

SFA

0.1.0

Generated by Doxygen 1.8.13



# Contents

<b>1</b>	<b>Class Index</b>	<b>1</b>
1.1	Class List . . . . .	1
<b>2</b>	<b>File Index</b>	<b>3</b>
2.1	File List . . . . .	3
<b>3</b>	<b>Class Documentation</b>	<b>5</b>
3.1	Netlist::InitDataObj Struct Reference . . . . .	5
3.1.1	Detailed Description . . . . .	5
3.1.2	Member Data Documentation . . . . .	5
3.1.2.1	instArray . . . . .	5
3.1.2.2	netArray . . . . .	6
3.2	Netlist::InitInst Struct Reference . . . . .	6
3.2.1	Detailed Description . . . . .	6
3.2.2	Member Data Documentation . . . . .	6
3.2.2.1	len . . . . .	6
3.2.2.2	name . . . . .	6
3.2.2.3	netIdArray . . . . .	7
3.2.2.4	type . . . . .	7
3.2.2.5	wid . . . . .	7
3.3	Netlist::InitNet Struct Reference . . . . .	7
3.3.1	Detailed Description . . . . .	7
3.3.2	Member Data Documentation . . . . .	7
3.3.2.1	id . . . . .	7

3.3.2.2	name	8
3.4	InitNetlist Class Reference	8
3.4.1	Detailed Description	8
3.4.2	Constructor & Destructor Documentation	9
3.4.2.1	InitNetlist() [1/2]	9
3.4.2.2	InitNetlist() [2/2]	9
3.4.3	Member Function Documentation	9
3.4.3.1	read()	9
3.4.4	Member Data Documentation	9
3.4.4.1	_netlistDB	9
3.5	Inst Class Reference	10
3.5.1	Detailed Description	10
3.5.2	Constructor & Destructor Documentation	11
3.5.2.1	Inst() [1/3]	11
3.5.2.2	Inst() [2/3]	11
3.5.2.3	Inst() [3/3]	11
3.5.3	Member Function Documentation	12
3.5.3.1	addPinId()	12
3.5.3.2	id()	12
3.5.3.3	len()	12
3.5.3.4	name()	12
3.5.3.5	pinIdArray()	13
3.5.3.6	setLen()	13
3.5.3.7	setWid()	13
3.5.3.8	type()	13
3.5.3.9	wid()	13
3.5.4	Member Data Documentation	13
3.5.4.1	_id	14
3.5.4.2	_len	14
3.5.4.3	_name	14

3.5.4.4	<a href="#">_pinIdArray</a>	14
3.5.4.5	<a href="#">_type</a>	14
3.5.4.6	<a href="#">_wid</a>	14
3.6	<a href="#">MosPair Class Reference</a>	14
3.6.1	<a href="#">Detailed Description</a>	15
3.6.2	<a href="#">Constructor &amp; Destructor Documentation</a>	15
3.6.2.1	<a href="#">MosPair() [1/2]</a>	16
3.6.2.2	<a href="#">MosPair() [2/2]</a>	16
3.6.3	<a href="#">Member Function Documentation</a>	16
3.6.3.1	<a href="#">inVld()</a>	16
3.6.3.2	<a href="#">isEqual()</a>	16
3.6.3.3	<a href="#">mosId1()</a>	17
3.6.3.4	<a href="#">mosId2()</a>	17
3.6.3.5	<a href="#">nextPinType1()</a>	17
3.6.3.6	<a href="#">nextPinType2()</a>	17
3.6.3.7	<a href="#">pattern()</a>	17
3.6.3.8	<a href="#">setSrchPinType1()</a>	17
3.6.3.9	<a href="#">setSrchPinType2()</a>	18
3.6.3.10	<a href="#">srchPinType1()</a>	18
3.6.3.11	<a href="#">srchPinType2()</a>	18
3.6.3.12	<a href="#">valid()</a>	18
3.6.4	<a href="#">Member Data Documentation</a>	18
3.6.4.1	<a href="#">_mosId1</a>	18
3.6.4.2	<a href="#">_mosId2</a>	19
3.6.4.3	<a href="#">_pattern</a>	19
3.6.4.4	<a href="#">_srchPinType1</a>	19
3.6.4.5	<a href="#">_srchPinType2</a>	19
3.6.4.6	<a href="#">_valid</a>	19
3.7	<a href="#">Net Class Reference</a>	19
3.7.1	<a href="#">Detailed Description</a>	20

3.7.2	Constructor & Destructor Documentation . . . . .	20
3.7.2.1	Net() [1/2] . . . . .	20
3.7.2.2	Net() [2/2] . . . . .	20
3.7.3	Member Function Documentation . . . . .	20
3.7.3.1	addPinId() . . . . .	20
3.7.3.2	id() . . . . .	21
3.7.3.3	name() . . . . .	21
3.7.3.4	netType() . . . . .	21
3.7.3.5	pinIdArray() . . . . .	21
3.7.4	Member Data Documentation . . . . .	21
3.7.4.1	_id . . . . .	21
3.7.4.2	_name . . . . .	21
3.7.4.3	_pinIdArray . . . . .	22
3.8	Netlist Class Reference . . . . .	22
3.8.1	Detailed Description . . . . .	23
3.8.2	Constructor & Destructor Documentation . . . . .	24
3.8.2.1	Netlist() . . . . .	24
3.8.3	Member Function Documentation . . . . .	24
3.8.3.1	addInst() . . . . .	24
3.8.3.2	addNet() . . . . .	24
3.8.3.3	addPin() . . . . .	24
3.8.3.4	drainNetId() . . . . .	25
3.8.3.5	fltrInstMosType() . . . . .	25
3.8.3.6	fltrInstNetConnPinType() . . . . .	25
3.8.3.7	fltrInstPinConnPinType() . . . . .	26
3.8.3.8	gateNetId() . . . . .	26
3.8.3.9	getInstNetConn() . . . . .	26
3.8.3.10	getInstPinConn() . . . . .	26
3.8.3.11	getPinTypeInstNetConn() . . . . .	27
3.8.3.12	getPinTypeInstPinConn() . . . . .	27

3.8.3.13	<a href="#">init()</a>	28
3.8.3.14	<a href="#">inst()</a>	28
3.8.3.15	<a href="#">instNetId()</a>	28
3.8.3.16	<a href="#">instPinId()</a>	28
3.8.3.17	<a href="#">isMos()</a>	29
3.8.3.18	<a href="#">isPasvDev()</a>	29
3.8.3.19	<a href="#">isSignal()</a>	29
3.8.3.20	<a href="#">mosType()</a>	29
3.8.3.21	<a href="#">net()</a>	30
3.8.3.22	<a href="#">numInst()</a>	30
3.8.3.23	<a href="#">numNet()</a>	30
3.8.3.24	<a href="#">numPin()</a>	30
3.8.3.25	<a href="#">pin()</a>	30
3.8.3.26	<a href="#">print_all()</a>	30
3.8.3.27	<a href="#">rmvInstHasPin()</a>	30
3.8.3.28	<a href="#">srcNetId()</a>	31
3.8.4	<a href="#">Member Data Documentation</a>	31
3.8.4.1	<a href="#">_instArray</a>	31
3.8.4.2	<a href="#">_netArray</a>	31
3.8.4.3	<a href="#">_pinArray</a>	31
3.9	<a href="#">Pattern Class Reference</a>	32
3.9.1	<a href="#">Detailed Description</a>	33
3.9.2	<a href="#">Constructor &amp; Destructor Documentation</a>	33
3.9.2.1	<a href="#">Pattern()</a>	33
3.9.3	<a href="#">Member Function Documentation</a>	33
3.9.3.1	<a href="#">crossPairCascode()</a>	33
3.9.3.2	<a href="#">crossPairLoad()</a>	33
3.9.3.3	<a href="#">diffPairCascode()</a>	34
3.9.3.4	<a href="#">diffPairInput()</a>	34
3.9.3.5	<a href="#">matchedSize()</a>	34

3.9.3.6	<a href="#">matchedType()</a>	34
3.9.3.7	<a href="#">pattern()</a>	34
3.9.3.8	<a href="#">validPairCascode()</a>	35
3.9.3.9	<a href="#">validPairLoad()</a>	35
3.9.4	<a href="#">Member Data Documentation</a>	35
3.9.4.1	<a href="#">_netlist</a>	35
3.10	<a href="#">Pin Class Reference</a>	35
3.10.1	<a href="#">Detailed Description</a>	36
3.10.2	<a href="#">Constructor &amp; Destructor Documentation</a>	36
3.10.2.1	<a href="#">Pin() [1/2]</a>	36
3.10.2.2	<a href="#">Pin() [2/2]</a>	36
3.10.3	<a href="#">Member Function Documentation</a>	37
3.10.3.1	<a href="#">id()</a>	37
3.10.3.2	<a href="#">instId()</a>	37
3.10.3.3	<a href="#">isPasvDev()</a>	37
3.10.3.4	<a href="#">netId()</a>	37
3.10.3.5	<a href="#">nextPinType()</a>	37
3.10.3.6	<a href="#">type()</a>	38
3.10.4	<a href="#">Member Data Documentation</a>	38
3.10.4.1	<a href="#">_id</a>	38
3.10.4.2	<a href="#">_instId</a>	38
3.10.4.3	<a href="#">_netId</a>	39
3.10.4.4	<a href="#">_type</a>	39
3.11	<a href="#">SymDetect Class Reference</a>	39
3.11.1	<a href="#">Detailed Description</a>	40
3.11.2	<a href="#">Constructor &amp; Destructor Documentation</a>	40
3.11.2.1	<a href="#">SymDetect()</a>	40
3.11.3	<a href="#">Member Function Documentation</a>	41
3.11.3.1	<a href="#">addSelfSym()</a>	41
3.11.3.2	<a href="#">dfsDiffPair()</a>	41



3.11.3.3	<a href="#">endSrch()</a>	42
3.11.3.4	<a href="#">existPair()</a> [1/2]	42
3.11.3.5	<a href="#">existPair()</a> [2/2]	42
3.11.3.6	<a href="#">getDiffPair()</a>	42
3.11.3.7	<a href="#">getPatrnNetConn()</a>	43
3.11.3.8	<a href="#">getVldDrainMos()</a>	43
3.11.3.9	<a href="#">hiSymDetect()</a>	43
3.11.3.10	<a href="#">inVldDiffPairSrch()</a>	44
3.11.3.11	<a href="#">MosPairPtrn()</a>	44
3.11.3.12	<a href="#">pushNextSrchObj()</a>	44
3.11.3.13	<a href="#">selfSymSrch()</a>	45
3.11.3.14	<a href="#">validDiffPair()</a>	45
3.11.3.15	<a href="#">validSrchObj()</a>	46
3.11.4	<a href="#">Member Data Documentation</a>	46
3.11.4.1	<a href="#">_netlist</a>	47
3.11.4.2	<a href="#">_pattern</a>	47
<b>4</b>	<b><a href="#">File Documentation</a></b>	<b>49</b>
4.1	<a href="#">src/db/Inst.h File Reference</a>	49
4.1.1	<a href="#">Detailed Description</a>	50
4.2	<a href="#">src/db/MosPair.cpp File Reference</a>	50
4.2.1	<a href="#">Detailed Description</a>	51
4.3	<a href="#">src/db/MosPair.h File Reference</a>	51
4.3.1	<a href="#">Detailed Description</a>	53
4.4	<a href="#">src/db/Net.cpp File Reference</a>	53
4.4.1	<a href="#">Detailed Description</a>	54
4.4.2	<a href="#">Variable Documentation</a>	54
4.4.2.1	<a href="#">GROUND_NET_NAMES</a>	54
4.4.2.2	<a href="#">POWER_NET_NAMES</a>	54
4.5	<a href="#">src/db/Net.h File Reference</a>	54
4.5.1	<a href="#">Detailed Description</a>	56

4.6	src/db/Netlist.cpp File Reference	56
4.6.1	Detailed Description	57
4.6.2	Variable Documentation	57
4.6.2.1	MOS_PIN_TYPE	57
4.6.2.2	RES_PIN_TYPE	57
4.7	src/db/Netlist.h File Reference	58
4.7.1	Detailed Description	59
4.8	src/db/Pin.cpp File Reference	59
4.8.1	Detailed Description	60
4.9	src/db/Pin.h File Reference	60
4.9.1	Detailed Description	61
4.10	src/global/global.h File Reference	61
4.10.1	Detailed Description	62
4.11	src/global/namespace.h File Reference	63
4.11.1	Detailed Description	63
4.11.2	Macro Definition Documentation	63
4.11.2.1	PROJECT_NAMESPACE	64
4.11.2.2	PROJECT_NAMESPACE_BEGIN	64
4.11.2.3	PROJECT_NAMESPACE_END	64
4.12	src/global/type.h File Reference	64
4.12.1	Detailed Description	66
4.12.2	Typedef Documentation	66
4.12.2.1	Byte	66
4.12.2.2	IndexType	66
4.12.2.3	IntType	66
4.12.2.4	RealType	66
4.12.3	Enumeration Type Documentation	66
4.12.3.1	InstType	66
4.12.3.2	MosPattern	67
4.12.3.3	MosType	67

4.12.3.4	NetType	68
4.12.3.5	PinType	68
4.12.4	Variable Documentation	68
4.12.4.1	INDEX_TYPE_MAX	68
4.12.4.2	INT_TYPE_MAX	69
4.12.4.3	INT_TYPE_MIN	69
4.12.4.4	REAL_TYPE_MAX	69
4.12.4.5	REAL_TYPE_MIN	69
4.12.4.6	REAL_TYPE_TOL	69
4.13	src/main/main.cpp File Reference	69
4.13.1	Detailed Description	70
4.13.2	Macro Definition Documentation	71
4.13.2.1	__SFA_TEST__	71
4.13.3	Function Documentation	71
4.13.3.1	main()	71
4.14	src/parser/InitNetlist.cpp File Reference	71
4.14.1	Detailed Description	72
4.15	src/parser/InitNetlist.h File Reference	72
4.15.1	Detailed Description	74
4.16	src/sym_detect/Pattern.cpp File Reference	74
4.16.1	Detailed Description	75
4.17	src/sym_detect/Pattern.h File Reference	75
4.17.1	Detailed Description	76
4.18	src/sym_detect/SymDetect.cpp File Reference	76
4.18.1	Detailed Description	77
4.19	src/sym_detect/SymDetect.h File Reference	78
4.19.1	Detailed Description	79



# Chapter 1

## Class Index

### 1.1 Class List

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

<a href="#">Netlist::InitDataObj</a>	
Instantiate <a href="#">Netlist</a> class . . . . .	5
<a href="#">Netlist::InitInst</a>	
<a href="#">Inst</a> for instantiation . . . . .	6
<a href="#">Netlist::InitNet</a>	
<a href="#">Net</a> for instantiation . . . . .	7
<a href="#">InitNetlist</a>	
<a href="#">InitNetlist</a> class . . . . .	8
<a href="#">Inst</a>	
<a href="#">Inst</a> class . . . . .	10
<a href="#">MosPair</a>	
A pair of Mosfet with <a href="#">MosPattern</a> . . . . .	14
<a href="#">Net</a>	
<a href="#">Net</a> class . . . . .	19
<a href="#">Netlist</a>	
<a href="#">Netlist</a> class . . . . .	22
<a href="#">Pattern</a>	
<a href="#">Pattern</a> class . . . . .	32
<a href="#">Pin</a>	
<a href="#">Pin</a> class . . . . .	35
<a href="#">SymDetect</a>	
<a href="#">SymDetect</a> class . . . . .	39



## Chapter 2

# File Index

### 2.1 File List

Here is a list of all files with brief descriptions:

src/db/ <a href="#">Inst.h</a>	
Instance class . . . . .	49
src/db/ <a href="#">MosPair.cpp</a>	
MosPair implementation . . . . .	50
src/db/ <a href="#">MosPair.h</a>	
A pair of Mosfet with MosPattern . . . . .	51
src/db/ <a href="#">Net.cpp</a>	
Net class implementation . . . . .	53
src/db/ <a href="#">Net.h</a>	
Net class . . . . .	54
src/db/ <a href="#">Netlist.cpp</a>	
Netlist class implementation . . . . .	56
src/db/ <a href="#">Netlist.h</a>	
Netlist class . . . . .	58
src/db/ <a href="#">Pin.cpp</a>	
Net class implementation . . . . .	59
src/db/ <a href="#">Pin.h</a>	
Pin class . . . . .	60
src/global/ <a href="#">global.h</a>	
Global header file . . . . .	61
src/global/ <a href="#">namespace.h</a>	
Namespace header file . . . . .	63
src/global/ <a href="#">type.h</a>	
Type header file . . . . .	64
src/main/ <a href="#">main.cpp</a>	
Main.cpp . . . . .	69
src/parser/ <a href="#">InitNetlist.cpp</a>	
Parser implementation . . . . .	71
src/parser/ <a href="#">InitNetlist.h</a>	
Parser to initialize netlist . . . . .	72
src/sym_detect/ <a href="#">Pattern.cpp</a>	
Pattern definitions . . . . .	74
src/sym_detect/ <a href="#">Pattern.h</a>	
Mosfet pair patterns . . . . .	75
src/sym_detect/ <a href="#">SymDetect.cpp</a>	
Detect symmetric patterns . . . . .	76
src/sym_detect/ <a href="#">SymDetect.h</a>	
Detect symmetric patterns . . . . .	78





## Chapter 3

# Class Documentation

### 3.1 Netlist::InitDataObj Struct Reference

Instantiate [Netlist](#) class.

```
#include <Netlist.h>
```

#### Public Attributes

- `std::vector< InitNet > netArray`
- `std::vector< InitInst > instArray`

#### 3.1.1 Detailed Description

Instantiate [Netlist](#) class.

See also

[init\(InitDataObj &\).](#)

#### 3.1.2 Member Data Documentation

##### 3.1.2.1 instArray

```
std::vector<InitInst> Netlist::InitDataObj::instArray
```

### 3.1.2.2 netArray

```
std::vector<InitNet> Netlist::InitDataObj::netArray
```

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

- [src/db/Netlist.h](#)

## 3.2 Netlist::InitInst Struct Reference

[Inst](#) for instantiation.

```
#include <Netlist.h>
```

### Public Attributes

- [InstType](#) type = [InstType::OTHER](#)
- std::vector< [IndexType](#) > [netIdArray](#)
- std::string [name](#)
- [RealType](#) wid = 0
- [RealType](#) len = 0

### 3.2.1 Detailed Description

[Inst](#) for instantiation.

### 3.2.2 Member Data Documentation

#### 3.2.2.1 len

```
RealType Netlist::InitInst::len = 0
```

#### 3.2.2.2 name

```
std::string Netlist::InitInst::name
```

### 3.2.2.3 netIdArray

```
std::vector<IndexType> Netlist::InitInst::netIdArray
```

### 3.2.2.4 type

```
InstType Netlist::InitInst::type = InstType::OTHER
```

### 3.2.2.5 wid

```
RealType Netlist::InitInst::wid = 0
```

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

- [src/db/Netlist.h](#)

## 3.3 Netlist::InitNet Struct Reference

[Net](#) for instantiation.

```
#include <Netlist.h>
```

### Public Attributes

- std::string [name](#)
- [IndexType](#) [id](#) = [INDEX\\_TYPE\\_MAX](#)

### 3.3.1 Detailed Description

[Net](#) for instantiation.

### 3.3.2 Member Data Documentation

#### 3.3.2.1 id

```
IndexType Netlist::InitNet::id = INDEX\_TYPE\_MAX
```

### 3.3.2.2 name

```
std::string Netlist::InitNet::name
```

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

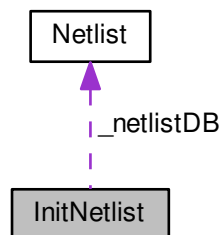
- [src/db/Netlist.h](#)

## 3.4 InitNetlist Class Reference

[InitNetlist](#) class.

```
#include <InitNetlist.h>
```

Collaboration diagram for InitNetlist:



### Public Member Functions

- [InitNetlist](#) ()=default  
*Default Constructor.*
- [InitNetlist](#) ([Netlist](#) &netlist)  
*Constructor with initialization.*
- bool [read](#) (const std::string &filename)  
*Parse file and build netlist.*

### Private Attributes

- [Netlist](#) & [\\_netlistDB](#)

### 3.4.1 Detailed Description

[InitNetlist](#) class.

## 3.4.2 Constructor & Destructor Documentation

### 3.4.2.1 InitNetlist() [1/2]

```
InitNetlist::InitNetlist ( ) [explicit], [default]
```

Default Constructor.

### 3.4.2.2 InitNetlist() [2/2]

```
InitNetlist::InitNetlist (
    Netlist & netlist ) [inline], [explicit]
```

Constructor with initialization.

## 3.4.3 Member Function Documentation

### 3.4.3.1 read()

```
PROJECT_NAMESPACE_BEGIN bool InitNetlist::read (
    const std::string & filename )
```

Parse file and build netlist.

Input files should follow same format generated through scripts/create\_init\_obj.py. Sample input files for c++ are under benchmarks. The python scripts take standardized hspice/spectre netlist files as inputs.

#### Parameters

<i>filename</i>	Input file to parse.
-----------------	----------------------

## 3.4.4 Member Data Documentation

### 3.4.4.1 \_netlistDB

```
Netlist& InitNetlist::_netlistDB [private]
```

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

- [src/parser/InitNetlist.h](#)
- [src/parser/InitNetlist.cpp](#)

## 3.5 Inst Class Reference

[Inst](#) class.

```
#include <Inst.h>
```

### Public Member Functions

- [Inst](#) ()=default  
*Default constructor.*
- [Inst](#) (const std::string &[name](#), [InstType](#) type, [IndexType](#) id)  
*Constructor for [Inst](#).*
- [Inst](#) (const std::string &[name](#), [InstType](#) type, [IndexType](#) id, [RealType](#) wid, [RealType](#) len)  
*Constructor for [Inst](#).*
- const std::string & [name](#) () const
- [InstType](#) type () const  
*Return type of [Inst](#).*
- [IndexType](#) id () const  
*Return Id of [Inst](#).*
- const std::vector< [IndexType](#) > & [pinIdArray](#) () const  
*Return the index array for pins of the [Inst](#).*
- [RealType](#) wid () const  
*Return width of [Inst](#).*
- [RealType](#) len () const  
*Return length of [Inst](#).*
- void [addPinId](#) ([IndexType](#) pinId)  
*Add pin index to [Inst](#).*
- void [setWid](#) ([RealType](#) wid)  
*Assign width of [Inst](#).*
- void [setLen](#) ([RealType](#) len)  
*Assign length of [Inst](#).*

### Private Attributes

- std::string [\\_name](#)
- [InstType](#) [\\_type](#)
- [IndexType](#) [\\_id](#)
- std::vector< [IndexType](#) > [\\_pinIdArray](#)
- [RealType](#) [\\_wid](#)
- [RealType](#) [\\_len](#)

#### 3.5.1 Detailed Description

[Inst](#) class.

## 3.5.2 Constructor & Destructor Documentation

### 3.5.2.1 `Inst()` [1/3]

```
Inst::Inst ( ) [explicit], [default]
```

Default constructor.

### 3.5.2.2 `Inst()` [2/3]

```
Inst::Inst (
    const std::string & name,
    InstType type,
    IndexType id ) [inline], [explicit]
```

Constructor for [Inst](#).

Constructor for netlist instances that does not have width and length attributes.

#### Parameters

<i>name</i>	Name of <a href="#">Inst</a> .
<i>type</i>	Type of <a href="#">Inst</a> . Member of InstType.

#### See also

[type.h](#)

#### Parameters

<i>id</i>	Id of <a href="#">Inst</a> .
-----------	------------------------------

### 3.5.2.3 `Inst()` [3/3]

```
Inst::Inst (
    const std::string & name,
    InstType type,
    IndexType id,
    RealType wid,
    RealType len ) [inline], [explicit]
```

Constructor for [Inst](#).

Constructor for netlist instances that have width and length attributes.

## Parameters

<i>name</i>	Name of <a href="#">Inst.</a>
<i>type</i>	Type of <a href="#">Inst.</a> Member of InstType.
<i>id</i>	Id of INst.
<i>wid</i>	Width of <a href="#">Inst.</a>
<i>len</i>	Length of <a href="#">Inst.</a>

## 3.5.3 Member Function Documentation

## 3.5.3.1 addPinId()

```
void Inst::addPinId (
    IndexType pinId ) [inline]
```

Add pin index to [Inst.](#)

## Parameters

<i>pin↔ Id</i>	Added pin Id.
--------------------	---------------

## 3.5.3.2 id()

```
IndexType Inst::id ( ) const [inline]
```

Return Id of [Inst.](#)

## 3.5.3.3 len()

```
RealType Inst::len ( ) const [inline]
```

Return length of [Inst.](#)

## 3.5.3.4 name()

```
const std::string& Inst::name ( ) const [inline]
```

Return name of [Inst.](#)



#### 3.5.3.5 pinIdArray()

```
const std::vector<IndexType>& Inst::pinIdArray ( ) const [inline]
```

Return the index array for pins of the [Inst](#).

#### 3.5.3.6 setLen()

```
void Inst::setLen (
    RealType len ) [inline]
```

Assign length of [Inst](#).

#### 3.5.3.7 setWid()

```
void Inst::setWid (
    RealType wid ) [inline]
```

Assign width of [Inst](#).

#### 3.5.3.8 type()

```
InstType Inst::type ( ) const [inline]
```

Return type of [Inst](#).

See also

[InstType](#)

#### 3.5.3.9 wid()

```
RealType Inst::wid ( ) const [inline]
```

Return width of [Inst](#).

### 3.5.4 Member Data Documentation

#### 3.5.4.1 `_id`

```
IndexType Inst::_id [private]
```

#### 3.5.4.2 `_len`

```
RealType Inst::_len [private]
```

#### 3.5.4.3 `_name`

```
std::string Inst::_name [private]
```

#### 3.5.4.4 `_pinIdArray`

```
std::vector<IndexType> Inst::_pinIdArray [private]
```

#### 3.5.4.5 `_type`

```
InstType Inst::_type [private]
```

#### 3.5.4.6 `_wid`

```
RealType Inst::_wid [private]
```

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

- `src/db/Inst.h`

## 3.6 MosPair Class Reference

A pair of Mosfet with MosPattern.

```
#include <MosPair.h>
```

## Public Member Functions

- [MosPair](#) ()=default  
*Default Constructor.*
- [MosPair](#) ([IndexType](#) mosId1, [IndexType](#) mosId2, [MosPattern](#) pattern)  
*Constructor for [MosPair](#).*
- [IndexType](#) mosId1 () const  
*Get mosId1.*
- [IndexType](#) mosId2 () const  
*Get mosId2.*
- bool [valid](#) () const  
*Return if valid search pair.*
- [MosPattern](#) pattern () const  
*Get pattern.*
- [PinType](#) srchPinType1 () const  
*Get PinType on how DFS reached mosId1 of the pair.*
- [PinType](#) srchPinType2 () const  
*Get PinType on how DFS reached mosId1 of the pair.*
- void [inVld](#) ()  
*Invalidate pair.*
- void [setSrchPinType1](#) ([PinType](#) type)  
*set reached PinType.*
- void [setSrchPinType2](#) ([PinType](#) type)  
*set reached PinType.*
- [PinType](#) nextPinType1 ()  
*Return next PinType to search for mosId1.*
- [PinType](#) nextPinType2 ()  
*Return next PinType to search for mosId2.*
- bool [isEqual](#) (const [MosPair](#) &right) const  
*Equal operator.*

## Private Attributes

- [IndexType](#) \_mosId1
- [IndexType](#) \_mosId2
- [MosPattern](#) \_pattern
- bool \_valid
- [PinType](#) \_srchPinType1
- [PinType](#) \_srchPinType2

### 3.6.1 Detailed Description

A pair of Mosfet with MosPattern.

This class stores a pair of Mosfet Id and also assists DFS in [SymDetect.h](#). This class has no reference to netlist, pattern needs to be set at construction.

### 3.6.2 Constructor & Destructor Documentation

### 3.6.2.1 MosPair() [1/2]

```
MosPair::MosPair ( ) [explicit], [default]
```

Default Constructor.

### 3.6.2.2 MosPair() [2/2]

```
MosPair::MosPair (
    IndexType mosId1,
    IndexType mosId2,
    MosPattern pattern ) [inline], [explicit]
```

Constructor for [MosPair](#).

Sequence of Ids does not matter. pattern is set according to input.

#### Parameters

<i>mosId1</i>	Id for Mos1
<i>mosId2</i>	Id for Mos2

< valid is set true as default.

< reached [Pin](#) set as SOURCE default.

## 3.6.3 Member Function Documentation

### 3.6.3.1 inVld()

```
void MosPair::inVld ( ) [inline]
```

Invalidate pair.

### 3.6.3.2 isEqual()

```
PROJECT_NAMESPACE_BEGIN bool MosPair::isEqual (
    const MosPair & right ) const
```

Equal operator.

Two pairs are equal if Id are equal. Sequence of Id does not matter.

### 3.6.3.3 mosId1()

```
IndexType MosPair::mosId1 ( ) const [inline]
```

Get mosId1.

### 3.6.3.4 mosId2()

```
IndexType MosPair::mosId2 ( ) const [inline]
```

Get mosId2.

### 3.6.3.5 nextPinType1()

```
PinType MosPair::nextPinType1 ( ) [inline]
```

Return next PinType to search for mosId1.

### 3.6.3.6 nextPinType2()

```
PinType MosPair::nextPinType2 ( ) [inline]
```

Return next PinType to search for mosId2.

### 3.6.3.7 pattern()

```
MosPattern MosPair::pattern ( ) const [inline]
```

Get pattern.

### 3.6.3.8 setSrchPinType1()

```
void MosPair::setSrchPinType1 (
    PinType type ) [inline]
```

set reached PinType.

This is how mosId1 of the pair is reached through DFS search.

### 3.6.3.9 setSrchPinType2()

```
void MosPair::setSrchPinType2 (
    PinType type ) [inline]
```

set reached PinType.

This is how mosId2 of the pair is reached through DFS search.

### 3.6.3.10 srchPinType1()

```
PinType MosPair::srchPinType1 ( ) const [inline]
```

Get PinType on how DFS reached mosId1 of the pair.

### 3.6.3.11 srchPinType2()

```
PinType MosPair::srchPinType2 ( ) const [inline]
```

Get PinType on how DFS reached mosId1 of the pair.

### 3.6.3.12 valid()

```
bool MosPair::valid ( ) const [inline]
```

Return if valid search pair.

See also

[SymDetect::inVldDiffPairSrch](#)

## 3.6.4 Member Data Documentation

### 3.6.4.1 \_mosId1

```
IndexType MosPair::_mosId1 [private]
```

3.6.4.2 `_mosId2`

```
IndexType MosPair::_mosId2 [private]
```

3.6.4.3 `_pattern`

```
MosPattern MosPair::_pattern [private]
```

3.6.4.4 `_srchPinType1`

```
PinType MosPair::_srchPinType1 [private]
```

3.6.4.5 `_srchPinType2`

```
PinType MosPair::_srchPinType2 [private]
```

3.6.4.6 `_valid`

```
bool MosPair::_valid [private]
```

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

- [src/db/MosPair.h](#)
- [src/db/MosPair.cpp](#)

## 3.7 Net Class Reference

[Net](#) class.

```
#include <Net.h>
```

### Public Member Functions

- [Net](#) ()=default
- [Net](#) (const std::string &name, [IndexType](#) id)  
*Constructor of [Net](#).*
- const std::string & [name](#) () const
- [IndexType](#) id () const
- const std::vector< [IndexType](#) > & [pinIdArray](#) () const
- void [addPinId](#) ([IndexType](#) pinId)
- [NetType](#) [netType](#) () const  
*Return net type.*

## Private Attributes

- `std::string _name`
- `IndexType _id`
- `std::vector< IndexType > _pinIdArray`

### 3.7.1 Detailed Description

`Net` class.

### 3.7.2 Constructor & Destructor Documentation

#### 3.7.2.1 `Net()` [1/2]

```
Net::Net ( ) [explicit], [default]
```

Default constructor.

#### 3.7.2.2 `Net()` [2/2]

```
Net::Net (
    const std::string & name,
    IndexType id ) [inline], [explicit]
```

Constructor of `Net`.

#### Parameters

<i>name</i>	Name of <code>Net</code> .
<i>id</i>	Id of <code>Net</code> .

### 3.7.3 Member Function Documentation

#### 3.7.3.1 `addPinId()`

```
void Net::addPinId (
    IndexType pinId ) [inline]
```

Connect a pin to the net.



### 3.7.3.2 id()

```
IndexType Net::id ( ) const [inline]
```

Return Id of [Net](#).

### 3.7.3.3 name()

```
const std::string& Net::name ( ) const [inline]
```

Return name of [Net](#).

### 3.7.3.4 netType()

```
NetType Net::netType ( ) const
```

Return net type.

See also

[NetType](#).

Return netType of net based on name. Currently supported Power/Ground names are limited to conventional VD↔D/VSS. Add unsupported names for Power/Ground filtering to POWER\_NET\_NAMES and GROUND\_NET\_NAMES to /db/Net.cpp.

### 3.7.3.5 pinIdArray()

```
const std::vector<IndexType>& Net::pinIdArray ( ) const [inline]
```

Return index array of connected pins.

## 3.7.4 Member Data Documentation

### 3.7.4.1 \_id

```
IndexType Net::_id [private]
```

### 3.7.4.2 \_name

```
std::string Net::_name [private]
```

### 3.7.4.3 \_pinIdArray

```
std::vector<IndexType> Net::_pinIdArray [private]
```

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

- src/db/Net.h
- src/db/Net.cpp

## 3.8 Netlist Class Reference

[Netlist](#) class.

```
#include <Netlist.h>
```

### Classes

- struct [InitDataObj](#)  
*Instantiate [Netlist](#) class.*
- struct [InitInst](#)  
*[Inst](#) for instantiation.*
- struct [InitNet](#)  
*[Net](#) for instantiation.*

### Public Member Functions

- [Netlist](#) ()=default  
*Default Constructor.*
- void [init](#) ([InitDataObj](#) &obj)  
*Initialize [Netlist](#) class.*
- void [print\\_all](#) () const
- bool [isMos](#) ([InstType](#) instType) const  
*Return true if [InstType](#) is a Mosfet. NMOS and PMOS are Mosfets.*
- bool [isPasvDev](#) ([InstType](#) instType) const  
*Return true if [InstType](#) is passive device. RES and CAP are passive devices.*
- bool [isSignal](#) ([IndexType](#) netId) const  
*Return true if corresponding net [NetType::Signal](#).*
- [MosType](#) [mosType](#) ([IndexType](#) mosId) const  
*Return MosType of corresponding instance id.*
- [IndexType](#) [instNetId](#) ([IndexType](#) instId, [PinType](#) pinType) const  
*Return Id of [Net](#) connected to [Inst](#) by certain [PinType](#).*
- [IndexType](#) [instPinId](#) ([IndexType](#) instId, [PinType](#) pinType) const  
*Return Id of [Pin](#) with [PinType](#) connected to [Inst](#).*
- [IndexType](#) [srcNetId](#) ([IndexType](#) mosId) const  
*Return Source [Net](#) Id of [Inst](#) mosId. Equivalent as [instNetId](#)(mosId, [PinType::SOURCE](#));.*
- [IndexType](#) [drainNetId](#) ([IndexType](#) mosId) const  
*Return Drain [Net](#) Id of [Inst](#) mosId. Equivalent as [instNetId](#)(mosId, [PinType::DRAIN](#));.*
- [IndexType](#) [gateNetId](#) ([IndexType](#) mosId) const

- Return Gate *Net* Id of *Inst* mosId. Equivalent as `instNetId(mosId, PinType::GATE);`.
- `PinType getPinTypeInstPinConn (IndexType instId, IndexType pinId) const`  
Get *PinType* of a pin such that *Inst* and *Pin* are connected through this pin.
  - `PinType getPinTypeInstNetConn (IndexType instId, IndexType netId) const`  
Get *PinType* of a pin such that *Inst* and *Net* are connected through this pin.
  - `void getInstNetConn (std::vector< IndexType > &instArray, IndexType netId) const`  
Get all *Inst* that are connected to *netId*.
  - `void getInstPinConn (std::vector< IndexType > &instArray, IndexType pinId) const`  
Get all *Inst* that are connected to *pinId*(through some net).
  - `void rmvInstHasPin (std::vector< IndexType > &instArray, IndexType pinId) const`  
Remove from array, *Inst* that has *pinId*.
  - `void fltrInstPinConnPinType (std::vector< IndexType > &instArray, IndexType pinId, PinType connPinType) const`  
Filter *instArray*. Remove *Inst* that are connected to *pinId* through *connPinType*.
  - `void fltrInstNetConnPinType (std::vector< IndexType > &instArray, IndexType netId, PinType connPinType) const`  
Filter *instArray*. Remove *Inst* that are connected to *netId* through *connPinType*.
  - `void fltrInstMosType (std::vector< IndexType > &instArray, MosType mosType) const`  
Filter *instArray*. Remove *Inst* whose type are *mosType*.
  - `const Pin & pin (IndexType id) const`  
Return *Pin* of *Id*.
  - `const Net & net (IndexType id) const`  
Return *Net* of *Id*.
  - `const Inst & inst (IndexType id) const`  
Return *Inst* of *Id*.
  - `IndexType numPin () const`  
Return number of *Pin*.
  - `IndexType numNet () const`  
Return number of *Net*.
  - `IndexType numInst () const`  
Return number of *Inst*.
  - `void addPin (Pin &pin)`  
Add *Pin* to *Netlist*.
  - `void addNet (Net &net)`  
Add *Net* to *Netlist*.
  - `void addInst (Inst &inst)`  
Add *Inst* to *Netlist*.

## Private Attributes

- `std::vector< Net > _netArray`
- `std::vector< Pin > _pinArray`
- `std::vector< Inst > _instArray`

### 3.8.1 Detailed Description

[Netlist](#) class.

## 3.8.2 Constructor & Destructor Documentation

### 3.8.2.1 Netlist()

```
Netlist::Netlist ( ) [explicit], [default]
```

Default Constructor.

## 3.8.3 Member Function Documentation

### 3.8.3.1 addInst()

```
void Netlist::addInst (
    Inst & inst ) [inline]
```

Add [Inst](#) to [Netlist](#).

### 3.8.3.2 addNet()

```
void Netlist::addNet (
    Net & net ) [inline]
```

Add [Net](#) to [Netlist](#).

### 3.8.3.3 addPin()

```
void Netlist::addPin (
    Pin & pin ) [inline]
```

Add [Pin](#) to [Netlist](#).

#### 3.8.3.4 drainNetId()

```
IndexType Netlist::drainNetId (
    IndexType mosId ) const [inline]
```

Return Drain [Net](#) Id of [Inst](#) mosId. Equivalent as instNetId(mosId, PinType::DRAIN);.

See also

[instNetId](#)

#### 3.8.3.5 fltrInstMosType()

```
void Netlist::fltrInstMosType (
    std::vector< IndexType > & instArray,
    MosType mosType ) const
```

Filter instArray. Remove [Inst](#) whose type are mosType.

Removed instId if mosType(instId) == mosType. O(n) complexity. Similar implementation of std::remove().

See also

[getPinTypeInstNetConn.](#)

#### 3.8.3.6 fltrInstNetConnPinType()

```
void Netlist::fltrInstNetConnPinType (
    std::vector< IndexType > & instArray,
    IndexType netId,
    PinType connPinType ) const
```

Filter instArray. Remove [Inst](#) that are connected to netId through connPinType.

Removed instId if getPinTypeInstNetConn(instId, pinId) == connPinType. O(n) complexity. Similar implementation of std::remove().

See also

[getPinTypeInstNetConn.](#)

### 3.8.3.7 fltrInstPinConnPinType()

```
void Netlist::fltrInstPinConnPinType (
    std::vector< IndexType > & instArray,
    IndexType pinId,
    PinType connPinType ) const
```

Filter instArray. Remove [Inst](#) that are connected to pinId through connPinType.

Removed instId if getPinTypeInstPinConn(instId, pinId) == connPinType. O(n) complexity. Similar implementation of std::remove().

See also

[getPinTypeInstPinConn](#).

### 3.8.3.8 gateNetId()

```
IndexType Netlist::gateNetId (
    IndexType mosId ) const [inline]
```

Return Gate [Net](#) Id of [Inst](#) mosId. Equivalent as instNetId(mosId, PinType::GATE);.

See also

[instNetId](#).

### 3.8.3.9 getInstNetConn()

```
void Netlist::getInstNetConn (
    std::vector< IndexType > & instArray,
    IndexType netId ) const
```

Get all [Inst](#) that are connected to netId.

Parameters

out	<i>instArray</i>	Array of the returned <a href="#">Inst</a> Id.
in	<i>netId</i>	Id of net.

### 3.8.3.10 getInstPinConn()

```
void Netlist::getInstPinConn (
```

```
std::vector< IndexType > & instArray,
IndexType pinId ) const
```

Get all [Inst](#) that are connected to pinId(through some net).

The instance that pinId itself belongs to is not returned.

#### Parameters

out	<i>instArray</i>	Array of the returned <a href="#">Inst</a> Id.
in	<i>pinId</i>	Id of pin.

#### 3.8.3.11 getPinTypeInstNetConn()

```
PinType Netlist::getPinTypeInstNetConn (
    IndexType instId,
    IndexType netId ) const
```

Get PinType of a pin such that [Inst](#) and [Net](#) are connected through this pin.

Example: Suppose pin[0] of inst[1] is connected to net[2]. `getPinTypeInstNetConn(1,2)` would return PinType of pin[0]. This function allows us to query for connection types and determine future search directions.

By definition this pin must belong to instId and be connected to netId. If no such pin exists [PinType::OTHER](#) is returned.

#### Parameters

<i>instId</i>	Id of <a href="#">Inst</a> that returned pin is connected.
<i>netId</i>	Id of <a href="#">Net</a> that returned pin is connected.

#### 3.8.3.12 getPinTypeInstPinConn()

```
PinType Netlist::getPinTypeInstPinConn (
    IndexType instId,
    IndexType pinId ) const
```

Get PinType of a pin such that [Inst](#) and [Pin](#) are connected through this pin.

Example: Suppose pin[0] of inst[1] is connected to pin[2] (through some net). `getPinTypeInstPinConn(1,2)` would return PinType of pin[0]. This function allows us to query for connection types and determine future search directions.

By definition this pin must belong to instId and be connected to pinId through some net. If no such pin exists [PinType::OTHER](#) is returned.

## Parameters

<i>inst↔ Id</i>	Id of <a href="#">Inst</a> that returned pin is connected.
<i>pinId</i>	Id of <a href="#">Pin</a> that returned pin is connected.

3.8.3.13 `init()`

```
void Netlist::init (
    InitDataObj & obj )
```

Initialize [Netlist](#) class.

3.8.3.14 `inst()`

```
const Inst& Netlist::inst (
    IndexType id ) const [inline]
```

Return [Inst](#) of Id.

3.8.3.15 `instNetId()`

```
IndexType Netlist::instNetId (
    IndexType instId,
    PinType pinType ) const
```

Return Id of [Net](#) connected to [Inst](#) by certain [PinType](#).

Example: `instNetId(0, PinType::DRAIN)` would return the net index connected to `inst[0]` through a pin which [Pin↔  
Type::DRAIN](#). Or this returns `inst[0]` drain net. If the [Inst](#) does not have a [PinType](#) connected, `INDEX_TYPE_MAX` would be returned. Use at risk and only if [InstType](#) is known.

## Parameters

<i>instId</i>	Id of <a href="#">Inst</a> .
<i>pinType</i>	Returned <a href="#">Net</a> Id connected to this <a href="#">PinType</a> .

3.8.3.16 `instPinId()`

```
IndexType Netlist::instPinId (
```



```

    IndexType instId,
    PinType pinType ) const

```

Return Id of [Pin](#) with PinType connected to [Inst](#).

Example: `instPinId(0,PinType::DRAIN)` would return the pin index connected to `inst[0]` which is [PinType::DRAIN](#). Or this returns `inst[0]` drain pin index. If [Inst](#) does not have a PinType connected, `INDEX_TYPE_MAX` would be returned. Use at risk and only if `InstType` is known.

#### Parameters

<i>instId</i>	Id of <a href="#">Inst</a> .
<i>pinType</i>	Returned <a href="#">Pin</a> Id should be this PinType.

#### 3.8.3.17 isMos()

```

bool Netlist::isMos (
    InstType instType ) const

```

Return true if `InstType` is a Mosfet. NMOS and PMOS are Mosfets.

#### 3.8.3.18 isPasvDev()

```

bool Netlist::isPasvDev (
    InstType instType ) const

```

Return true if `InstType` is passive device. RES and CAP are passive devices.

#### 3.8.3.19 isSignal()

```

bool Netlist::isSignal (
    IndexType netId ) const [inline]

```

Return true if corresponding net `NetType::Signal`.

#### 3.8.3.20 mosType()

```

MosType Netlist::mosType (
    IndexType mosId ) const

```

Return `MosType` of corresponding instance id.

### 3.8.3.21 net()

```
const Net& Netlist::net (
    IndexType id ) const [inline]
```

Return [Net](#) of Id.

### 3.8.3.22 numInst()

```
IndexType Netlist::numInst ( ) const [inline]
```

Return number of [Inst](#).

### 3.8.3.23 numNet()

```
IndexType Netlist::numNet ( ) const [inline]
```

Return number of [Net](#).

### 3.8.3.24 numPin()

```
IndexType Netlist::numPin ( ) const [inline]
```

Return number of [Pin](#).

### 3.8.3.25 pin()

```
const Pin& Netlist::pin (
    IndexType id ) const [inline]
```

Return [Pin](#) of Id.

### 3.8.3.26 print\_all()

```
void Netlist::print_all ( ) const
```

Print netlist.

### 3.8.3.27 rmvInstHasPin()

```
void Netlist::rmvInstHasPin (
    std::vector< IndexType > & instArray,
    IndexType pinId ) const
```

Remove from array, [Inst](#) that has pinId.

O(n) complexity guaranteed. Similar implementation of std::remove().

## Parameters

<i>instArray</i>	Reference to instance Id array.
<i>pinId</i>	Id of pin.

## 3.8.3.28 srcNetId()

```
IndexType Netlist::srcNetId (  
    IndexType mosId ) const [inline]
```

Return Source [Net](#) Id of [Inst](#) mosId. Equivalent as `instNetId(mosId, PinType::SOURCE);`.

## See also

[instNetId](#).

## 3.8.4 Member Data Documentation

## 3.8.4.1 \_instArray

```
std::vector<Inst> Netlist::_instArray [private]
```

## 3.8.4.2 \_netArray

```
std::vector<Net> Netlist::_netArray [private]
```

## 3.8.4.3 \_pinArray

```
std::vector<Pin> Netlist::_pinArray [private]
```

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

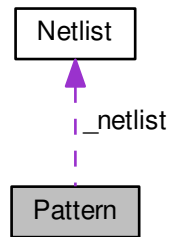
- [src/db/Netlist.h](#)
- [src/db/Netlist.cpp](#)

### 3.9 Pattern Class Reference

[Pattern](#) class.

```
#include <Pattern.h>
```

Collaboration diagram for Pattern:



#### Public Member Functions

- [Pattern](#) (const [Netlist](#) &netlist)  
*Constructor.*
- [MosPattern](#) pattern ([IndexType](#) mosId1, [IndexType](#) mosId2) const  
*Return pattern for pair of mosfets.*

#### Private Member Functions

- bool [matchedType](#) ([IndexType](#) mosId1, [IndexType](#) mosId2) const  
*Return true if [Inst](#) pair have same [InstType](#).*
- bool [matchedSize](#) ([IndexType](#) mosId1, [IndexType](#) mosId2) const  
*Return true if [Inst](#) pair have same size attributes.*
- bool [diffPairInput](#) ([IndexType](#) mosId1, [IndexType](#) mosId2) const  
*Return true if fits [MosPattern::DIFF\\_SOURCE](#).*
- bool [diffPairCascocode](#) ([IndexType](#) mosId1, [IndexType](#) mosId2) const  
*Return true if fits [MosPattern::DIFF\\_CASCADE](#).*
- bool [validPairCascocode](#) ([IndexType](#) mosId1, [IndexType](#) mosId2) const  
*Return true if fits [MosPattern::CASCADE](#).*
- bool [validPairLoad](#) ([IndexType](#) mosId1, [IndexType](#) mosId2) const  
*Return true if fits [MosPattern::LOAD](#).*
- bool [crossPairCascocode](#) ([IndexType](#) mosId1, [IndexType](#) mosId2) const  
*Return true if fits [MosPattern::CROSS\\_CASCADE](#).*
- bool [crossPairLoad](#) ([IndexType](#) mosId1, [IndexType](#) mosId2) const  
*Return true if fits [MosPattern::CROSS\\_LOAD](#).*

## Private Attributes

- const [Netlist](#) & `_netlist`

### 3.9.1 Detailed Description

[Pattern](#) class.

### 3.9.2 Constructor & Destructor Documentation

#### 3.9.2.1 Pattern()

```
Pattern::Pattern (
    const Netlist & netlist ) [inline], [explicit]
```

Constructor.

Parameters

<i>netlist</i>	<a href="#">Netlist</a> for pattern search.
----------------	---

### 3.9.3 Member Function Documentation

#### 3.9.3.1 crossPairCascode()

```
bool Pattern::crossPairCascode (
    IndexType mosId1,
    IndexType mosId2 ) const [private]
```

Return true if fits [MosPattern::CROSS\\_CASCADE](#).

#### 3.9.3.2 crossPairLoad()

```
bool Pattern::crossPairLoad (
    IndexType mosId1,
    IndexType mosId2 ) const [private]
```

Return true if fits [MosPattern::CROSS\\_LOAD](#).

### 3.9.3.3 diffPairCascode()

```
bool Pattern::diffPairCascode (
    IndexType mosId1,
    IndexType mosId2 ) const [private]
```

Return true if fits [MosPattern::DIFF\\_CASCADE](#).

### 3.9.3.4 diffPairInput()

```
bool Pattern::diffPairInput (
    IndexType mosId1,
    IndexType mosId2 ) const [private]
```

Return true if fits [MosPattern::DIFF\\_SOURCE](#).

### 3.9.3.5 matchedSize()

```
bool Pattern::matchedSize (
    IndexType mosId1,
    IndexType mosId2 ) const [private]
```

Return true if [Inst](#) pair have same size attributes.

### 3.9.3.6 matchedType()

```
PROJECT_NAMESPACE_BEGIN bool Pattern::matchedType (
    IndexType mosId1,
    IndexType mosId2 ) const [private]
```

Return true if [Inst](#) pair have same InstType.

### 3.9.3.7 pattern()

```
MosPattern Pattern::pattern (
    IndexType mosId1,
    IndexType mosId2 ) const
```

Return pattern for pair of mosfets.

Valid patterns have same InstType. Currently they also have same size attribute.

TODO Add ratio pair detection in future.

See also

[MosPattern](#).

## Parameters

<i>mosId1</i>	Id for mosfet.
<i>mosId2</i>	Id for mosfet.

## 3.9.3.8 validPairCascode()

```
bool Pattern::validPairCascode (
    IndexType mosId1,
    IndexType mosId2 ) const [private]
```

Return true if fits [MosPattern::CASCODE](#).

## 3.9.3.9 validPairLoad()

```
bool Pattern::validPairLoad (
    IndexType mosId1,
    IndexType mosId2 ) const [private]
```

Return true if fits [MosPattern::LOAD](#).

## 3.9.4 Member Data Documentation

## 3.9.4.1 \_netlist

```
const Netlist& Pattern::_netlist [private]
```

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

- src/sym\_detect/[Pattern.h](#)
- src/sym\_detect/[Pattern.cpp](#)

## 3.10 Pin Class Reference

[Pin](#) class.

```
#include <Pin.h>
```

## Public Member Functions

- [Pin](#) ()=default
- [Pin](#) ([IndexType](#) id, [IndexType](#) instId, [IndexType](#) netId, [PinType](#) type)  
*Constructor for [Pin](#).*
- [IndexType](#) id () const
- [IndexType](#) instId () const
- [IndexType](#) netId () const
- [PinType](#) type () const  
*Return type of [Pin](#).*

## Static Public Member Functions

- static [PinType](#) nextPinType ([PinType](#) type)  
*Return the next search PinType for DFS.*
- static bool isPasvDev ([PinType](#) type)

## Private Attributes

- [IndexType](#) \_id
- [IndexType](#) \_instId
- [IndexType](#) \_netId
- [PinType](#) \_type

### 3.10.1 Detailed Description

[Pin](#) class.

### 3.10.2 Constructor & Destructor Documentation

#### 3.10.2.1 [Pin\(\)](#) [1/2]

```
Pin::Pin ( ) [explicit], [default]
```

Default Constructor.

#### 3.10.2.2 [Pin\(\)](#) [2/2]

```
Pin::Pin (
    IndexType id,
    IndexType instId,
    IndexType netId,
    PinType type ) [inline], [explicit]
```

Constructor for [Pin](#).



## Parameters

<i>id</i>	Id of <a href="#">Pin</a> .
<i>inst↔ Id</i>	Id of connected <a href="#">Inst</a> .
<i>netId</i>	Id of connected <a href="#">Net</a> .
<i>type</i>	Type of <a href="#">Pin</a> .

## 3.10.3 Member Function Documentation

## 3.10.3.1 id()

```
IndexType Pin::id ( ) const [inline]
```

Return id of [Pin](#).

## 3.10.3.2 instId()

```
IndexType Pin::instId ( ) const [inline]
```

Return id of connected [Inst](#).

## 3.10.3.3 isPasvDev()

```
bool Pin::isPasvDev (
    PinType type ) [static]
```

Return true if [PinType](#) belongs to a passive device.

A pin is said to belong to a passive device if the [PinType](#) is [PinType::THIS](#) or [PinType::THAT](#).

## 3.10.3.4 netId()

```
IndexType Pin::netId ( ) const [inline]
```

Return id of connected [Net](#).

## 3.10.3.5 nextPinType()

```
PROJECT\_NAMESPACE\_BEGIN PinType Pin::nextPinType (
    PinType type ) [static]
```

Return the next search [PinType](#) for DFS.

## Parameters

<i>type</i>	Query the next search PinType.
-------------	--------------------------------

## See also

[PinType](#)

The DFS search for symmetry relies on [Pin::nextPinType](#) to define the search path direction. For example, if a Mosfet was reached through a source then the DFS algorithm would search for connected [Inst](#) of the drain. Currently supported search paths:

Input PinType	nextPinType
SOURCE	DRAIN
DRAIN	SOURCE
THIS	THAT
THAT	THIS

3.10.3.6 `type()`

```
PinType Pin::type ( ) const [inline]
```

Return type of [Pin](#).

## See also

[PinType](#)

## 3.10.4 Member Data Documentation

3.10.4.1 `_id`

```
IndexType Pin::_id [private]
```

3.10.4.2 `_instId`

```
IndexType Pin::_instId [private]
```

3.10.4.3 `_netId`

```
IndexType Pin::_netId [private]
```

3.10.4.4 `_type`

```
PinType Pin::_type [private]
```

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

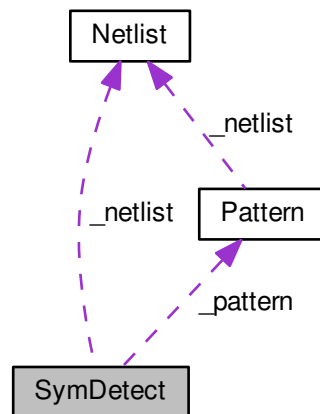
- [src/db/Pin.h](#)
- [src/db/Pin.cpp](#)

## 3.11 SymDetect Class Reference

[SymDetect](#) class.

```
#include <SymDetect.h>
```

Collaboration diagram for SymDetect:



### Public Member Functions

- [SymDetect](#) (const [Netlist](#) &netlist)  
*Constructor Only needs netlist as input. [Pattern](#) class inherently constructed.*
- void [hiSymDetect](#) (std::vector< std::vector< [MosPair](#) >> &symGroup) const  
*Hierarchy symmetry detection.*

## Private Member Functions

- [MosPattern](#) [MosPairPtrn](#) ([MosPair](#) &obj) const  
*Return pattern of [MosPair](#).*
- bool [existPair](#) (std::vector< [MosPair](#) > &library, [IndexType](#) instld1, [IndexType](#) instld2) const  
*bool endSrch([IndexType](#) mosld, [PinType](#) pinType) const;*
- bool [existPair](#) (std::vector< [MosPair](#) > &library, [IndexType](#) instld) const
- bool [endSrch](#) ([MosPair](#) &obj) const  
*Check if pair already reached.*
- bool [validSrchObj](#) ([IndexType](#) instld1, [IndexType](#) instld2, [IndexType](#) srchPinld1, [IndexType](#) srchPinld2) const  
*Return true if a valid pair.*
- bool [validDiffPair](#) ([IndexType](#) instld1, [IndexType](#) instld2, [IndexType](#) srchPinld1, [IndexType](#) srchPinld2) const  
*Return true if a valid DIFF\_SOURCE gate connected.*
- void [pushNextSrchObj](#) (std::vector< [MosPair](#) > &dfsVstPair, std::vector< [MosPair](#) > &dfsStack, [MosPair](#) &currObj, std::vector< [MosPair](#) > &diffPairSrc) const  
*Push next valid [MosPair](#) to dfsStack.*
- void [getPatrnNetConn](#) (std::vector< [MosPair](#) > &diffPair, [IndexType](#) netld, [MosPattern](#) srchPatrn) const  
*Get srchPatrn [MosPair](#) connected to netld.*
- void [getDiffPair](#) (std::vector< [MosPair](#) > &diffPair) const  
*Get valid DFS source of netlist.*
- void [dfsDiffPair](#) (std::vector< [MosPair](#) > &dfsVstPair, [MosPair](#) &diffPair, std::vector< [MosPair](#) > &diffPair↔Srch) const  
*DFS search with given source. Visited [MosPair](#) are stored.*
- void [inVldDiffPairSrch](#) (std::vector< [MosPair](#) > &diffPairSrch, [MosPair](#) &currPair) const  
*Invalidate visited pairs from sources.*
- void [getVldDrainMos](#) (std::vector< [IndexType](#) > &vldMos, [IndexType](#) netld) const  
*Get valid drain connected mosfet to netld.*
- void [selfSymSrch](#) (std::vector< [MosPair](#) > &dfsVstPair, [MosPair](#) &diffPair) const  
*Iteratively search for self symmetry given diffPair.*
- void [addSelfSym](#) (std::vector< [MosPair](#) > &dfsVstPair) const  
*Top function to call to add self symmetry to already searched symmetry group.*

## Private Attributes

- const [Netlist](#) & [\\_netlist](#)
- [Pattern](#) [\\_pattern](#)

### 3.11.1 Detailed Description

[SymDetect](#) class.

### 3.11.2 Constructor & Destructor Documentation

#### 3.11.2.1 [SymDetect](#)()

```
SymDetect::SymDetect (
    const Netlist & netlist ) [inline], [explicit]
```

Constructor Only needs netlist as input. [Pattern](#) class inherently constructed.

## Parameters

<i>netlist</i>	<a href="#">Netlist</a> class.
----------------	--------------------------------

## 3.11.3 Member Function Documentation

## 3.11.3.1 addSelfSym()

```
void SymDetect::addSelfSym (
    std::vector< MosPair > & dfsVstPair ) const [private]
```

Top function to call to add self symmetry to already searched symmetry group.

Iteratively searches for self symmetry instances for [MosPattern::DIFF\\_SOURCE](#) pairs in dfsVstPair. Valid self symmetry instances will be appended. This function is called at the end of every DFS search for symmetry pairs.

## Parameters

<i>dfsVstPair</i>	Symmetry group.
-------------------	-----------------

## See also

[selfSymSrch](#)  
[hiSymDetect](#)

## 3.11.3.2 dfsDiffPair()

```
void SymDetect::dfsDiffPair (
    std::vector< MosPair > & dfsVstPair,
    MosPair & diffPair,
    std::vector< MosPair > & diffPairSrch ) const [private]
```

DFS search with given source. Visited [MosPair](#) are stored.

Search for symmetry patterns in DFS manner with search source as diffPair. Store visited valid [MosPair](#) at dfsVstPair. diffPairSrch are needed as input to invalidate reached sources. dfsVstPair would be in the same hierarchy symmetry group.

## See also

[pushNextSrchObj](#)

## Parameters

out	<i>dfsVstPair</i>	Vector to store all visited <a href="#">MosPair</a>
in	<i>diffPair</i>	DFS search source
in	<i>diffPairSrch</i>	Vector of all stored DFS search source

## 3.11.3.3 endSrch()

```
bool SymDetect::endSrch (
    MosPair & obj ) const [private]
```

Check if pair already reached.

Return true if end of search path.

Current end search terminations: (1) Connected PASSIVE (2) DIFF\_SOURCE reached through DRAIN (3) LOAD, CROSS\_LOAD (4) gate connected pairs

## 3.11.3.4 existPair() [1/2]

```
bool SymDetect::existPair (
    std::vector< MosPair > & library,
    IndexType instId1,
    IndexType instId2 ) const [private]
```

```
bool endSrch(IndexType mosId, PinType pinType) const;
```

Check if pair already reached.

## 3.11.3.5 existPair() [2/2]

```
bool SymDetect::existPair (
    std::vector< MosPair > & library,
    IndexType instId ) const [private]
```

Check if self symmetry pair already reached.

## 3.11.3.6 getDiffPair()

```
void SymDetect::getDiffPair (
    std::vector< MosPair > & diffPair ) const [private]
```

Get valid DFS source of netlist.

Iterate all signal nets for getPatrnNetConn. Commonly srchPatrn are DIFF\_SOURCE and CROSS\_LOAD. This would return all DFS sources.

See also

[getDiffPairNetConn](#)

## Parameters

<i>diffPair</i>	Store the output vector
-----------------	-------------------------

## 3.11.3.7 getPatrnNetConn()

```
PROJECT_NAMESPACE_BEGIN void SymDetect::getPatrnNetConn (
    std::vector< MosPair > & diffPair,
    IndexType netId,
    MosPattern srchPatrn ) const [private]
```

Get srchPatrn [MosPair](#) connected to netId.

Find [MosPair](#) that follow srchPatrn. These [MosPair](#) are appended to diffPair. Used to get valid DFS source. srch↔Patrn inputs commonly are DIFF\_SOURCE and CROSS\_LOAD. Currently pairs should follow: (1) Have MosPattern srchPatrn (2) source connected to netId (3) [MosType::DIFF](#)

## Parameters

<i>netId</i>	Source should be connected to netId.
<i>diffPair</i>	Stored output vector.

## 3.11.3.8 getVldDrainMos()

```
void SymDetect::getVldDrainMos (
    std::vector< IndexType > & vldMos,
    IndexType netId ) const [private]
```

Get valid drain connected mosfet to netId.

Valid Mosfets must be connected to netId through [PinType::DRAIN](#), it should also have [MosType::DIFF](#). This is used to search self symmetric pairs connected to [MosPattern::DIFF\\_SOURCE](#).

## Parameters

<i>vldMos</i>	Vector to store valid Mosfet.
<i>netId</i>	Id of connected net.

## 3.11.3.9 hiSymDetect()

```
void SymDetect::hiSymDetect (
    std::vector< std::vector< MosPair >> & symGroup ) const
```

Hierarchy symmetry detection.

Output would contain 2 levels of hierarchy. symGroup is a vector of `std::vector<MosPair>` oneGroup. Where one↔Group is a group of [MosPair](#) in the same symmetry group. Each [MosPair](#) should follow a MosPattern, or it should be of self symmetry. This funtion has been also updated to contain basic passive pair symmetry.

#### Parameters

<i>symGroup</i>	Detected symmetry groups of netlist.
-----------------	--------------------------------------

#### See also

[MosPattern](#)

[MosPair](#)

#### 3.11.3.10 inVldDiffPairSrch()

```
void SymDetect::inVldDiffPairSrch (
    std::vector< MosPair > & diffPairSrch,
    MosPair & currPair ) const [private]
```

Invalidate visited pairs from sources.

If a [MosPair](#) have already been visited and is a DFS source, it should be invalidated as a DFS search source to avoid revisiting.

#### Parameters

<i>diffPairSrch</i>	Vector of all DFS sources.
<i>currPair</i>	<a href="#">MosPair</a> to invalidate.

#### 3.11.3.11 MosPairPtrn()

```
MosPattern SymDetect::MosPairPtrn (
    MosPair & obj ) const [private]
```

Return pattern of [MosPair](#).

#### 3.11.3.12 pushNextSrchObj()

```
void SymDetect::pushNextSrchObj (
    std::vector< MosPair > & dfsVstPair,
    std::vector< MosPair > & dfsStack,
```



```

    MosPair & currObj,
    std::vector< MosPair > & diffPairSrc ) const [private]

```

Push next valid [MosPair](#) to dfsStack.

This function push valid pairs that could be reached from currObj to dfsStack. It also removes reached DIFF\_SOURCE [MosPair](#) from diffPairSrc. A pair is valid either a valid load or a valid second stage input DIFF\_SOURCE.

See also

[inVldDiffPairSrch](#)  
[validSrchObj](#)  
[validDiffPair](#)

#### Parameters

<i>dfsVstPair</i>	All current visited <a href="#">MosPair</a>
<i>dfsStack</i>	Stack to store to visit <a href="#">MosPair</a>
<i>currObj</i>	Current <a href="#">MosPair</a> under visit
<i>diffPairSrc</i>	All DFS sources

#### 3.11.3.13 selfSymSrch()

```

void SymDetect::selfSymSrch (
    std::vector< MosPair > & dfsVstPair,
    MosPair & diffPair ) const [private]

```

Iteratively search for self symmetry given diffPair.

diffPair should be of [MosPattern::DIFF\\_SOURCE](#). Valid self symmetric instances are added to dfsVstPair. Redundancy is also removed from dfsVstPair.

#### Parameters

<i>dfsVstPair</i>	Self symmetric pairs will be added to this vector.
<i>diffPair</i>	<a href="#">MosPattern::DIFF_SOURCE</a> pair to begin self symmetry search.

See also

[getVldDrainMos](#)

#### 3.11.3.14 validDiffPair()

```

bool SymDetect::validDiffPair (
    IndexType instId1,

```

```

IndexType instId2,
IndexType srchPinId1,
IndexType srchPinId2 ) const [private]

```

Return true if a valid DIFF\_SOURCE gate connected.

This funtion is used to expand symmetry groups through DRAIN to GATE connections like searching for 2 stage OTAs. Since validSrchObj funtion blocks all gate connections, this funtion is used to check for DIFF\_SOURCE second stage "input" pairs.

Valid pairs have following attributes: (1) Reached through gate (2) DIFF\_SOURCE pattern type.

See also

[validSrchObj](#)

#### Parameters

<i>instId1</i>	Reached pair instId1
<i>instId2</i>	Reached pair instId2
<i>srchPinId1</i>	instId1 reached by srchPinId1.
<i>srchPinId2</i>	instId2 reached by srchPinId2.

#### 3.11.3.15 validSrchObj()

```

bool SymDetect::validSrchObj (
    IndexType instId1,
    IndexType instId2,
    IndexType srchPinId1,
    IndexType srchPinId2 ) const [private]

```

Return true if a valid pair.

Valid pairs have following attributes: (1) Any mosfet pairs not reached by PASSIVE (2) Reached through same PinType (3) Not reached through gate (4) Valid MosPattern

#### Parameters

<i>instId1</i>	Reached pair instId1
<i>instId2</i>	Reached pair instId2
<i>srchPinId1</i>	instId1 reached by srchPinId1.
<i>srchPinId2</i>	instId2 reached by srchPinId2.

### 3.11.4 Member Data Documentation

#### 3.11.4.1 `_netlist`

```
const Netlist& SymDetect::_netlist [private]
```

#### 3.11.4.2 `_pattern`

```
Pattern SymDetect::_pattern [private]
```

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

- [src/sym\\_detect/SymDetect.h](#)
- [src/sym\\_detect/SymDetect.cpp](#)



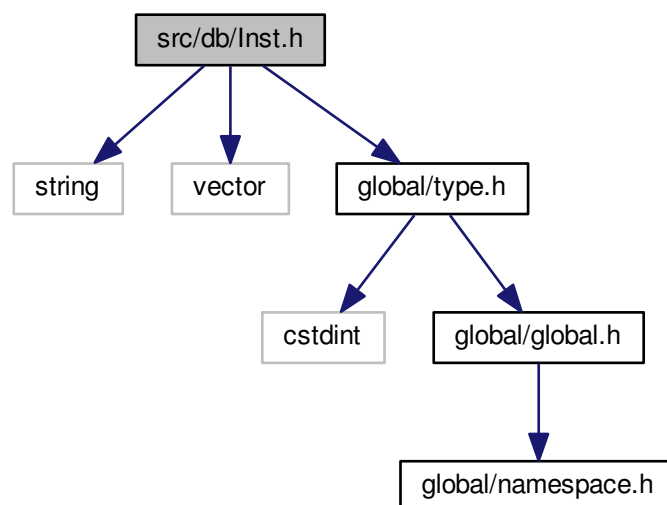
## Chapter 4

# File Documentation

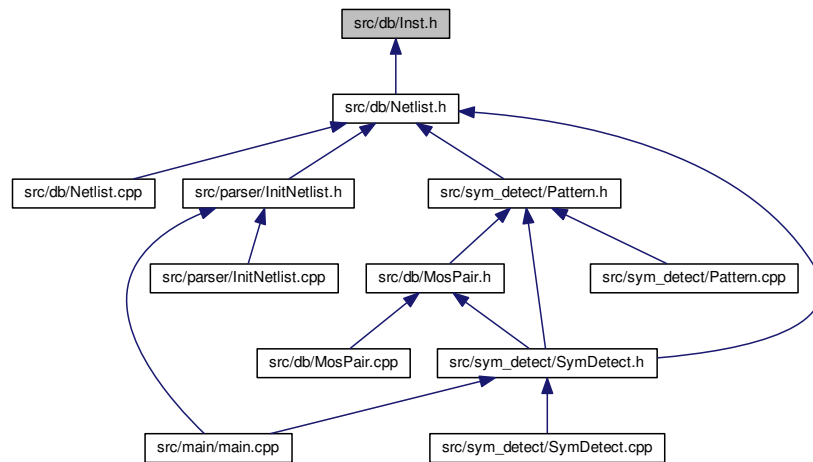
### 4.1 src/db/Inst.h File Reference

Instance class.

```
#include <string>
#include <vector>
#include "global/type.h"
Include dependency graph for Inst.h:
```



This graph shows which files directly or indirectly include this file:



## Classes

- class [Inst](#)  
*Inst* class.

### 4.1.1 Detailed Description

Instance class.

Author

Mingjie Liu

Date

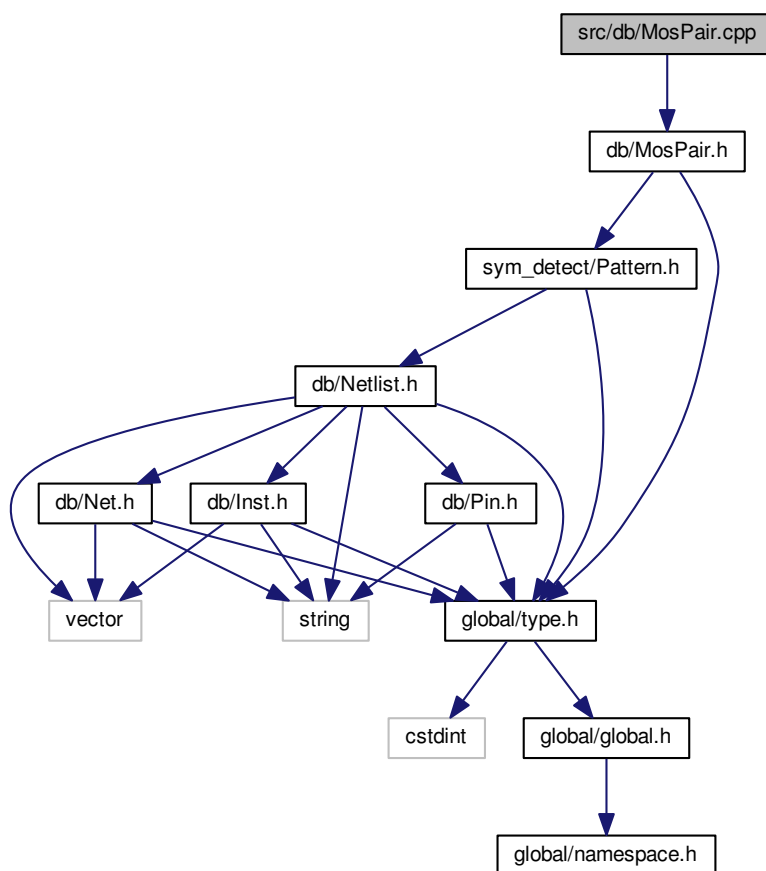
11/24/2018

## 4.2 src/db/MosPair.cpp File Reference

[MosPair](#) implementation.

```
#include "db/MosPair.h"
```

Include dependency graph for MosPair.cpp:



#### 4.2.1 Detailed Description

[MosPair](#) implementation.

Author

Mingjie Liu

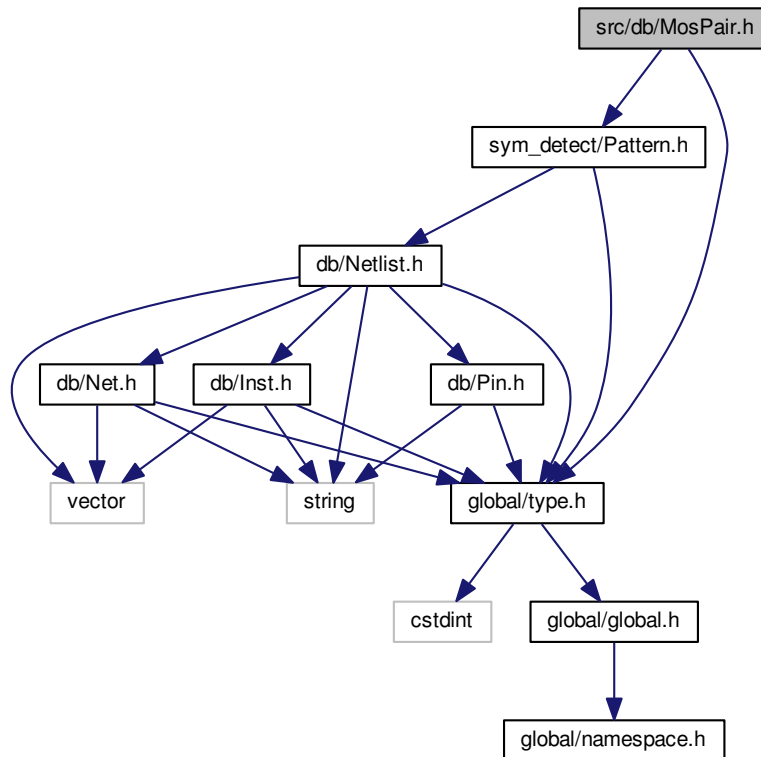
Date

11/27/2018

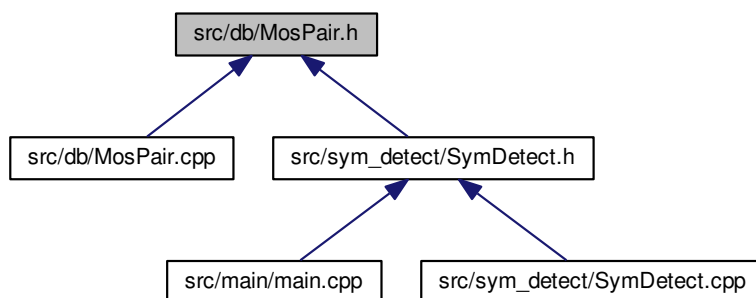
### 4.3 src/db/MosPair.h File Reference

A pair of Mosfet with MosPattern.

```
#include "global/type.h"
#include "sym_detect/Pattern.h"
Include dependency graph for MosPair.h:
```



This graph shows which files directly or indirectly include this file:



## Classes

- class [MosPair](#)

*A pair of Mosfet with MosPattern.*



### 4.3.1 Detailed Description

A pair of Mosfet with MosPattern.

Author

Mingjie Liu

Date

11/27/2018

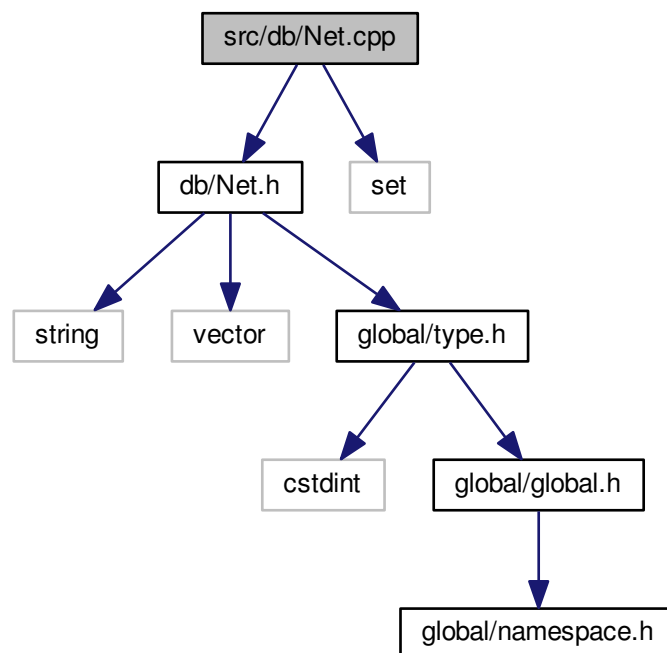
## 4.4 src/db/Net.cpp File Reference

[Net](#) class implementation.

```
#include "db/Net.h"
```

```
#include <set>
```

Include dependency graph for Net.cpp:



### Variables

- static `PROJECT_NAMESPACE_BEGIN` const std::set< std::string > `POWER_NET_NAMES` = {"vdd", "VDD", "Vdd", "VDDA", "vdda", "Vdda"}
- static const std::set< std::string > `GROUND_NET_NAMES` = {"vss", "VSS", "Vss", "VSSA", "vssa", "Vssa", "gnd", "Gnd", "GND"}

### 4.4.1 Detailed Description

[Net](#) class implementation.

Author

Mingjie Liu

Date

11/24/2018

### 4.4.2 Variable Documentation

#### 4.4.2.1 GROUND\_NET\_NAMES

```
const std::set<std::string> GROUND_NET_NAMES = {"vss", "VSS", "Vss", "VSSA", "vssa", "Vssa",  
"gnd", "Gnd", "GND"} [static]
```

A set of possible ground net names.

#### 4.4.2.2 POWER\_NET\_NAMES

```
PROJECT_NAMESPACE_BEGIN const std::set<std::string> POWER_NET_NAMES = {"vdd", "VDD", "vdd",  
"VDDA", "vdda", "Vdda"} [static]
```

A set of possible power net names.

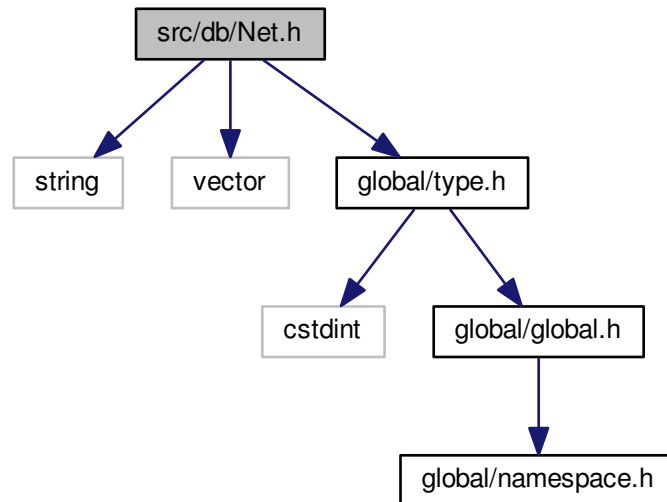
## 4.5 src/db/Net.h File Reference

[Net](#) class.

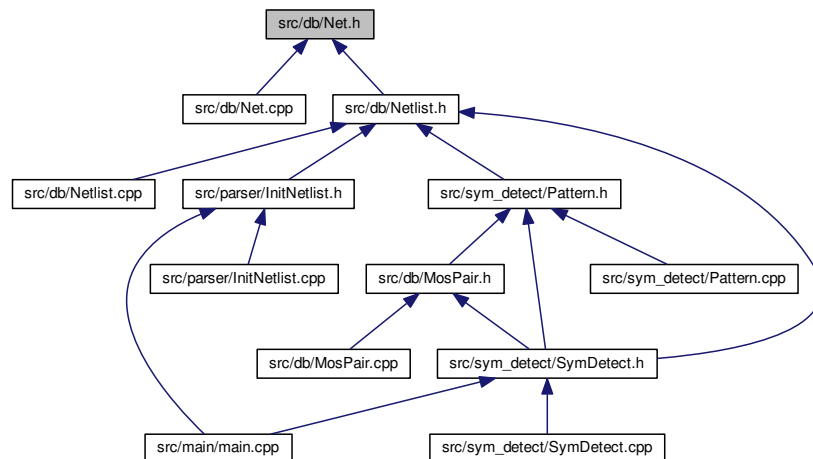
```
#include <string>  
#include <vector>
```

```
#include "global/type.h"
```

Include dependency graph for Net.h:



This graph shows which files directly or indirectly include this file:



## Classes

- class [Net](#)  
*Net class.*

### 4.5.1 Detailed Description

[Net](#) class.

Author

Mingjie Llu

Date

11/24/2018

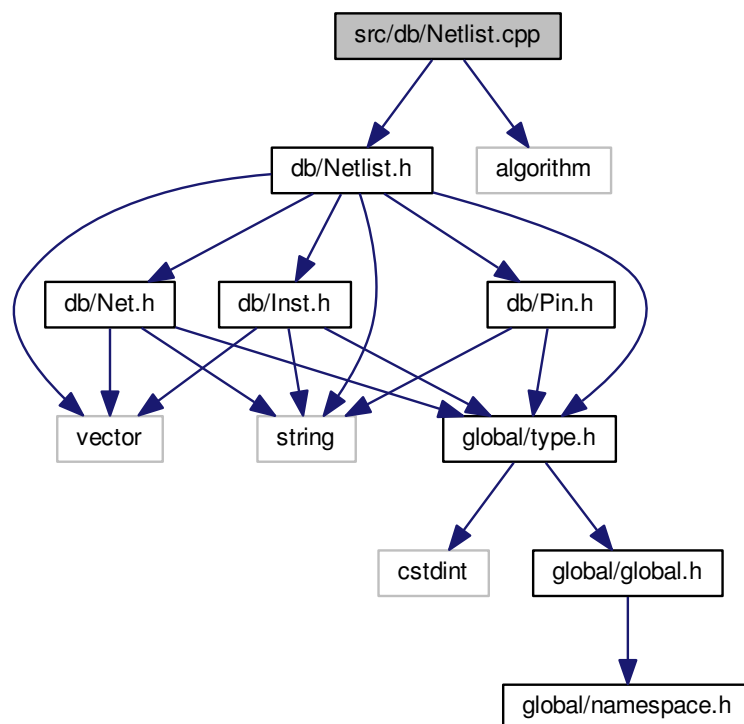
## 4.6 src/db/Netlist.cpp File Reference

[Netlist](#) class implementation.

```
#include "db/Netlist.h"
```

```
#include <algorithm>
```

Include dependency graph for Netlist.cpp:



## Variables

- static `PROJECT_NAMESPACE_BEGIN` const `PinType` `MOS_PIN_TYPE` [4] = {`PinType::DRAIN`, `PinType::GATE`, `PinType::SOURCE`, `PinType::BULK`}  
*Mos Pin Types.*
- static const `PinType` `RES_PIN_TYPE` [3] = {`PinType::THIS`, `PinType::THAT`, `PinType::OTHER`}  
*Res/Cap Pin Types.*

### 4.6.1 Detailed Description

`Netlist` class implementation.

#### Author

Mingjie Liu

#### Date

11/24/2018

### 4.6.2 Variable Documentation

#### 4.6.2.1 MOS\_PIN\_TYPE

```
PROJECT_NAMESPACE_BEGIN const PinType MOS_PIN_TYPE[4] = {PinType::DRAIN, PinType::GATE, PinType::SOURCE, PinType::BULK} [static]
```

Mos `Pin` Types.

#### 4.6.2.2 RES\_PIN\_TYPE

```
const PinType RES_PIN_TYPE[3] = {PinType::THIS, PinType::THAT, PinType::OTHER} [static]
```

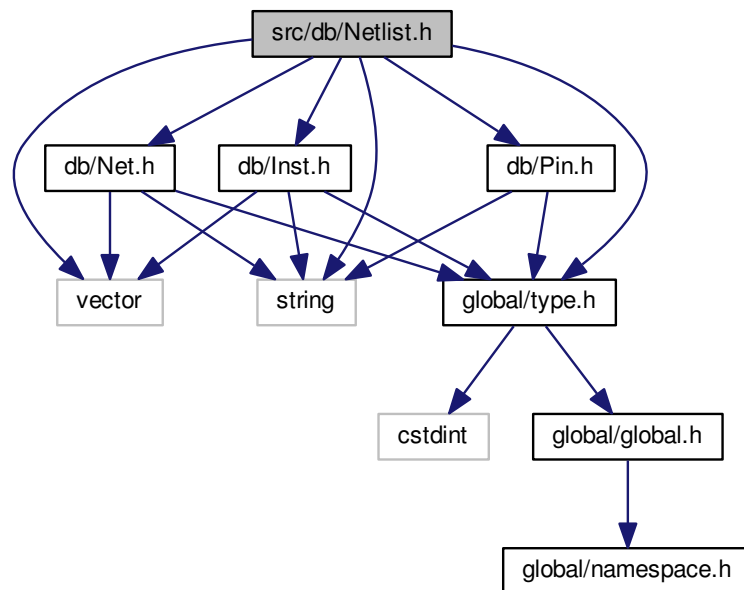
Res/Cap `Pin` Types.

## 4.7 src/db/Netlist.h File Reference

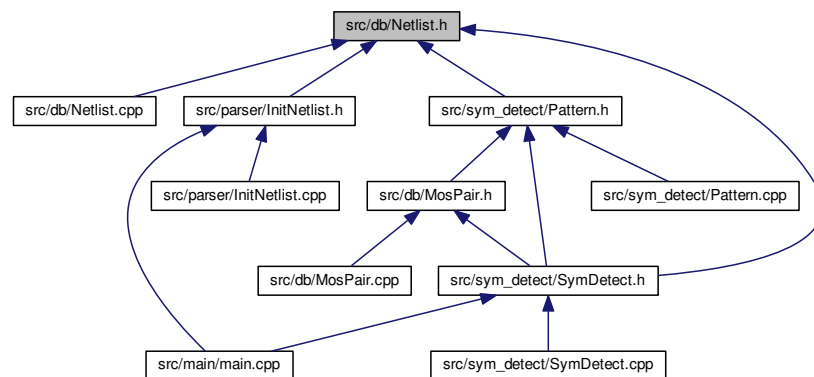
[Netlist](#) class.

```
#include <vector>
#include <string>
#include "global/type.h"
#include "db/Net.h"
#include "db/Pin.h"
#include "db/Inst.h"
```

Include dependency graph for Netlist.h:



This graph shows which files directly or indirectly include this file:



## Classes

- class [Netlist](#)  
*Netlist* class.
- struct [Netlist::InitNet](#)  
*Net* for instantiation.
- struct [Netlist::InitInst](#)  
*Inst* for instantiation.
- struct [Netlist::InitDataObj](#)  
Instantiate *Netlist* class.

### 4.7.1 Detailed Description

[Netlist](#) class.

#### Author

Mingjie Liu

#### Date

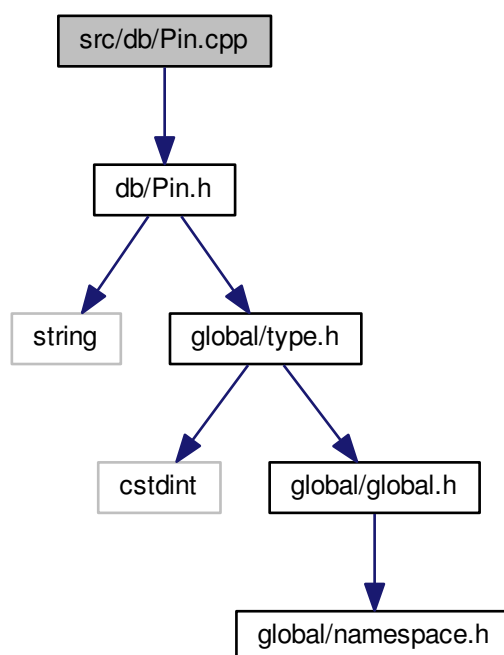
11/24/2018

## 4.8 src/db/Pin.cpp File Reference

[Net](#) class implementation.

```
#include "db/Pin.h"
```

Include dependency graph for Pin.cpp:



### 4.8.1 Detailed Description

[Net](#) class implementation.

Author

Mingjie Liu

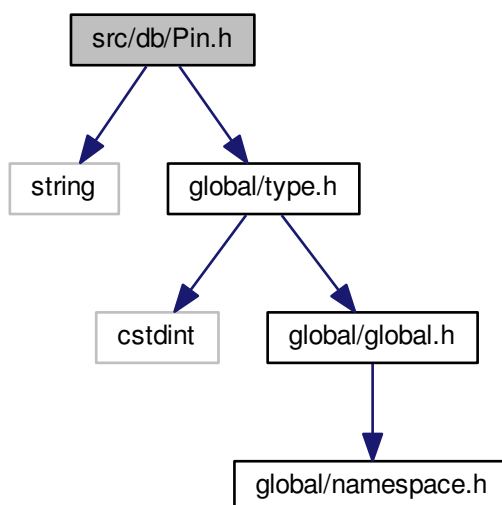
Date

11/24/2018

## 4.9 src/db/Pin.h File Reference

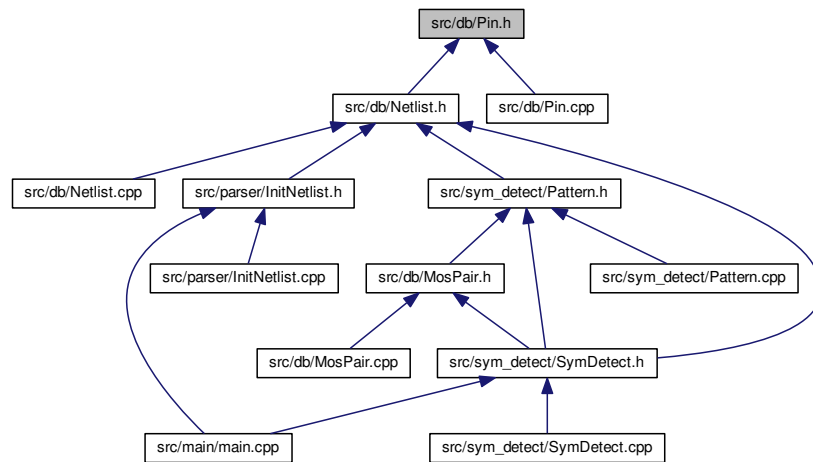
[Pin](#) class.

```
#include <string>
#include "global/type.h"
Include dependency graph for Pin.h:
```





This graph shows which files directly or indirectly include this file:



## Classes

- class [Pin](#)  
*Pin* class.

### 4.9.1 Detailed Description

[Pin](#) class.

#### Author

Mingjie Liu

#### Date

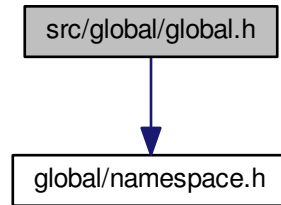
11/24/2018

## 4.10 src/global/global.h File Reference

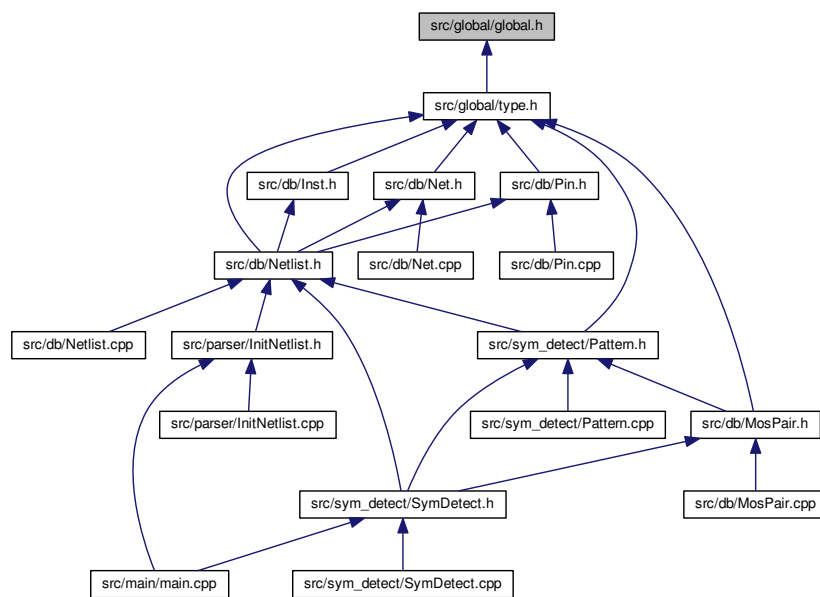
Global header file.

```
#include "global/namespace.h"
```

Include dependency graph for global.h:



This graph shows which files directly or indirectly include this file:



#### 4.10.1 Detailed Description

Global header file.

Author

Mingjie Liu

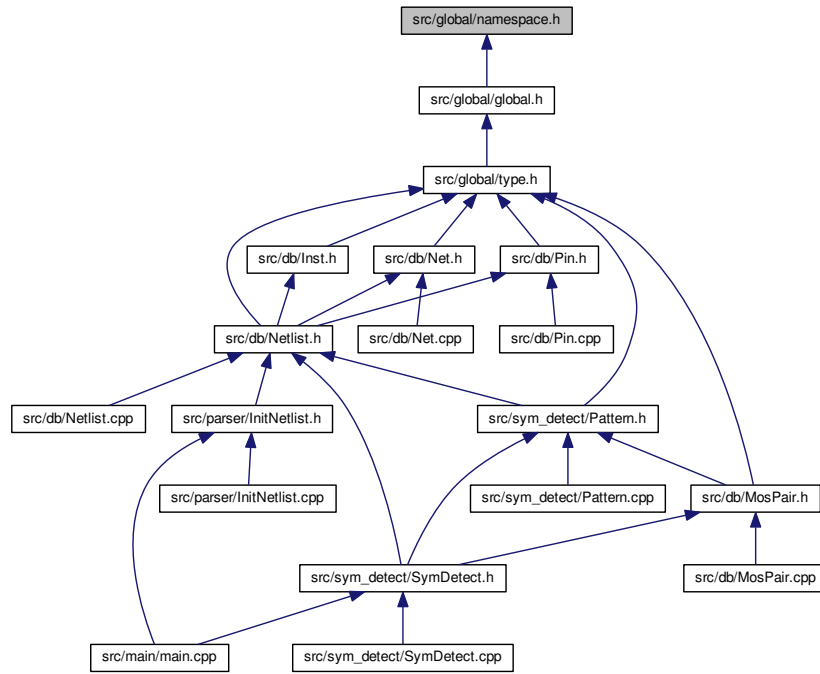
Date

11/24/2018

## 4.11 src/global/namespace.h File Reference

Namespace header file.

This graph shows which files directly or indirectly include this file:



### Macros

- `#define PROJECT_NAMESPACE SFA`
- `#define PROJECT_NAMESPACE_BEGIN namespace PROJECT_NAMESPACE {`
- `#define PROJECT_NAMESPACE_END }`

#### 4.11.1 Detailed Description

Namespace header file.

Author

Mingjie Liu

Date

11/24/2018

#### 4.11.2 Macro Definition Documentation

#### 4.11.2.1 PROJECT\_NAMESPACE

```
#define PROJECT_NAMESPACE SFA
```

#### 4.11.2.2 PROJECT\_NAMESPACE\_BEGIN

```
#define PROJECT_NAMESPACE_BEGIN namespace PROJECT_NAMESPACE {
```

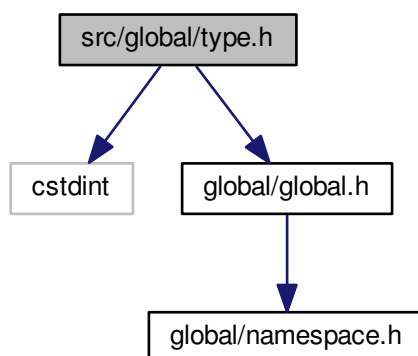
#### 4.11.2.3 PROJECT\_NAMESPACE\_END

```
#define PROJECT_NAMESPACE_END }
```

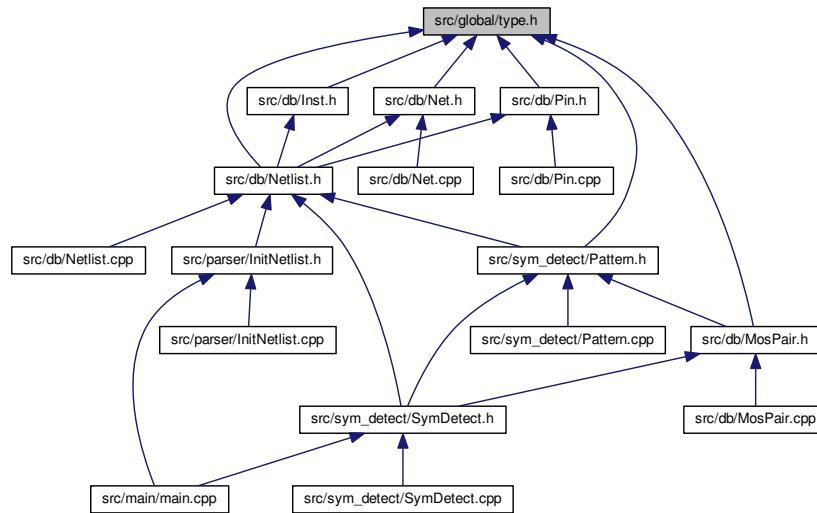
### 4.12 src/global/type.h File Reference

Type header file.

```
#include <stdint>
#include "global/global.h"
Include dependency graph for type.h:
```



This graph shows which files directly or indirectly include this file:



## Typedefs

- using `IndexType` = `std::uint32_t`
- using `IntType` = `std::int32_t`
- using `RealType` = `double`
- using `Byte` = `std::uint8_t`

## Enumerations

- enum `InstType` : `Byte` {  
`InstType::RES`, `InstType::PMOS`, `InstType::NMOS`, `InstType::CAP`,  
`InstType::OTHER` }  
*Type of `Inst`.*
- enum `NetType` : `Byte` { `NetType::POWER`, `NetType::GROUND`, `NetType::SIGNAL` }  
*Type of `Net`.*
- enum `PinType` : `Byte` {  
`PinType::SOURCE`, `PinType::DRAIN`, `PinType::GATE`, `PinType::BULK`,  
`PinType::THIS`, `PinType::THAT`, `PinType::OTHER` }  
*Type of `Pin`.*
- enum `MosType` : `Byte` { `MosType::DIFF`, `MosType::DIODE`, `MosType::CAP`, `MosType::DUMMY` }  
*Connection type of `Mosfet`.*
- enum `MosPattern` : `Byte` {  
`MosPattern::DIFF_SOURCE`, `MosPattern::DIFF_CASCADE`, `MosPattern::CASCADE`, `MosPattern::LOAD`,  
`MosPattern::CROSS_CASCADE`, `MosPattern::CROSS_LOAD`, `MosPattern::PASSIVE`, `MosPattern::SELF`,  
`MosPattern::INVALID` }  
*Pattern for pair of `Mosfet`.*

## Variables

- constexpr [IndexType](#) INDEX\_TYPE\_MAX = 1000000000
- constexpr [IntType](#) INT\_TYPE\_MAX = 1000000000
- constexpr [IntType](#) INT\_TYPE\_MIN = -1000000000
- constexpr [RealType](#) REAL\_TYPE\_MAX = 1e100
- constexpr [RealType](#) REAL\_TYPE\_MIN = -1e100
- constexpr [RealType](#) REAL\_TYPE\_TOL = 1e-6

### 4.12.1 Detailed Description

Type header file.

Author

Mingjie Liu

Date

11/24/2018

### 4.12.2 Typedef Documentation

#### 4.12.2.1 Byte

```
using Byte = std::uint8_t
```

#### 4.12.2.2 IndexType

```
using IndexType = std::uint32_t
```

#### 4.12.2.3 IntType

```
using IntType = std::int32_t
```

#### 4.12.2.4 RealType

```
using RealType = double
```

### 4.12.3 Enumeration Type Documentation

#### 4.12.3.1 InstType

```
enum InstType : Byte [strong]
```

Type of [Inst](#).

## Enumerator

RES	Resistor
PMOS	PMos
NMOS	NMos
CAP	Capacitor
OTHER	Other

## 4.12.3.2 MosPattern

```
enum MosPattern : Byte [strong]
```

[Pattern](#) for pair of Mosfet.

The patterns have been augmented to also handle self symmetry pairs and passive devices. The name retains as legacy.

## See also

[Pattern::pattern\(\)](#)

## Enumerator

DIFF_SOURCE	Source connected diff pair.
DIFF_CASCODE	Cascode diff pair.
CASCODE	Gate connected cascode pair.
LOAD	Cascode pair with source connected to Power/Ground.
CROSS_CASCODE	Cross coupled cascode pair.
CROSS_LOAD	Cross coupled load.
PASSIVE	Matched passive device.
SELF	Self symmetry <a href="#">Inst.</a>
INVALID	No pattern detected.

## 4.12.3.3 MosType

```
enum MosType : Byte [strong]
```

Connection type of Mosfet.

## See also

[Netlist::mosType\(\)](#).

**Enumerator**

DIFF	D/G/S diff
DIODE	G/D connected
CAP	G/S connected
DUMMY	D/S connected

**4.12.3.4 NetType**

```
enum NetType : Byte [strong]
```

Type of [Net](#).

**Enumerator**

POWER	Power
GROUND	Ground
SIGNAL	Signal

**4.12.3.5 PinType**

```
enum PinType : Byte [strong]
```

Type of [Pin](#).

**Enumerator**

SOURCE	<a href="#">Inst</a> is Mosfet
DRAIN	<a href="#">Inst</a> is Mosfet
GATE	<a href="#">Inst</a> is Mosfet
BULK	<a href="#">Inst</a> is Mosfet
THIS	<a href="#">Inst</a> is Passive
THAT	<a href="#">Inst</a> is Passive
OTHER	Other

**4.12.4 Variable Documentation****4.12.4.1 INDEX\_TYPE\_MAX**

```
constexpr IndexType INDEX_TYPE_MAX = 1000000000
```



#### 4.12.4.2 INT\_TYPE\_MAX

```
constexpr IntType INT_TYPE_MAX = 1000000000
```

#### 4.12.4.3 INT\_TYPE\_MIN

```
constexpr IntType INT_TYPE_MIN = -1000000000
```

#### 4.12.4.4 REAL\_TYPE\_MAX

```
constexpr RealType REAL_TYPE_MAX = 1e100
```

#### 4.12.4.5 REAL\_TYPE\_MIN

```
constexpr RealType REAL_TYPE_MIN = -1e100
```

#### 4.12.4.6 REAL\_TYPE\_TOL

```
constexpr RealType REAL_TYPE_TOL = 1e-6
```

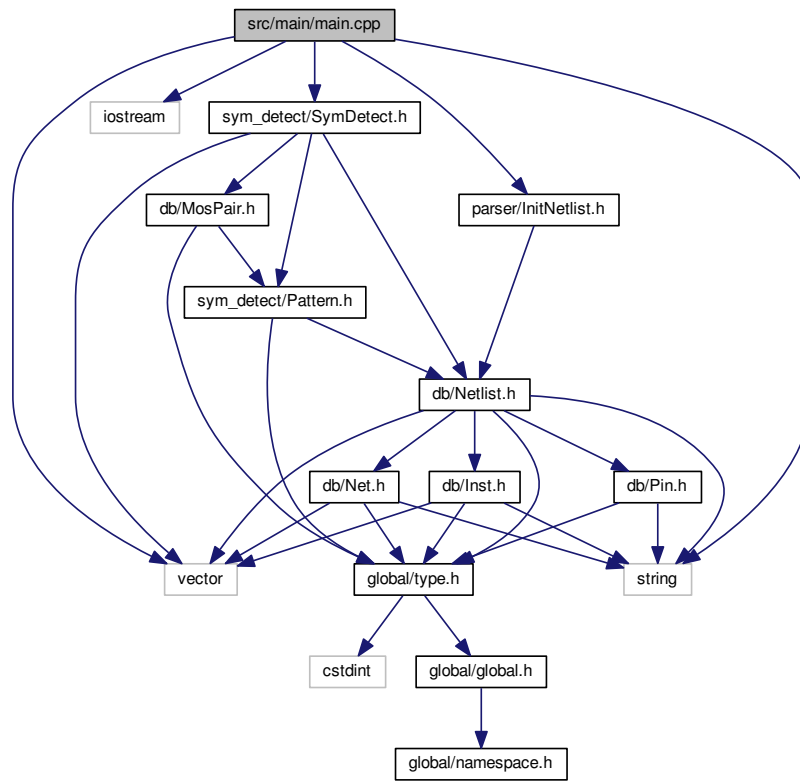
## 4.13 src/main/main.cpp File Reference

[main.cpp](#)

```
#include <string>
#include <iostream>
#include <vector>
#include "parser/InitNetlist.h"
```

```
#include "sym_detect/SymDetect.h"
```

Include dependency graph for main.cpp:



## Macros

- `#define __SFA_TEST__`

## Functions

- `int main (int argc, char *argv[])`

### 4.13.1 Detailed Description

[main.cpp](#)

#### Author

Mingjie Llu

#### Date

11/25/2018

Takes 1 argument input. Parse the file into [Netlist](#). Detect hierarchy symmetry groups and print to command line. Input file should be of certain format. See [parser/InitNetlist.h](#) for details.

## 4.13.2 Macro Definition Documentation

### 4.13.2.1 \_\_SFA\_TEST\_\_

```
#define __SFA_TEST__
```

## 4.13.3 Function Documentation

### 4.13.3.1 main()

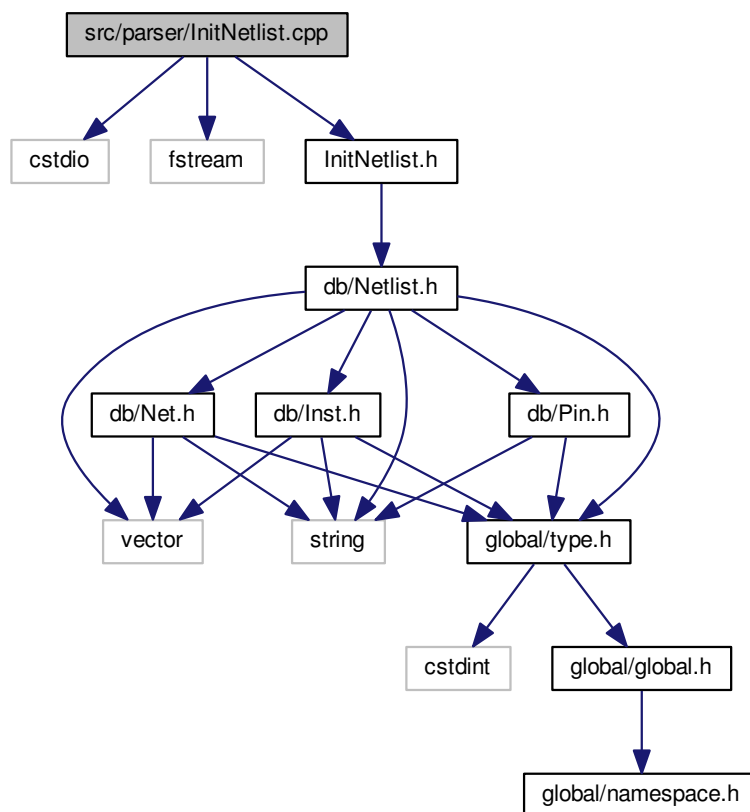
```
int main (  
    int argc,  
    char * argv[] )
```

## 4.14 src/parser/InitNetlist.cpp File Reference

Parser implementation.

```
#include <stdio>  
#include <fstream>  
#include "InitNetlist.h"
```

Include dependency graph for InitNetlist.cpp:



#### 4.14.1 Detailed Description

Parser implementation.

Author

Mingjie Liu

Date

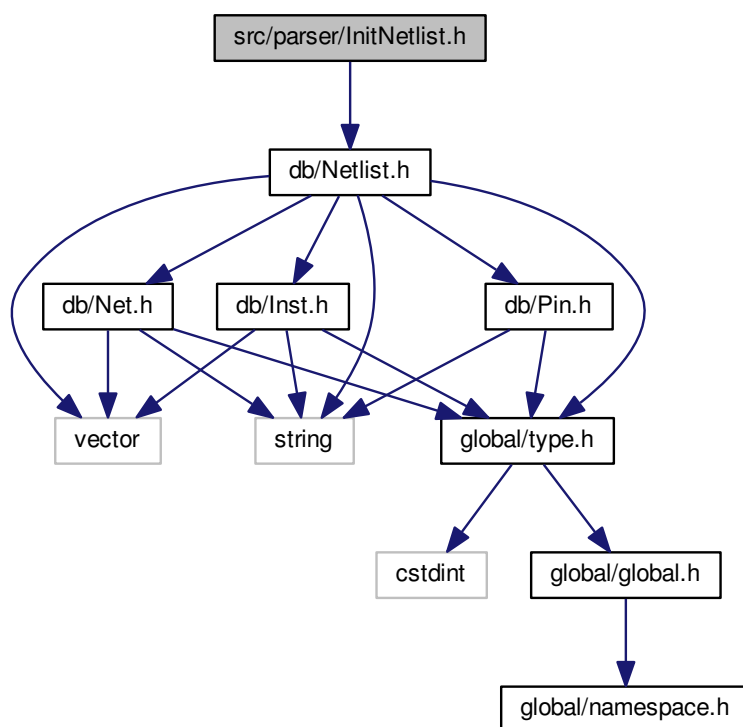
11/24/2018

#### 4.15 src/parser/InitNetlist.h File Reference

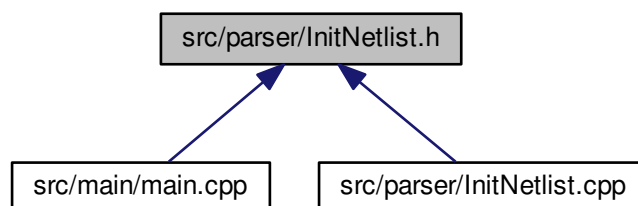
Parser to initialize netlist.

```
#include "db/Netlist.h"
```

Include dependency graph for InitNetlist.h:



This graph shows which files directly or indirectly include this file:



## Classes

- class [InitNetlist](#)  
*InitNetlist* class.

### 4.15.1 Detailed Description

Parser to initialize netlist.

Author

Mingjie Liu

Date

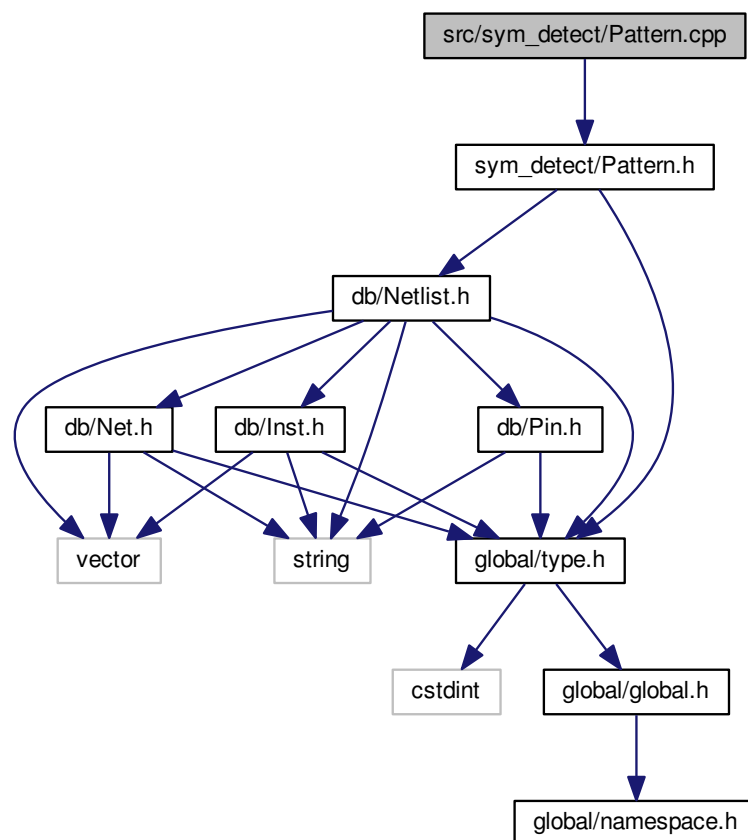
11/24/2018

Input file should follow same format generated through scripts/create\_init\_obj.py. The python scripts take standardized hspice/spectre netlist files as inputs. Sample input files for c++ are under benchmarks.

## 4.16 src/sym\_detect/Pattern.cpp File Reference

[Pattern](#) definitions.

```
#include "sym_detect/Pattern.h"
Include dependency graph for Pattern.cpp:
```



### 4.16.1 Detailed Description

[Pattern](#) definitions.

Author

Mingjie Liu

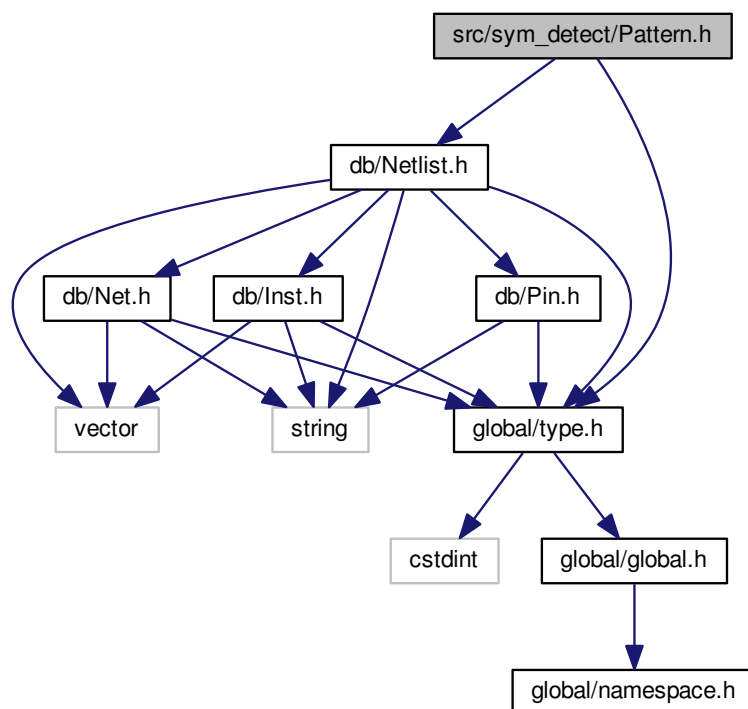
Date

11/24/2018

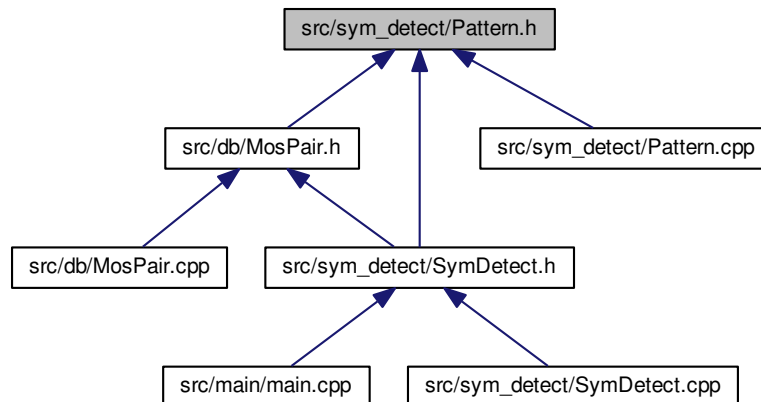
## 4.17 src/sym\_detect/Pattern.h File Reference

Mosfet pair patterns.

```
#include "db/Netlist.h"  
#include "global/type.h"  
Include dependency graph for Pattern.h:
```



This graph shows which files directly or indirectly include this file:



## Classes

- class [Pattern](#)  
*Pattern* class.

### 4.17.1 Detailed Description

Mosfet pair patterns.

This class has been augmented also to handle passive device matching and self symmetry mosfets. The name remains as legacy.

#### Author

Mingjie Liu

#### Date

11/24/2018

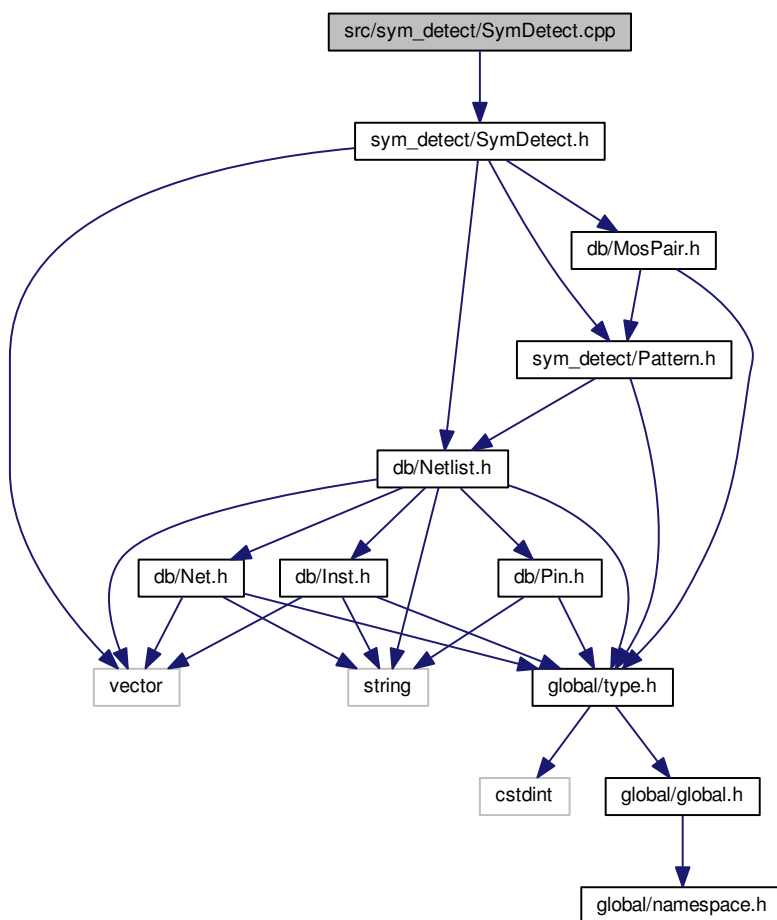
## 4.18 src/sym\_detect/SymDetect.cpp File Reference

Detect symmetric patterns.



```
#include "sym_detect/SymDetect.h"
```

Include dependency graph for SymDetect.cpp:



### 4.18.1 Detailed Description

Detect symmetric patterns.

Author

Mingjie Liu

Date

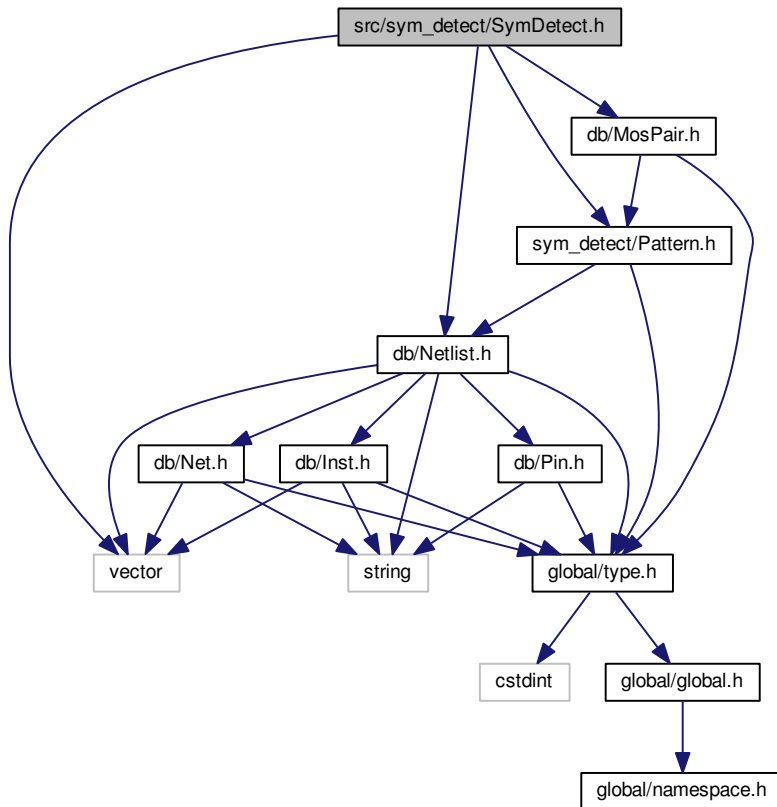
11/24/2018

## 4.19 src/sym\_detect/SymDetect.h File Reference

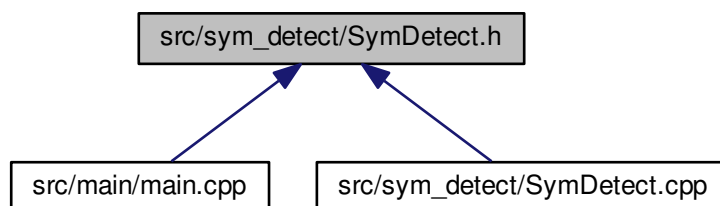
Detect symmetric patterns.

```
#include "db/Netlist.h"
#include "db/MosPair.h"
#include "sym_detect/Pattern.h"
#include <vector>
```

Include dependency graph for SymDetect.h:



This graph shows which files directly or indirectly include this file:



## Classes

- class [SymDetect](#)  
*SymDetect* class.

### 4.19.1 Detailed Description

Detect symmetric patterns.

#### Author

Mingjie Liu

#### Date

11/24/2018



# Index

- `__SFA_TEST__`
  - `main.cpp`, 71
- `_id`
  - `Inst`, 13
  - `Net`, 21
  - `Pin`, 38
- `_instArray`
  - `Netlist`, 31
- `_instId`
  - `Pin`, 38
- `_len`
  - `Inst`, 14
- `_mosId1`
  - `MosPair`, 18
- `_mosId2`
  - `MosPair`, 18
- `_name`
  - `Inst`, 14
  - `Net`, 21
- `_netArray`
  - `Netlist`, 31
- `_netId`
  - `Pin`, 38
- `_netlist`
  - `Pattern`, 35
  - `SymDetect`, 46
- `_netlistDB`
  - `InitNetlist`, 9
- `_pattern`
  - `MosPair`, 19
  - `SymDetect`, 47
- `_pinArray`
  - `Netlist`, 31
- `_pinIdArray`
  - `Inst`, 14
  - `Net`, 21
- `_srchPinType1`
  - `MosPair`, 19
- `_srchPinType2`
  - `MosPair`, 19
- `_type`
  - `Inst`, 14
  - `Pin`, 39
- `_valid`
  - `MosPair`, 19
- `_wid`
  - `Inst`, 14
- `addInst`
  - `Netlist`, 24
- `addNet`
  - `Netlist`, 24
- `addPin`
  - `Netlist`, 24
- `addPinId`
  - `Inst`, 12
  - `Net`, 20
- `addSelfSym`
  - `SymDetect`, 41
- `Byte`
  - `type.h`, 66
- `crossPairCascode`
  - `Pattern`, 33
- `crossPairLoad`
  - `Pattern`, 33
- `dfsDiffPair`
  - `SymDetect`, 41
- `diffPairCascode`
  - `Pattern`, 33
- `diffPairInput`
  - `Pattern`, 34
- `drainNetId`
  - `Netlist`, 24
- `endSrch`
  - `SymDetect`, 42
- `existPair`
  - `SymDetect`, 42
- `fltrInstMosType`
  - `Netlist`, 25
- `fltrInstNetConnPinType`
  - `Netlist`, 25
- `fltrInstPinConnPinType`
  - `Netlist`, 25
- `GROUND_NET_NAMES`
  - `Net.cpp`, 54
- `gateNetId`
  - `Netlist`, 26
- `getDiffPair`
  - `SymDetect`, 42
- `getInstNetConn`
  - `Netlist`, 26
- `getInstPinConn`
  - `Netlist`, 26
- `getPatrnNetConn`
  - `SymDetect`, 43

- getPinTypeInstNetConn
  - Netlist, 27
- getPinTypeInstPinConn
  - Netlist, 27
- getVldDrainMos
  - SymDetect, 43
- hiSymDetect
  - SymDetect, 43
- INDEX\_TYPE\_MAX
  - type.h, 68
- INT\_TYPE\_MAX
  - type.h, 68
- INT\_TYPE\_MIN
  - type.h, 69
- id
  - Inst, 12
  - Net, 20
  - Netlist::InitNet, 7
  - Pin, 37
- inVld
  - MosPair, 16
- inVldDiffPairSrch
  - SymDetect, 44
- IndexType
  - type.h, 66
- init
  - Netlist, 28
- InitNetlist, 8
  - \_netlistDB, 9
  - InitNetlist, 9
  - read, 9
- Inst, 10
  - \_id, 13
  - \_len, 14
  - \_name, 14
  - \_pinIdArray, 14
  - \_type, 14
  - \_wid, 14
  - addPinId, 12
  - id, 12
  - Inst, 11
  - len, 12
  - name, 12
  - pinIdArray, 12
  - setLen, 13
  - setWid, 13
  - type, 13
  - wid, 13
- inst
  - Netlist, 28
- instArray
  - Netlist::InitDataObj, 5
- instId
  - Pin, 37
- instNetId
  - Netlist, 28
- instPinId
  - Netlist, 28
- InstType
  - type.h, 66
- IntType
  - type.h, 66
- isEqual
  - MosPair, 16
- isMos
  - Netlist, 29
- isPasvDev
  - Netlist, 29
  - Pin, 37
- isSignal
  - Netlist, 29
- len
  - Inst, 12
  - Netlist::InitInst, 6
- MOS\_PIN\_TYPE
  - Netlist.cpp, 57
- main
  - main.cpp, 71
- main.cpp
  - \_\_SFA\_TEST\_\_, 71
  - main, 71
- matchedSize
  - Pattern, 34
- matchedType
  - Pattern, 34
- mosId1
  - MosPair, 16
- mosId2
  - MosPair, 17
- MosPair, 14
  - \_mosId1, 18
  - \_mosId2, 18
  - \_pattern, 19
  - \_srchPinType1, 19
  - \_srchPinType2, 19
  - \_valid, 19
  - inVld, 16
  - isEqual, 16
  - mosId1, 16
  - mosId2, 17
  - MosPair, 15, 16
  - nextPinType1, 17
  - nextPinType2, 17
  - pattern, 17
  - setSrchPinType1, 17
  - setSrchPinType2, 17
  - srchPinType1, 18
  - srchPinType2, 18
  - valid, 18
- MosPairPtrn
  - SymDetect, 44
- MosPattern
  - type.h, 67
- MosType

- type.h, 67
- mosType
  - Netlist, 29
- name
  - Inst, 12
  - Net, 21
  - Netlist::InitInst, 6
  - Netlist::InitNet, 7
- namespace.h
  - PROJECT\_NAMESPACE\_BEGIN, 64
  - PROJECT\_NAMESPACE\_END, 64
  - PROJECT\_NAMESPACE, 63
- Net, 19
  - \_id, 21
  - \_name, 21
  - \_pinIdArray, 21
  - addPinId, 20
  - id, 20
  - name, 21
  - Net, 20
  - netType, 21
  - pinIdArray, 21
- net
  - Netlist, 29
- Net.cpp
  - GROUND\_NET\_NAMES, 54
  - POWER\_NET\_NAMES, 54
- netArray
  - Netlist::InitDataObj, 5
- netId
  - Pin, 37
- netIdArray
  - Netlist::InitInst, 6
- NetType
  - type.h, 68
- netType
  - Net, 21
- Netlist, 22
  - \_instArray, 31
  - \_netArray, 31
  - \_pinArray, 31
  - addInst, 24
  - addNet, 24
  - addPin, 24
  - drainNetId, 24
  - fltrInstMosType, 25
  - fltrInstNetConnPinType, 25
  - fltrInstPinConnPinType, 25
  - gateNetId, 26
  - getInstNetConn, 26
  - getInstPinConn, 26
  - getPinTypeInstNetConn, 27
  - getPinTypeInstPinConn, 27
  - init, 28
  - inst, 28
  - instNetId, 28
  - instPinId, 28
  - isMos, 29
  - isPasvDev, 29
  - isSignal, 29
  - mosType, 29
  - net, 29
  - Netlist, 24
  - numInst, 30
  - numNet, 30
  - numPin, 30
  - pin, 30
  - print\_all, 30
  - rmvInstHasPin, 30
  - srcNetId, 31
- Netlist.cpp
  - MOS\_PIN\_TYPE, 57
  - RES\_PIN\_TYPE, 57
- Netlist::InitDataObj, 5
  - instArray, 5
  - netArray, 5
- Netlist::InitInst, 6
  - len, 6
  - name, 6
  - netIdArray, 6
  - type, 7
  - wid, 7
- Netlist::InitNet, 7
  - id, 7
  - name, 7
- nextPinType
  - Pin, 37
- nextPinType1
  - MosPair, 17
- nextPinType2
  - MosPair, 17
- numInst
  - Netlist, 30
- numNet
  - Netlist, 30
- numPin
  - Netlist, 30
- POWER\_NET\_NAMES
  - Net.cpp, 54
- PROJECT\_NAMESPACE\_BEGIN
  - namespace.h, 64
- PROJECT\_NAMESPACE\_END
  - namespace.h, 64
- PROJECT\_NAMESPACE
  - namespace.h, 63
- Pattern, 32
  - \_netlist, 35
  - crossPairCascode, 33
  - crossPairLoad, 33
  - diffPairCascode, 33
  - diffPairInput, 34
  - matchedSize, 34
  - matchedType, 34
  - Pattern, 33
  - pattern, 34
  - validPairCascode, 35

- validPairLoad, 35
- pattern
  - MosPair, 17
  - Pattern, 34
- Pin, 35
  - \_id, 38
  - \_instId, 38
  - \_netId, 38
  - \_type, 39
  - id, 37
  - instId, 37
  - isPasvDev, 37
  - netId, 37
  - nextPinType, 37
  - Pin, 36
  - type, 38
- pin
  - Netlist, 30
- pinIdArray
  - Inst, 12
  - Net, 21
- PinType
  - type.h, 68
- print\_all
  - Netlist, 30
- pushNextSrchObj
  - SymDetect, 44
- REAL\_TYPE\_MAX
  - type.h, 69
- REAL\_TYPE\_MIN
  - type.h, 69
- REAL\_TYPE\_TOL
  - type.h, 69
- RES\_PIN\_TYPE
  - Netlist.cpp, 57
- read
  - InitNetlist, 9
- RealType
  - type.h, 66
- rmvInstHasPin
  - Netlist, 30
- selfSymSrch
  - SymDetect, 45
- setLen
  - Inst, 13
- setSrchPinType1
  - MosPair, 17
- setSrchPinType2
  - MosPair, 17
- setWid
  - Inst, 13
- src/db/Inst.h, 49
- src/db/MosPair.cpp, 50
- src/db/MosPair.h, 51
- src/db/Net.cpp, 53
- src/db/Net.h, 54
- src/db/Netlist.cpp, 56
- src/db/Netlist.h, 58
- src/db/Pin.cpp, 59
- src/db/Pin.h, 60
- src/global/global.h, 61
- src/global/namespace.h, 63
- src/global/type.h, 64
- src/main/main.cpp, 69
- src/parser/InitNetlist.cpp, 71
- src/parser/InitNetlist.h, 72
- src/sym\_detect/Pattern.cpp, 74
- src/sym\_detect/Pattern.h, 75
- src/sym\_detect/SymDetect.cpp, 76
- src/sym\_detect/SymDetect.h, 78
- srcNetId
  - Netlist, 31
- srchPinType1
  - MosPair, 18
- srchPinType2
  - MosPair, 18
- SymDetect, 39
  - \_netlist, 46
  - \_pattern, 47
  - addSelfSym, 41
  - dfsDiffPair, 41
  - endSrch, 42
  - existPair, 42
  - getDiffPair, 42
  - getPatrnNetConn, 43
  - getVldDrainMos, 43
  - hiSymDetect, 43
  - inVldDiffPairSrch, 44
  - MosPairPtrn, 44
  - pushNextSrchObj, 44
  - selfSymSrch, 45
  - SymDetect, 40
  - validDiffPair, 45
  - validSrchObj, 46
- type
  - Inst, 13
  - Netlist::InitInst, 7
  - Pin, 38
- type.h
  - Byte, 66
  - INDEX\_TYPE\_MAX, 68
  - INT\_TYPE\_MAX, 68
  - INT\_TYPE\_MIN, 69
  - IndexType, 66
  - InstType, 66
  - IntType, 66
  - MosPattern, 67
  - MosType, 67
  - NetType, 68
  - PinType, 68
  - REAL\_TYPE\_MAX, 69
  - REAL\_TYPE\_MIN, 69
  - REAL\_TYPE\_TOL, 69
  - RealType, 66



valid  
    MosPair, [18](#)  
validDiffPair  
    SymDetect, [45](#)  
validPairCascode  
    Pattern, [35](#)  
validPairLoad  
    Pattern, [35](#)  
validSrchObj  
    SymDetect, [46](#)  
  
wid  
    Inst, [13](#)  
    Netlist::InitInst, [7](#)