devices and nodes are stored in *_dev_list* and *_node_list* for later usage. The parser reads in the SPICE netlist line by line. It breaks each line into string tokens according to the delimiters, and then processes each token case by case. Notice the using of *tokenizer* and *captilizer*. These two functions break the line and convert them to upper case. (SPICE file are not case-sensitive)

Todo

Currently, only limited devices cards for SPICE are considered. You can Further develop this fucntion to accomodate more SPICE elements or cards such as subcircuit or controlled sources.

6.8.3.3 void Stamp::setup ()

Initialize the system matrix and carry out the stamping.

This function is most important one because it setups up the liear system

$$\left\{ \begin{array}{ccc} C\dot{x} + Gx & = & Bu \\ y & = & L^T x \end{array} \right.$$

and fills in the elements in the system matrices.

The matrix class *Matrix* needs to be designed by you. Sparse matrix structure is preferred than the dense one since the matrix will be very large while most of its elemets would be zero.

In the skeleton program, the devices are enumerated and they contributes to the system matrices by calling thier own stamping function.

Todo

= Some devices such as mutual inductance depends on other devices. You need to take care of this situation.

- We also have to enumerate the probe list to build the L^T matrix.
- SPICE file support multi-line command with a '+' at the beginning of the line. Please consider how to support this.

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

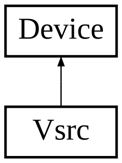
- /home/yangfan/project1/project1/stamp-1.0/src/stamp.h
- /home/yangfan/project1/project1/stamp-1.0/src/stamp.cpp

6.9 Vsrc Class Reference

The voltage source.

`#include <vsrc.h>`

Inheritance diagram for Vsrc::



Public Member Functions

- Vsrc (const string &name)

  Constructor.

- virtual void stamp (Matrix &C, Matrix &G, Matrix &B)

  stamping function of the voltage source

- virtual ~Vsrc ()

  Destructor.

- int auxNode () const

  Get auxiliary node.

- void setAux (int s)

  set auxiliary node number

6.9.1 Detailed Description

The voltage source.

A Vsrc instance corrsponds to the "v1 n1 n2 1" card in the SPICE netlist It records the information of the voltage source. Like the inductor, a voltage source will also introduce a new variable to the equation, we have to record its current.

6.9.2 Constructor & Destructor Documentation

6.9.2.1 Vsrc::Vsrc (const string & *name*) `[inline]`

Constructor.

Parameters:

name name of the voltage source

6.9.3 Member Function Documentation

6.9.3.1 int Vsrc::auxNode () const `[inline]`

Get auxiliary node.

Returns:

auxiliary node (current through the inductor)

6.9.3.2 void Vsrc::setAux (int *s*) `[inline]`

set auxiliary node number

Parameters:

s auxiliary node number

6.9.3.3 void Vsrc::stamp (Matrix & *C*, Matrix & *G*, Matrix & *B*) `[virtual]`

stamping function of the voltage source

Parameters:

C system matrix C

G system matrix G

B system matrix B

Todo

You have to fills in each stamp function.

Implements Device.

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

- /home/yangfan/project1/project1/stamp-1.0/src/vsrc.h
- /home/yangfan/project1/project1/stamp-1.0/src/vsrc.cpp

Chapter 7

File Documentation

7.1 /home/yangfan/project1/project1/stamp-1.0/src/cap.cpp File Reference

implementation of Capacitor

```
#include "cap.h"
```

7.1.1 Detailed Description

implementation of Capacitor

Author:

Yinghai

Date:

Fri Sep 12 14:25:35 2008

7.2 /home/yangfan/project1/project1/stamp-1.0/src/cap.h File Reference

header of the Capacitor class

```
#include "device.h"
```

Classes

- class Capacitor

  The capacitor.

7.2.1 Detailed Description

header of the Capacitor class

Author:

Yinghai

Date:

Fri Sep 12 14:21:48 2008

7.3 /home/yangfan/project1/project1/stamp-1.0/src/device.h File Reference

The header of Device class.

```
#include <string>
#include <iostream>
#include "mat.h"
```

Classes

- class Device

  The base class for various devices.

7.3.1 Detailed Description

The header of Device class.

Author:

Yinghai

Date:

Fri Sep 12 13:37:14 2008

7.4 /home/yangfan/project1/project1/stamp-1.0/src/ind.cpp File Reference

implementation of inductor

```
#include "ind.h"
```

7.4.1 Detailed Description

implementation of inductor

Author:

Yinghai

Date:

Fri Sep 12 14:32:59 2008

7.5 /home/yangfan/project1/project1/stamp-1.0/src/ind.h File Reference

header of the Inductor class

```
#include "device.h"
```

Classes

- class Inductor

  The inductor.

7.5.1 Detailed Description

header of the Inductor class

Author:

Yinghai

Date:

Fri Sep 12 14:28:21 2008

7.6 /home/yangfan/project1/project1/stamp-1.0/src/isrc.cpp File Reference

implementation of current source

```
#include "isrc.h"
```

7.6.1 Detailed Description

implementation of current source

Author:

Yinghai

Date:

Fri Sep 12 14:45:16 2008

7.7 /home/yangfan/project1/project1/stamp-1.0/src/isrc.h File Reference

header of the Current source class

`#include "device.h"`

Classes

- class Isrc
  The current source.

7.7.1 Detailed Description

header of the Current source class

Author:

Yinghai

Date:

Fri Sep 12 14:35:29 2008

7.8 /home/yangfan/project1/project1/stamp-1.0/src/main.cpp File Reference

This is the main function of STAMP.

`#include "stamp.h"`

`#include <iostream>`

Functions

- int main (int argc, char ∗argv[ ])

  entrance of the program

7.8.1 Detailed Description

This is the main function of STAMP.

Author:

Yinghai

Date:

Fri Sep 12 09:57:35 2008

7.8.2 Function Documentation

7.8.2.1 int main (int *argc*, char ∗ *argv*[ ])

entrance of the program

phase 1: parsing the netlist

phase 2: stamping

phase 3: output

7.9 /home/yangfan/project1/project1/stamp-1.0/src/mat.h File Reference

Header of the Matrix class.

`#include <cstring>`

Classes

- class Mat< T >
  the matrix class

Typedefs

- typedef Mat< double > **Matrix**

7.9.1 Detailed Description

Header of the Matrix class.

Author:

Date:

Fri Sep 12 14:06:47 2008

7.10 /home/yangfan/project1/project1/stamp-1.0/src/mut.cpp File Reference

implementation of mutual inductance

```
#include "mut.h"
```

7.10.1 Detailed Description

implementation of mutual inductance

Author:

Yinghai

Date:

Fri Sep 12 15:06:23 2008

7.11 /home/yangfan/project1/project1/stamp-1.0/src/mut.h File Reference

Header of the mutual inductance.

```
#include "device.h"
```

Classes

- class Mutual

  The mutual inductance between two inductors.

7.11.1 Detailed Description

Header of the mutual inductance.

Author:

Yinghai

Date:

Fri Sep 12 14:54:59 2008

7.12 /home/yangfan/project1/project1/stamp-1.0/src/res.cpp File Reference

implementation of Resistor

```
#include "res.h"
```

7.12.1 Detailed Description

implementation of Resistor

Author:

Yinghai

Date:

Fri Sep 12 14:21:15 2008

7.13 /home/yangfan/project1/project1/stamp-1.0/src/res.h File Reference

header of the Resistor class

```
#include "device.h"
```

Classes

- class Resistor

  The resistor.

7.13.1 Detailed Description

header of the Resistor class

Author:

Yinghai

Date:

Fri Sep 12 14:17:02 2008

7.14 /home/yangfan/project1/project1/stamp-1.0/src/stamp.cpp File Reference

Implementation of the Stamp class.

```
#include "stamp.h"
#include "cap.h"
#include "res.h"
#include "isrc.h"
#include "vsrc.h"
#include "mut.h"
#include "ind.h"
#include "util.h"
#include <fstream>
#include <sstream>
```

7.14.1 Detailed Description

Implementation of the Stamp class.

Author:

Yinghai

Date:

Fri Sep 12 13:05:24 2008

7.15 /home/yangfan/project1/project1/stamp-1.0/src/stamp.h File Reference

Header file for Stamp class.

```
#include "mat.h"
#include <map>
#include <vector>
#include <string>
```

Classes

- class Stamp

  Stamp engine.

7.15.1 Detailed Description

Header file for Stamp class.

Author:

Yinghai

Date:

Fri Sep 12 10:58:14 2008

7.16 /home/yangfan/project1/project1/stamp-1.0/src/util.cpp File Reference

Implementation of the utility functions.

```
#include "util.h"
```

Functions

- string tokenizer (string &line, const string &delims)

  This function breaks the string into tokens with the given delimiters.

- void capitalize (string &token)

  this function converts string to upper case

- double to_double (string &str)

  This function converts string to a double considering the units.

7.16.1 Detailed Description

Implementation of the utility functions.

Author:

Yinghai

Date:

Fri Sep 12 14:05:16 2008

7.16.2 Function Documentation

7.16.2.1 void capitalize (string & *token*)

this function converts string to upper case

Parameters:

token string to be converted

7.16.2.2 double to_double (string &)

This function converts string to a double considering the units.

In SPICE, units are used. So *9ns* should equal to *1e-9*. This function just help you take care of such things.

7.16.2.3 string tokenizer (string & *str*, const string & *de*)

This function breaks the string into tokens with the given delimiters.

Parameters:

str the long string to be break

de delimiters

Returns:

first token extracted

This function is a convenient C++ verion of *strtok*. Given the delimiters, it will extract the first token from the string. For example, *tokenizer*("Hello+$World","+$") will return the string *Hello*. You have run it several times in order to extract more tokens. If no more token can be extracted, it will return an empty string. Using this, you can decide whether a line is finished. In boost library, there is a more powerful tokening tool. If you'rer interested, please see to the doc of boost.

7.17 /home/yangfan/project1/project1/stamp-1.0/src/util.h File Reference

Header of the utility funtions.

```
#include <string>
#include <cctype>
#include <cstdlib>
```

Functions

- string tokenizer (string &str, const string &de)
  This function breaks the string into tokens with the given delimiters.
- void capitalize (string &token)
  this function converts string to upper case
- double to_double (string &)
  This function converts string to a double considering the units.

7.17.1 Detailed Description

Header of the utility funtions.

Author:

Yinghai

Date:

Fri Sep 12 13:55:42 2008

7.17.2 Function Documentation

7.17.2.1 void capitalize (string & *token*)

this function converts string to upper case

Parameters:

token string to be converted

7.17.2.2 double to_double (string &)

This function converts string to a double considering the units.

In SPICE, units are used. So *9ns* should equal to *1e-9*. This function just help you take care of such things.

7.17.2.3 string tokenizer (string & *str*, const string & *de*)

This function breaks the string into tokens with the given delimiters.

Parameters:

str the long string to be break

de delimiters

Returns:

first token extracted

This function is a convenient C++ verion of *strtok*. Given the delimiters, it will extract the first token from the string. For example, *tokenizer*("Hello+$World","+$") will return the string *Hello*. You have run it several times in order to extract more tokens. If no more token can be extracted, it will return an empty string. Using this, you can decide whether a line is finished. In boost library, there is a more powerful tokening tool. If you'rer interested, please see to the doc of boost.

7.18 /home/yangfan/project1/project1/stamp-1.0/src/vsrc.cpp File Reference

implementation of voltage source

```
#include "vsrc.h"
```

7.18.1 Detailed Description

implementation of voltage source

Author:

Yinghai

Date:

Fri Sep 12 14:53:40 2008

7.19 /home/yangfan/project1/project1/stamp-1.0/src/vsrc.h File Reference

header of voltage source

```
#include "device.h"
```

Classes

- class Vsrc

  The voltage source.

7.19.1 Detailed Description

header of voltage source

Author:

Yinghai

Date:

Fri Sep 12 14:45:49 2008

Index

∼Mat
  Mat, 22
∼Stamp
  Stamp, 29
/home/yangfan/project1/project1/stamp-1.0/src/cap.cpp, 35
/home/yangfan/project1/project1/stamp-1.0/src/cap.h, 36
/home/yangfan/project1/project1/stamp-1.0/src/device.h, 37
/home/yangfan/project1/project1/stamp-1.0/src/ind.cpp, 38
/home/yangfan/project1/project1/stamp-1.0/src/ind.h, 39
/home/yangfan/project1/project1/stamp-1.0/src/isrc.cpp, 40
/home/yangfan/project1/project1/stamp-1.0/src/isrc.h, 41
/home/yangfan/project1/project1/stamp-1.0/src/main.cpp, 42
/home/yangfan/project1/project1/stamp-1.0/src/mat.h, 43
/home/yangfan/project1/project1/stamp-1.0/src/mut.cpp, 44
/home/yangfan/project1/project1/stamp-1.0/src/mut.h, 45
/home/yangfan/project1/project1/stamp-1.0/src/res.cpp, 46
/home/yangfan/project1/project1/stamp-1.0/src/res.h, 47
/home/yangfan/project1/project1/stamp-1.0/src/stamp.cpp, 48
/home/yangfan/project1/project1/stamp-1.0/src/stamp.h, 49
/home/yangfan/project1/project1/stamp-1.0/src/util.cpp, 50
/home/yangfan/project1/project1/stamp-1.0/src/util.h, 52
/home/yangfan/project1/project1/stamp-1.0/src/vsrc.cpp, 54
/home/yangfan/project1/project1/stamp-1.0/src/vsrc.h, 55

auxNegNode
  Mutual, 24
auxNode
  Inductor, 18
  Vsrc, 33
auxPosNode
  Mutual, 24

Capacitor, 11
  Capacitor, 12
  stamp, 12
capitalize
  util.cpp, 50
  util.h, 52
column
  Mat, 22

Device, 13
  name, 14
  nnode, 14
  pnode, 14
  setName, 15
  setNnode, 15
  setPnode, 15
  setValue, 15
  stamp, 15
  value, 16

ind1
  Mutual, 24
ind2
  Mutual, 25
Inductor, 17
  auxNode, 18
  Inductor, 17
  setAux, 18
  stamp, 18
Isrc, 19
  Isrc, 19
  stamp, 19

main
  main.cpp, 42
main.cpp
  main, 42
Mat, 21
  ∼Mat, 22
  column, 22

Mat, 21
row, 22
Mutual, 23
auxNegNode, 24
auxPosNode, 24
ind1, 24
ind2, 25
Mutual, 24
setAuxNeg, 25
setAuxPos, 25
setInd1, 25
setInd2, 25
stamp, 25

name
Device, 14
nnode
Device, 14

output
Stamp, 30

parse
Stamp, 30
pnode
Device, 14

Resistor, 27
Resistor, 27
stamp, 27
row
Mat, 22

setAux
Inductor, 18
Vsrc, 33
setAuxNeg
Mutual, 25
setAuxPos
Mutual, 25
setInd1
Mutual, 25
setInd2
Mutual, 25
setName
Device, 15
setNnode
Device, 15
setPnode
Device, 15
setup
Stamp, 30
setValue
Device, 15
Stamp, 29
∼Stamp, 29
output, 30
parse, 30
setup, 30
stamp
Capacitor, 12
Device, 15
Inductor, 18
Isrc, 19
Mutual, 25
Resistor, 27
Vsrc, 33

to_double
util.cpp, 50
util.h, 52
tokenizer
util.cpp, 51
util.h, 53

util.cpp
capitalize, 50
to_double, 50
tokenizer, 51
util.h
capitalize, 52
to_double, 52
tokenizer, 53

value
Device, 16
Vsrc, 32
auxNode, 33
setAux, 33
stamp, 33
Vsrc, 32