SFA

0.1.0

Generated by Doxygen 1.8.13

Contents

Chapter 1

Class Index

1.1 Class List

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

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

Chapter 3

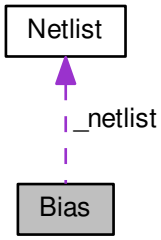
Class Documentation

3.1 Bias Class Reference

A vector of Mosfet.

`#include <Bias.h>`

Collaboration diagram for Bias:



Public Member Functions

- Bias ()=default

  Default Constructor.
- Bias (IndexType netId, const Netlist &netlist)

  Constructor for Bias.
- const std::vector< IndexType > & bias () const

  Get entire bias group.
- const std::vector< IndexType > & driver () const

  Get the driver group.
- bool valid () const
- void init ()

Private Attributes

- IndexType _netId
- const Netlist & _netlist
- std::vector< IndexType > _bias
- std::vector< IndexType > _driver

3.1.1 Detailed Description

A vector of Mosfet.

This class stores a group of Mosfet Id that are bias circuits.

3.1.2 Constructor & Destructor Documentation

3.1.2.1 Bias() [1/2]

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

Default Constructor.

3.1.2.2 Bias() [2/2]

```
Bias::Bias (
            IndexType netId,
            const Netlist & netlist )  [inline], [explicit]
```

Constructor for Bias.

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

Parameters

netId	Gate netId.
netlist	Netlist class object.

3.1.3 Member Function Documentation

3.1.3.1 bias()

```
const std::vector<IndexType>& Bias::bias ( ) const  [inline]
```

Get entire bias group.

3.1.3.2 driver()

```
const std::vector<IndexType>& Bias::driver ( ) const  [inline]
```

Get the driver group.

3.1.3.3 init()

```
PROJECT_NAMESPACE_BEGIN void Bias::init ( )
```

3.1.3.4 valid()

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

3.1.4 Member Data Documentation

3.1.4.1 _bias

```
std::vector<IndexType> Bias::_bias  [private]
```

3.1.4.2 _driver

```
std::vector<IndexType> Bias::_driver  [private]
```

3.1.4.3 _netId

```
IndexType Bias::_netId  [private]
```

3.1.4.4 _netlist

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

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

- src/db/Bias.h
- src/db/Bias.cpp

3.2 Netlist::InitDataObj Struct Reference

Instantiate Netlist class.

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

Public Attributes

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

3.2.1 Detailed Description

Instantiate Netlist class.

See also

init(InitDataObj &).

3.2.2 Member Data Documentation

3.2.2.1 instArray

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

3.2.2.2 netArray

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

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

- src/db/Netlist.h

3.3 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

3.3.1 Detailed Description

Inst for instantiation.

3.3.2 Member Data Documentation

3.3.2.1 len

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

3.3.2.2 name

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

3.3.2.3 netIdArray

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

3.3.2.4 type

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

3.3.2.5 wid

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

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

- src/db/Netlist.h

3.4 Netlist::InitNet Struct Reference

Net for instantiation.

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

Public Attributes

- std::string name
- IndexType id = INDEX_TYPE_MAX

3.4.1 Detailed Description

Net for instantiation.

3.4.2 Member Data Documentation

3.4.2.1 id

```
IndexType Netlist::InitNet::id = INDEX_TYPE_MAX
```

3.4.2.2 name

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

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

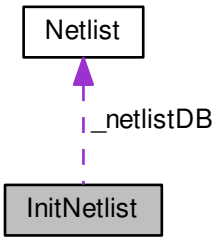
- src/db/Netlist.h

3.5 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

3.5.1 Detailed Description

InitNetlist class.

3.5.2 Constructor & Destructor Documentation

3.5.2.1 InitNetlist() [1/2]

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

Default Constructor.

3.5.2.2 InitNetlist() [2/2]

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

Constructor with initialization.

3.5.3 Member Function Documentation

3.5.3.1 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

filename	Input file to parse.

3.5.4 Member Data Documentation

3.5.4.1 _netlistDB

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

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

- src/parser/InitNetlist.h
- src/parser/InitNetlist.cpp

3.6 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

3.6.1 Detailed Description

Inst class.

3.6.2 Constructor & Destructor Documentation

3.6.2.1 Inst() [1/3]

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

Default constructor.

3.6.2.2 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

name	Name of Inst.
type	Type of Inst. Member of InstType.

See also

type.h

Parameters

id	Id of Inst.

3.6.2.3 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

name	Name of Inst.
type	Type of Inst. Member of InstType.
id	Id of INst.
wid	Width of Inst.
len	Length of Inst.

3.6.3 Member Function Documentation

3.6.3.1 addPinId()

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

Add pin index to Inst.

Parameters

pinId	Added pin Id.

3.6.3.2 id()

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

Return Id of Inst.

3.6.3.3 len()

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

Return length of Inst.

3.6.3.4 name()

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

Return name of Inst.

3.6.3.5 pinIdArray()

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

Return the index array for pins of the Inst.

3.6.3.6 setLen()

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

Assign length of Inst.

3.6.3.7 setWid()

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

Assign width of Inst.

3.6.3.8 type()

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

Return type of Inst.

See also

InstType

3.6.3.9 wid()

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

Return width of Inst.

3.6.4 Member Data Documentation

3.6.4.1 _id

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

3.6.4.2 _len

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

3.6.4.3 _name

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

3.6.4.4 _pinIdArray

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

3.6.4.5 _type

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

3.6.4.6 _wid

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

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

- src/db/Inst.h

3.7 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

3.7.1 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.

3.7.2 Constructor & Destructor Documentation

3.7.2.1 MosPair() [1/2]

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

Default Constructor.

3.7.2.2 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

mosId1	Id for Mos1
mosId2	Id for Mos2

< valid is set true as default.

< reached Pin set as SOURCE default.

3.7.3 Member Function Documentation

3.7.3.1 inVld()

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

Invalidate pair.

3.7.3.2 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.

3.7.3.3 mosId1()

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

Get mosId1.

3.7.3.4 mosId2()

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

Get mosId2.

3.7.3.5 nextPinType1()

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

Return next PinType to search for mosId1.

3.7.3.6 nextPinType2()

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

Return next PinType to search for mosId2.

3.7.3.7 pattern()

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

Get pattern.

3.7.3.8 setSrchPinType1()

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

set reached PinType.

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

3.7.3.9 setSrchPinType2()

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

set reached PinType.

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

3.7.3.10 srchPinType1()

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

Get PinType on how DFS reached mosId1 of the pair.

3.7.3.11 srchPinType2()

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

Get PinType on how DFS reached mosId1 of the pair.

3.7.3.12 valid()

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

Return if valid search pair.

See also

SymDetect::inVldDiffPairSrch

3.7.4 Member Data Documentation

3.7.4.1 _mosId1

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

3.7.4.2 _mosId2

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

3.7.4.3 _pattern

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

3.7.4.4 _srchPinType1

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

3.7.4.5 _srchPinType2

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

3.7.4.6 _valid

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

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

- src/db/MosPair.h
- src/db/MosPair.cpp

3.8 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

3.8.1 Detailed Description

Net class.

3.8.2 Constructor & Destructor Documentation

3.8.2.1 Net() [1/2]

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

Default constructor.

3.8.2.2 Net() [2/2]

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

Constructor of Net.

Parameters

name	Name of Net.
id	Id of Net.

3.8.3 Member Function Documentation

3.8.3.1 addPinId()

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

Connect a pin to the net.

3.8.3.2 id()

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

Return Id of Net.

3.8.3.3 name()

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

Return name of Net.

3.8.3.4 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 VDD/VSS. Add unsupported names for Power/Ground filtering to POWER_NET_NAMES and GROUND_NET_NAMES to /db/Net.cpp.

3.8.3.5 pinIdArray()

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

Return index array of connected pins.

3.8.4 Member Data Documentation

3.8.4.1 _id

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

3.8.4.2 _name

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

3.8.4.3 _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

3.9 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 Mosfet Inst whose type are not mosType.
- void fltrInstType (std::vector< IndexType > &instArray, InstType type) const

  Filter instArray. Remove Inst whose type are not type.
- 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

3.9.1 Detailed Description

Netlist class.

3.9.2 Constructor & Destructor Documentation

3.9.2.1 Netlist()

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

Default Constructor.

3.9.3 Member Function Documentation

3.9.3.1 addInst()

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

Add Inst to Netlist.

3.9.3.2 addNet()

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

Add Net to Netlist.

3.9.3.3 addPin()

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

Add Pin to Netlist.

3.9.3.4 drainNetId()

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

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

See also

instNetId

3.9.3.5 fltrInstMosType()

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

Filter instArray. Remove Mosfet Inst whose type are not mosType.

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

See also

getPinTypeInstNetConn.

3.9.3.6 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.

3.9.3.7 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.

3.9.3.8 fltrInstType()

```
void Netlist::fltrInstType (
            std::vector< IndexType > & instArray,
            InstType type ) const
```

Filter instArray. Remove Inst whose type are not type.

Removed instId if InstType of instance is different from input type.

3.9.3.9 gateNetId()

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

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

See also

instNetId.

3.9.3.10 getInstNetConn()

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

Get all Inst that are connected to netId.

Parameters

out	*instArray*	Array of the returned Inst Id.
in	*netId*	Id of net.

3.9.3.11 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	*instArray*	Array of the returned Inst Id.
in	*pinId*	Id of pin.

3.9.3.12 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 querry 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

instId	Id of Inst that returned pin is connected.
netId	Id of Net that returned pin is connected.

3.9.3.13 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 querry 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

instId	Id of Inst that returned pin is connected.
pinId	Id of Pin that returned pin is connected.

3.9.3.14 init()

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

Initialize Netlist class.

3.9.3.15 inst()

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

Return Inst of Id.

3.9.3.16 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 PinType::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

instId	Id of Inst.
pinType	Returned Net Id connected to this PinType.

3.9.3.17 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

instId	Id of Inst.
pinType	Returned Pin Id should be this PinType.

3.9.3.18 isMos()

```
bool Netlist::isMos (
            InstType instType ) const
```

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

3.9.3.19 isPasvDev()

```
bool Netlist::isPasvDev (
            InstType instType ) const
```

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

3.9.3.20 isSignal()

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

Return true if corresponding net NetType::Signal.

3.9.3.21 mosType()

```
MosType Netlist::mosType (
            IndexType mosId ) const
```

Return MosType of corresponding instance id.

3.9.3.22 net()

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

Return Net of Id.

3.9.3.23 numInst()

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

Return number of Inst.

3.9.3.24 numNet()

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

Return number of Net.

3.9.3.25 numPin()

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

Return number of Pin.

3.9.3.26 pin()

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

Return Pin of Id.

3.9.3.27 print_all()

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

Print netlist.

3.9.3.28 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

instArray	Reference to instance Id array.
pinId	Id of pin.

3.9.3.29 srcNetId()

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

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

See also

instNetId.

3.9.4 Member Data Documentation

3.9.4.1 _instArray

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

3.9.4.2 _netArray

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

3.9.4.3 _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

3.10 NetPair Class Reference

A pair of Net that are symmetric.

```
#include <NetPair.h>
```

Public Member Functions

- NetPair ()=default

  Default Constructor.
- NetPair (IndexType netId1, IndexType netId2)

  Constructor for NetPair.
- IndexType netId1 () const

  Get netId1.
- IndexType netId2 () const

  Get netId2.

Private Attributes

- IndexType _netId1
- IndexType _netId2

3.10.1 Detailed Description

A pair of Net that are symmetric.

This class stores a pair of Net Id. A pair of net are symmetric if all Inst connected could be grouped to form MosPair. This also contains self symmetry pairs that are connected to DIFF_SOURCE.

3.10.2 Constructor & Destructor Documentation

3.10.2.1 NetPair() [1/2]

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

Default Constructor.

3.10.2.2 NetPair() [2/2]

```
NetPair::NetPair (
            IndexType netId1,
            IndexType netId2 )  [inline], [explicit]
```

Constructor for NetPair.

Sequence of Ids does not matter.

Parameters

netId1	Id for Net1
netId2	Id for Net2

3.10.3 Member Function Documentation

3.10.3.1 netId1()

```
IndexType NetPair::netId1 ( ) const  [inline]
```

Get netId1.

3.10.3.2 netId2()

```
IndexType NetPair::netId2 ( ) const  [inline]
```

Get netId2.

3.10.4 Member Data Documentation

3.10.4.1 _netId1

```
IndexType NetPair::_netId1  [private]
```

3.10.4.2 _netId2

```
IndexType NetPair::_netId2  [private]
```

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

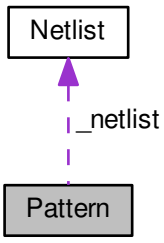
- src/db/NetPair.h

3.11 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 diffPairCascode (IndexType mosId1, IndexType mosId2) const

  Return true if fits MosPattern::DIFF_CASCODE.
- bool validPairCascode (IndexType mosId1, IndexType mosId2) const

  Return true if fits MosPattern::CASCODE.
- bool validPairLoad (IndexType mosId1, IndexType mosId2) const

  Return true if fits MosPattern::LOAD.
- bool crossPairCascode (IndexType mosId1, IndexType mosId2) const

  Return true if fits MosPattern::CROSS_CASCODE.
- bool crossPairLoad (IndexType mosId1, IndexType mosId2) const

  Return true if fits MosPattern::CROSS_LOAD.

Private Attributes

- const Netlist & _netlist

3.11.1 Detailed Description

Pattern class.

3.11.2 Constructor & Destructor Documentation

3.11.2.1 Pattern()

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

Constructor.

Parameters

netlist	Netlist for pattern search.

3.11.3 Member Function Documentation

3.11.3.1 crossPairCascode()

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

Return true if fits MosPattern::CROSS_CASCODE.

3.11.3.2 crossPairLoad()

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

Return true if fits MosPattern::CROSS_LOAD.

3.11.3.3 diffPairCascode()

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

Return true if fits MosPattern::DIFF_CASCODE.

3.11.3.4 diffPairInput()

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

Return true if fits MosPattern::DIFF_SOURCE.

3.11.3.5 matchedSize()

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

Return true if Inst pair have same size attributes.

3.11.3.6 matchedType()

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

Return true if Inst pair have same InstType.

3.11.3.7 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

mosId1	Id for mosfet.
mosId2	Id for mosfet.

3.11.3.8 validPairCascode()

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

Return true if fits MosPattern::CASCODE.

3.11.3.9 validPairLoad()

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

Return true if fits MosPattern::LOAD.

3.11.4 Member Data Documentation

3.11.4.1 _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

3.12 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

3.12.1 Detailed Description

Pin class.

3.12.2 Constructor & Destructor Documentation

3.12.2.1 Pin() [1/2]

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

Default Constructor.

3.12.2.2 Pin() [2/2]

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

Constructor for Pin.

Parameters

id	Id of Pin.
instId	Id of connected Inst.
netId	Id of connected Net.
type	Type of Pin.

3.12.3 Member Function Documentation

3.12.3.1 id()

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

Return id of Pin.

3.12.3.2 instId()

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

Return id of connected Inst.

3.12.3.3 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.

3.12.3.4 netId()

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

Return id of connected Net.

3.12.3.5 nextPinType()

```
PROJECT_NAMESPACE_BEGIN PinType Pin::nextPinType (
            PinType type )  [static]
```

Return the next search PinType for DFS.

Parameters

type	Querry the next search PinType.

See also

PinType

The DFS search for symmetry relys 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
GATE	DRAIN
DRAIN	SOURCE
THIS	THAT
THAT	THIS

3.12.3.6 type()

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

Return type of Pin.

See also

PinType

3.12.4 Member Data Documentation

3.12.4.1 _id

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

3.12.4.2 _instId

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

3.12.4.3 _netId

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

3.12.4.4 _type

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

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

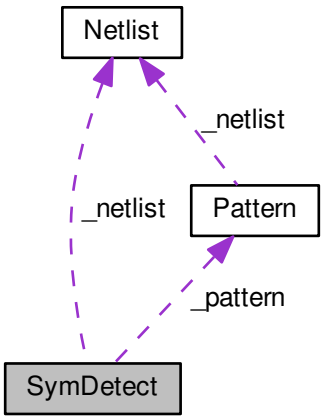
- src/db/Pin.h
- src/db/Pin.cpp

3.13 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 print () const

  Print symGroup for netlist.
- void dumpSym (const std::string file) const

  Dump symmetry constraint to file.
- void dumpNet (const std::string file) const

  Dump symmetry net to file.

Private Member Functions

- MosPattern MosPairPtrn (MosPair &obj) const

  Return pattern of MosPair.
- bool existPair (const std::vector< MosPair > &library, IndexType instId1, IndexType instId2) const

  Check if pair already reached.
- bool existPair (std::vector< MosPair > &library, IndexType instId) const
- bool existNetPair (std::vector< NetPair > &library, IndexType netId1, IndexType netId2) const

  Check if already contains NetPair in library.
- bool existNetPair (std::vector< NetPair > &library, IndexType netId) const

  Check if self symmetry Net in library.
- bool endSrch (MosPair &obj) const

Return true if end of search path.

- bool validSrchObj (IndexType instId1, IndexType instId2, IndexType srchPinId1, IndexType srchPinId2) const

  Return true if a valid pair.
- bool validDiffPair (IndexType instId1, IndexType instId2, IndexType srchPinId1, IndexType srchPinId2) const

  Return true if a valid DIFF_SOURCE gate connected.
- bool validNetPair (IndexType netId1, IndexType netId2, std::vector< NetPair > &netPair) const

  Return true if a valid symmetry NetPair.
- bool checkNetSym (IndexType netId1, IndexType netId2) const

  Check every pin of nets for symmetry.
- void pushNextSrchObj (std::vector< MosPair > &dfsVstPair, std::vector< MosPair > &dfsStack, MosPair &currObj, std::vector< MosPair > &diffPairSrc) const

  Push next valid MosPair to dfsStack.
- bool comBias (MosPair &currObj) const

  Return true if currObj have common gate connection.
- void addBiasSym (std::vector< MosPair > &dfsVstPair, MosPair &currObj) const

  A special case where a symmetry pair is formed in the bias group.
- void getPatrnNetConn (std::vector< MosPair > &diffPair, IndexType netId, MosPattern srchPatrn) const

  Get srchPatrn MosPair connected to netId.
- 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 netId) const

  Get valid drain connected mosfet to netId.
- 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.
- void addSymNet (std::vector< NetPair > &netPair, MosPair &currObj) const

  Based on currObj symmetry Inst pair, valid symmetry nets are appended to netPair.
- void flattenSymGroup (std::vector< std::vector< MosPair >> &symGroup, std::vector< MosPair > &flatPair) const

  Flatten symmetry group hierarchy into a single vector.
- void biasGroup (std::vector< MosPair > &flatPair, std::vector< Bias > &biasGroup, std::vector< NetPair > &netPair) const

  Find all bias groups.
- void biasMatch (std::vector< Bias > &biasGroup, std::vector< std::vector< MosPair >> &symGroup, std↩ ::vector< MosPair > &flatPair) const

  Search for symmetry pairs in each group.
- void hiSymDetect (std::vector< std::vector< MosPair >> &symGroup) const

  Hierarchy symmetry detection.

Private Attributes

- const Netlist & _netlist
- Pattern _pattern
- std::vector< NetPair > _symNet

  Symmetry nets of netlist.
- std::vector< std::vector< MosPair > > _symGroup

  Symmetry groups of netlist.
- std::vector< MosPair > _flatPair
- std::vector< Bias > _biasGroup

3.13.1 Detailed Description

SymDetect class.

3.13.2 Constructor & Destructor Documentation

3.13.2.1 SymDetect()

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

Constructor Only needs netlist as input. Pattern class inherently constructed.

Parameters

netlist	Netlist class.

3.13.3 Member Function Documentation

3.13.3.1 addBiasSym()

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

A special case where a symmetry pair is formed in the bias group.

3.13.3.2 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

dfsVstPair	Symmetry group.

See also

selfSymSrch
hiSymDetect

3.13.3.3 addSymNet()

```
void SymDetect::addSymNet (
            std::vector< NetPair > & netPair,
            MosPair & currObj ) const  [private]
```

Based on currObj symmetry Inst pair, valid symmetry nets are appended to netPair.

Valid symmetry net that are connected to symmetry Inst pair currObj would be added to vector.

See also

validNetPair

Parameters

netPair	Symmetry Net appended to this vector.
currObj	Current symmetry Inst pair.

3.13.3.4 biasGroup()

```
void SymDetect::biasGroup (
            std::vector< MosPair > & flatPair,
            std::vector< Bias > & biasGroup,
            std::vector< NetPair > & netPair ) const  [private]
```

Find all bias groups.

All MosPair in flattened symmetry group are first searched as source. For all valid bias search source that is comBias, bias groups would be saved to biasGroup.

Symmetry nets would be appended to netPair vector.

See also

comBias

Parameters

flatPair	Input flattened symmetry group.
biasGroup	Saved bias groups to vector.
netPair	Saved symmetry nets.

3.13.3.5 biasMatch()

```
void SymDetect::biasMatch (
            std::vector< Bias > & biasGroup,
            std::vector< std::vector< MosPair >> & symGroup,
            std::vector< MosPair > & flatPair ) const  [private]
```

Search for symmetry pairs in each group.

New symmetry pairs are searched in the biasGroup.

Parameters

biasGroup	A vector of bias group.
symGroup	Results appended to symGroup.
flatPair	Used to check for redundancy.

3.13.3.6 checkNetSym()

```
bool SymDetect::checkNetSym (
            IndexType netId1,
            IndexType netId2 ) const  [private]
```

Check every pin of nets for symmetry.

3.13.3.7 comBias()

```
bool SymDetect::comBias (
            MosPair & currObj ) const  [private]
```

Return true if currObj have common gate connection.

This function is used to check if a MosPair needs to search for a bias group. MosPair should have following attributes: (1) MosPattern::LOAD or CASCODE (2) Both mosId are of MosType::DIFF (3) Have common gate connection

3.13.3.8 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. All symmetry nets would be appended to netPair vector.

See also

pushNextSrchObj

Parameters

out	*dfsVstPair*	Vector to store all visited MosPair
in	*diffPair*	DFS search source
in	*diffPairSrch*	Vector of all stored DFS search source

3.13.3.9 dumpNet()

```
void SymDetect::dumpNet (
            const std::string file ) const
```

Dump symmetry net to file.

3.13.3.10 dumpSym()

```
PROJECT_NAMESPACE_BEGIN void SymDetect::dumpSym (
            const std::string file ) const
```

Dump symmetry constraint to file.

3.13.3.11 endSrch()

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

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

3.13.3.12 existNetPair() [1/2]

```
bool SymDetect::existNetPair (
            std::vector< NetPair > & library,
            IndexType netId1,
            IndexType netId2 ) const  [private]
```

Check if already contains NetPair in library.

3.13.3.13 existNetPair() [2/2]

```
bool SymDetect::existNetPair (
            std::vector< NetPair > & library,
            IndexType netId ) const  [private]
```

Check if self symmetry Net in library.

3.13.3.14 existPair() [1/2]

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

Check if pair already reached.

3.13.3.15 existPair() [2/2]

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

Check if self symmetry pair already reached.

3.13.3.16 flattenSymGroup()

```
void SymDetect::flattenSymGroup (
            std::vector< std::vector< MosPair >> & symGroup,
            std::vector< MosPair > & flatPair ) const  [private]
```

Flatten symmetry group hierarchy into a single vector.

3.13.3.17 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

diffPair	Store the output vector

3.13.3.18 getPatrnNetConn()

```
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

netId	Source should be connected to netId.
diffPair	Stored output vector.

3.13.3.19 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

vldMos	Vector to store valid Mosfet.
netId	Id of connected net.

3.13.3.20 hiSymDetect()

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

Hierarchy symmetry detection.

Output would contain 2 levels of hierarchy. symGroup is a vector of std::vector<MosPair> oneGroup. Where oneGroup 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

symGroup	Detected symmetry groups of netlist.

See also

MosPattern
MosPair

3.13.3.21 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

diffPairSrch	Vector of all DFS sources.
currPair	MosPair to invalidate.

3.13.3.22 MosPairPtrn()

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

Return pattern of MosPair.

3.13.3.23 print()

```
void SymDetect::print ( ) const
```

Print symGroup for netlist.

3.13.3.24 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

dfsVstPair	All current visited MosPair
dfsStack	Stack to store to visit MosPair
currObj	Current MosPair under visit
diffPairSrc	All DFS sources

3.13.3.25 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

dfsVstPair	Self symmetric pairs will be added to this vector.
diffPair	MosPattern::DIFF_SOURCE pair to begin self symmetry search.

See also

getVldDrainMos

3.13.3.26 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

instId1	Reached pair instId1
instId2	Reached pair instId2
srchPinId1	instId1 reached by srchPinId1.
srchPinId2	instId2 reached by srchPinId2.

3.13.3.27 validNetPair()

```
bool SymDetect::validNetPair (
            IndexType netId1,
            IndexType netId2,
            std::vector< NetPair > & netPair ) const  [private]
```

Return true if a valid symmetry NetPair.

A NetPair is a pair of symmetry nets. Symmetry nets connected Inst need to be all grouped into symmetry pairs. The current implementation is very naive and only checks that pin numbers are equal.

See also

NetPair
checkNetSym

Parameters

netId1	Id of Net1.
netId2	Id of Net2.
netPair	Library for symmetry nets.

3.13.3.28 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

instId1	Reached pair instId1
instId2	Reached pair instId2
srchPinId1	instId1 reached by srchPinId1.
srchPinId2	instId2 reached by srchPinId2.

3.13.4 Member Data Documentation

3.13.4.1 _biasGroup

```
std::vector<Bias> SymDetect::_biasGroup  [private]
```

3.13.4.2 _flatPair

```
std::vector<MosPair> SymDetect::_flatPair  [private]
```

3.13.4.3 _netlist

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

3.13.4.4 _pattern

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

3.13.4.5 _symGroup

```
std::vector<std::vector<MosPair> > SymDetect::_symGroup  [private]
```

Symmetry groups of netlist.

3.13.4.6 _symNet

```
std::vector<NetPair> SymDetect::_symNet  [private]
```

Symmetry nets of netlist.

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

- src/sym_detect/SymDetect.h
- src/sym_detect/SymDetect.cpp

Chapter 4

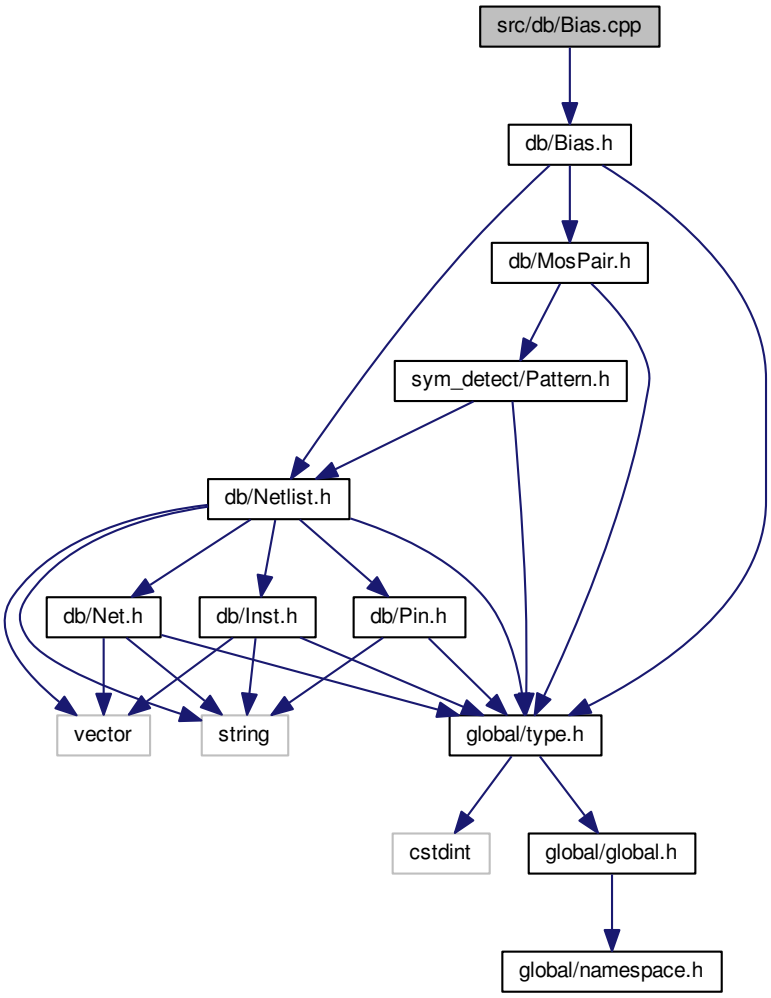
File Documentation

4.1 src/db/Bias.cpp File Reference

Bias implementation.

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

Include dependency graph for Bias.cpp:



4.1.1 Detailed Description

Bias implementation.

Author

Mingjie Liu

Date

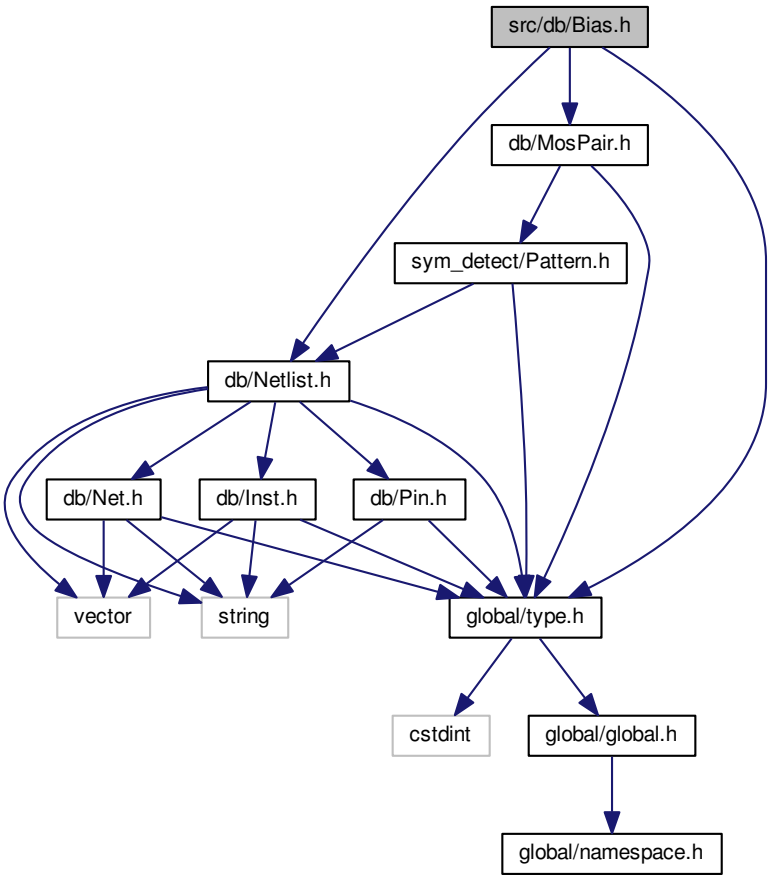
12/11/2018

4.2 src/db/Bias.h File Reference

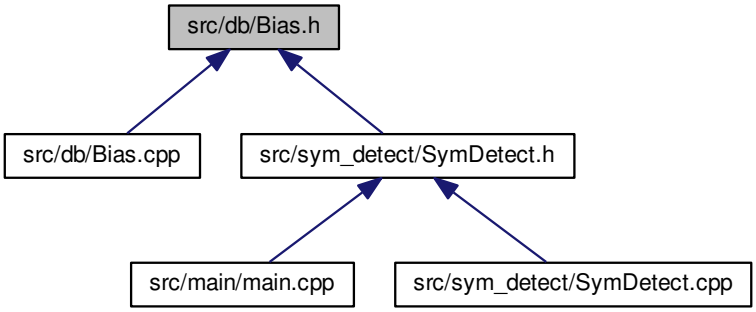
A vector of Mosfet Bias.

```
#include "global/type.h"
#include "db/Netlist.h"
#include "db/MosPair.h"
```

Include dependency graph for Bias.h:



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



Classes

- class Bias

  A vector of Mosfet.

4.2.1 Detailed Description

A vector of Mosfet Bias.

Author

Mingjie Liu

Date

12/11/2018

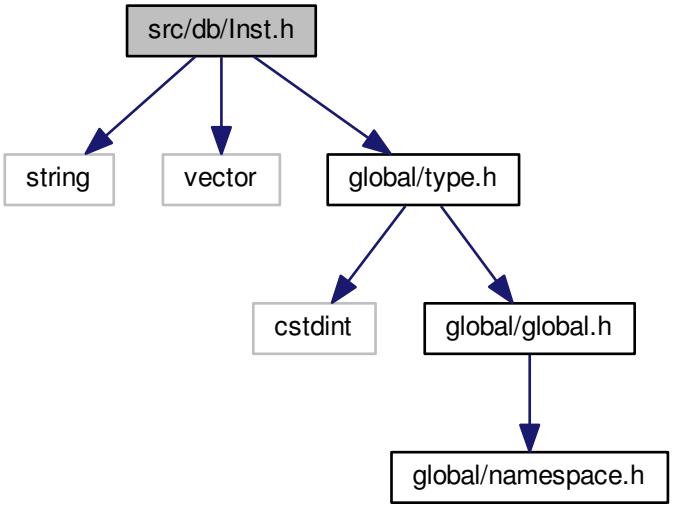
4.3 src/db/Inst.h File Reference

Instance class.

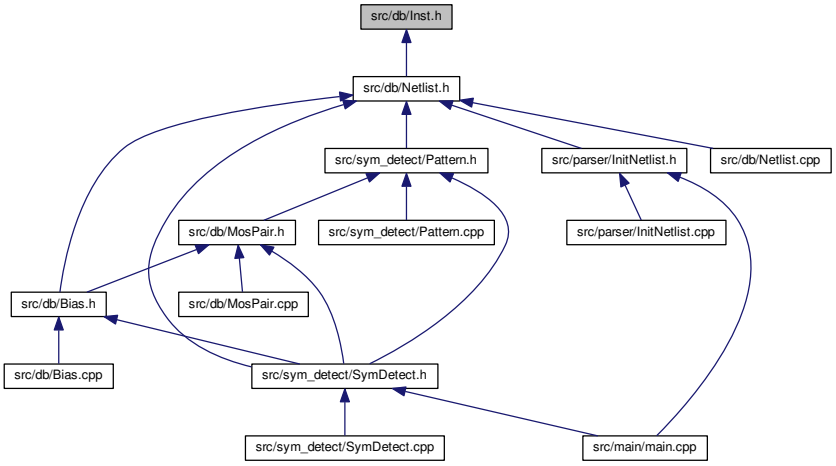
```
#include <string>
#include <vector>
```

```
#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.

4.3.1 Detailed Description

Instance class.

Author

Mingjie Liu

Date

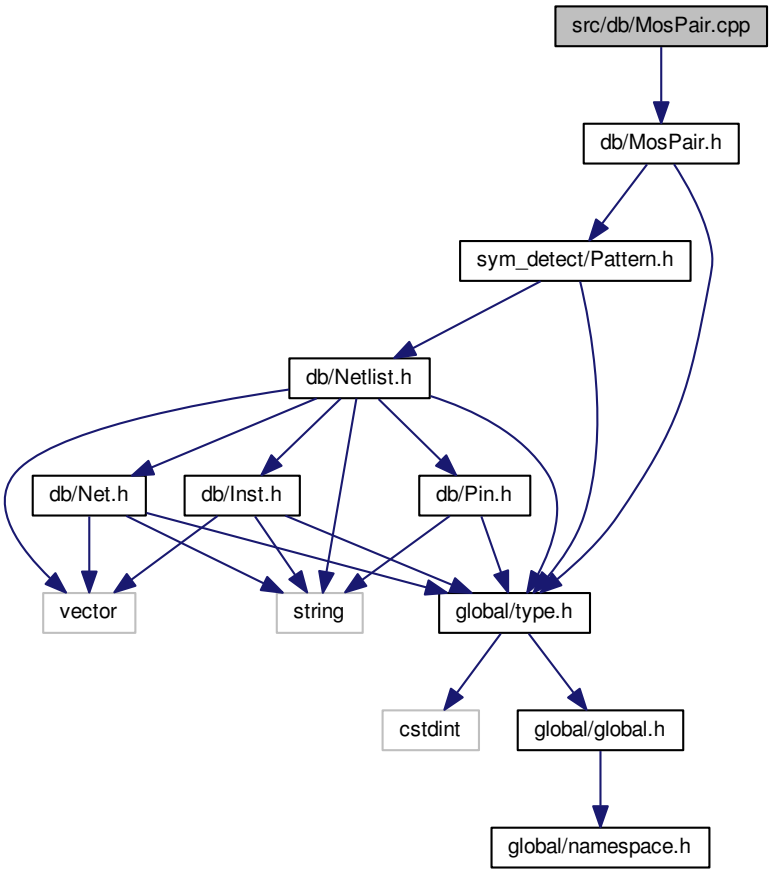
11/24/2018

4.4 src/db/MosPair.cpp File Reference

MosPair implementation.

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

Include dependency graph for MosPair.cpp:



4.4.1 Detailed Description

MosPair implementation.

Author

Mingjie Liu

Date

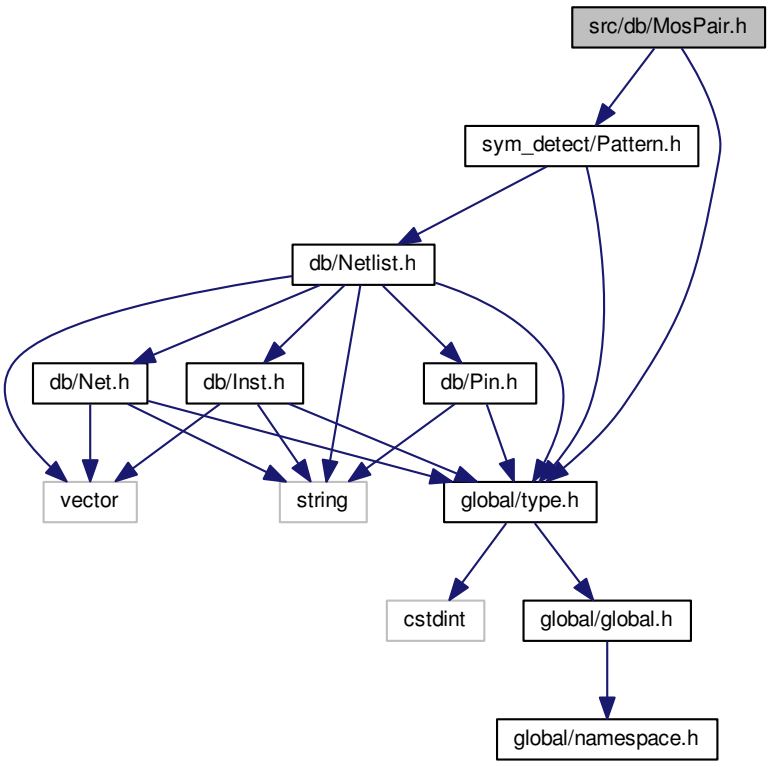
11/27/2018

4.5 src/db/MosPair.h File Reference

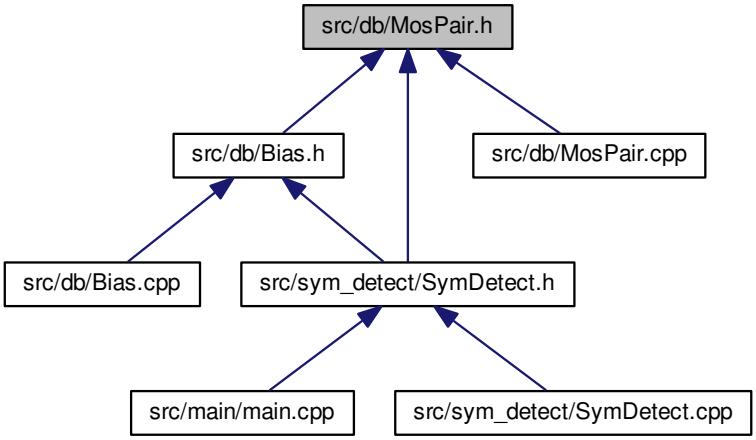
A pair of Mosfet with MosPattern.

```
#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.

4.5.1 Detailed Description

A pair of Mosfet with MosPattern.

Author

Mingjie Liu

Date

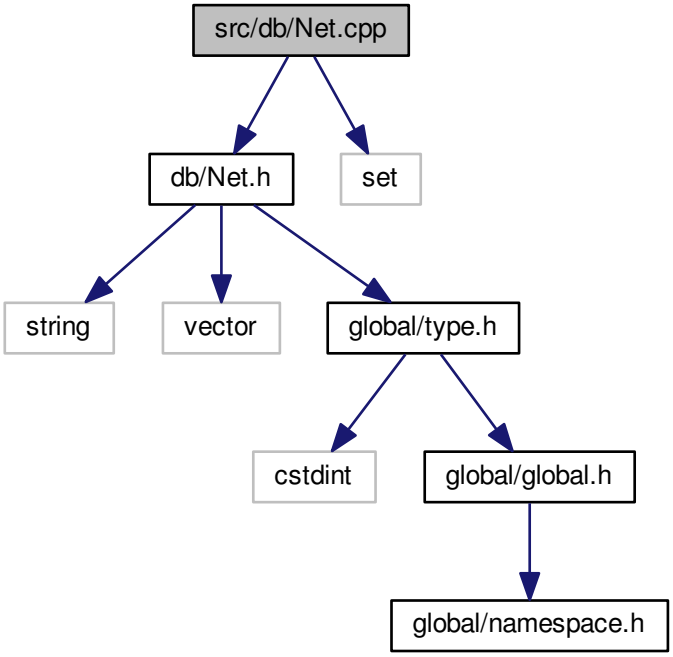
11/27/2018

4.6 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", "V↩DD", "Vdd", "VDDA", "vdda", "Vdda"}
- static const std::set< std::string > GROUND_NET_NAMES = {"vss", "VSS", "Vss", "VSSA", "vssa", "Vssa", "gnd", "Gnd", "GND"}

4.6.1 Detailed Description

Net class implementation.

Author

Mingjie Liu

Date

11/24/2018

4.6.2 Variable Documentation

4.6.2.1 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.

4.6.2.2 POWER_NET_NAMES

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

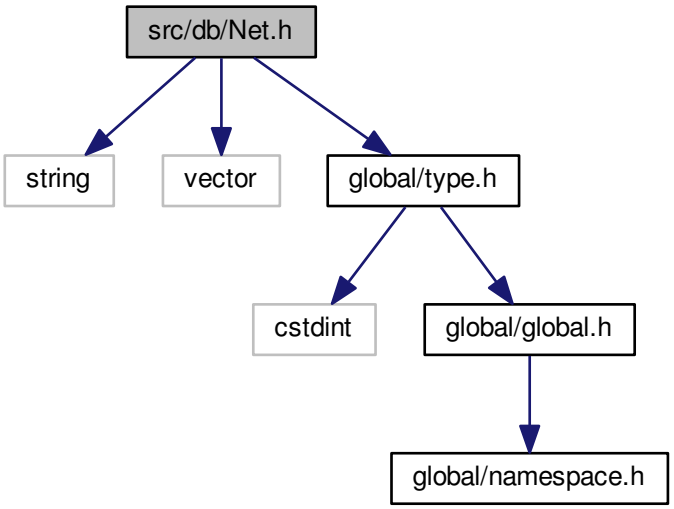
A set of possible power net names.

4.7 src/db/Net.h File Reference

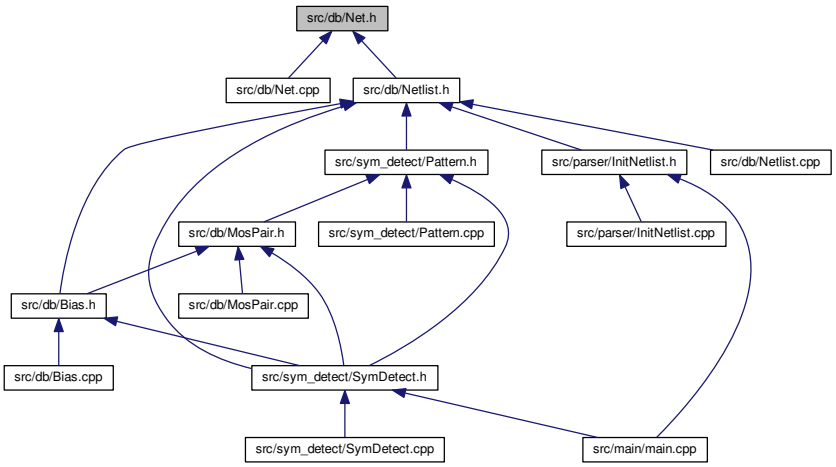
Net class.

```
#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.

4.7.1 Detailed Description

Net class.

Author

Mingjie Llu

Date

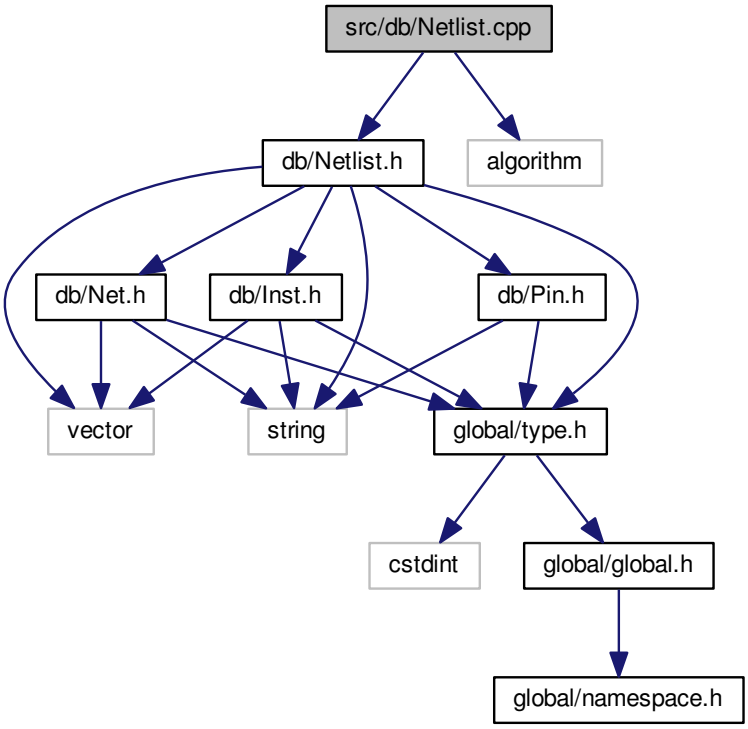
11/24/2018

4.8 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.

4.8.1 Detailed Description

Netlist class implementation.

Author

Mingjie Liu

Date

11/24/2018

4.8.2 Variable Documentation

4.8.2.1 MOS_PIN_TYPE

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

Mos Pin Types.

4.8.2.2 RES_PIN_TYPE

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

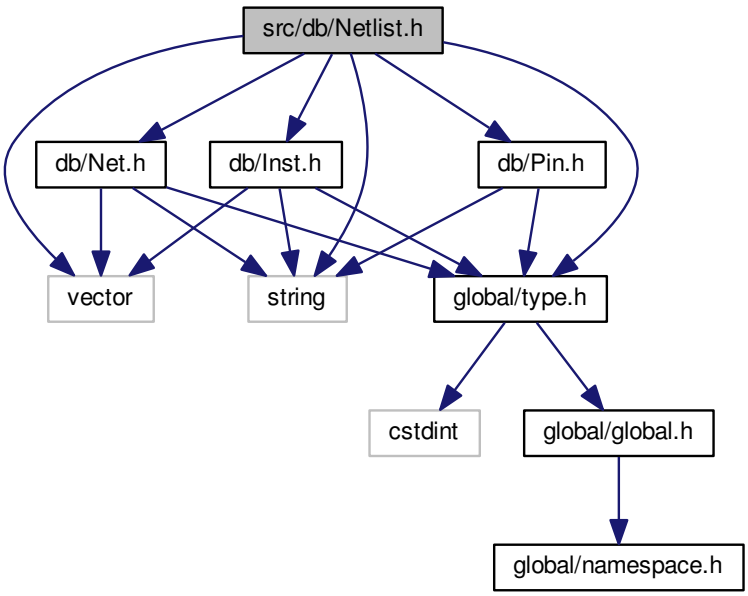
Res/Cap Pin Types.

4.9 src/db/Netlist.h File Reference

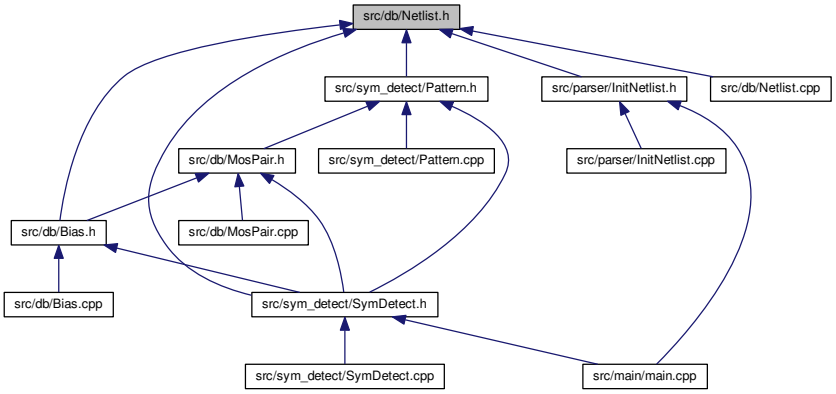
Netlist class.

```
#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.

4.9.1 Detailed Description

Netlist class.

Author

Mingjie Liu

Date

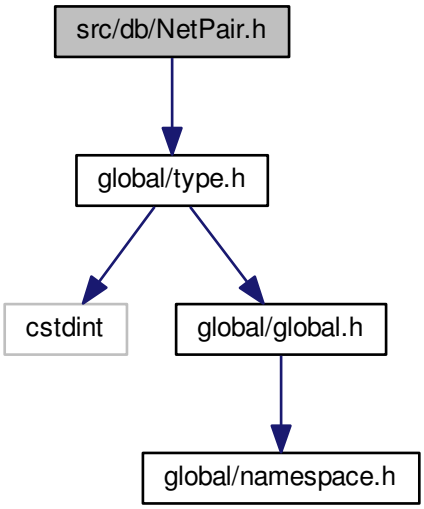
11/24/2018

4.10 src/db/NetPair.h File Reference

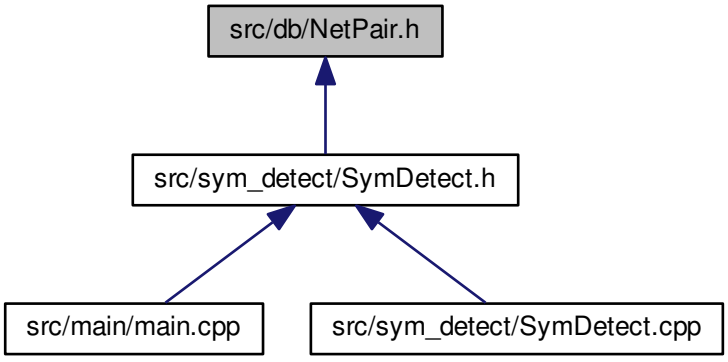
A pair of symmetry nets.

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

Include dependency graph for NetPair.h:



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



Classes

- class NetPair

  A pair of Net that are symmetric.

4.10.1 Detailed Description

A pair of symmetry nets.

Author

Mingjie Liu

Date

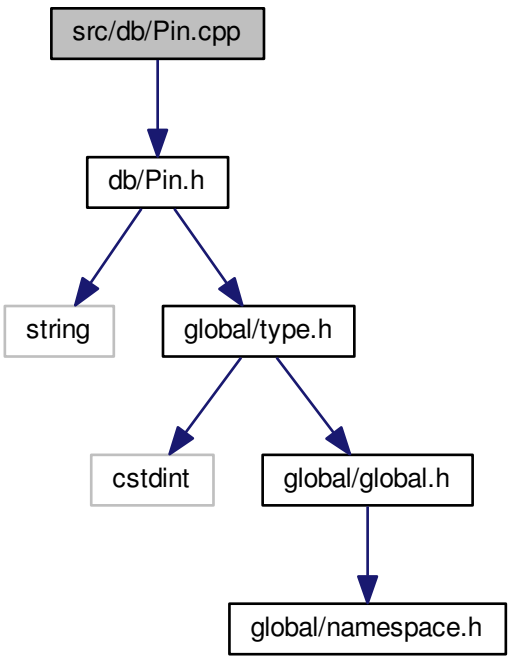
12/06/2018

4.11 src/db/Pin.cpp File Reference

Net class implementation.

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

Include dependency graph for Pin.cpp:



4.11.1 Detailed Description

Net class implementation.

Author

Mingjie Liu

Date

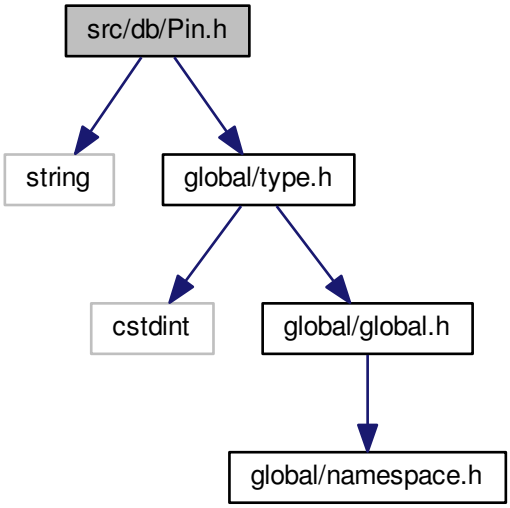
11/24/2018

4.12 src/db/Pin.h File Reference

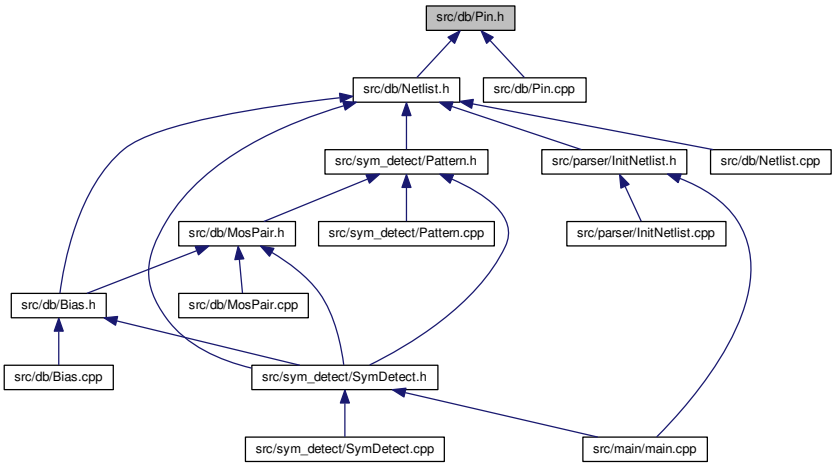
Pin class.

```
#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.

4.12.1 Detailed Description

Pin class.

Author

Mingjie Liu

Date

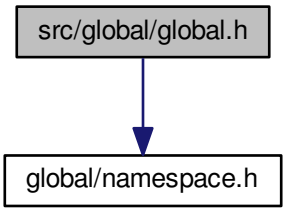
11/24/2018

4.13 src/global/global.h File Reference

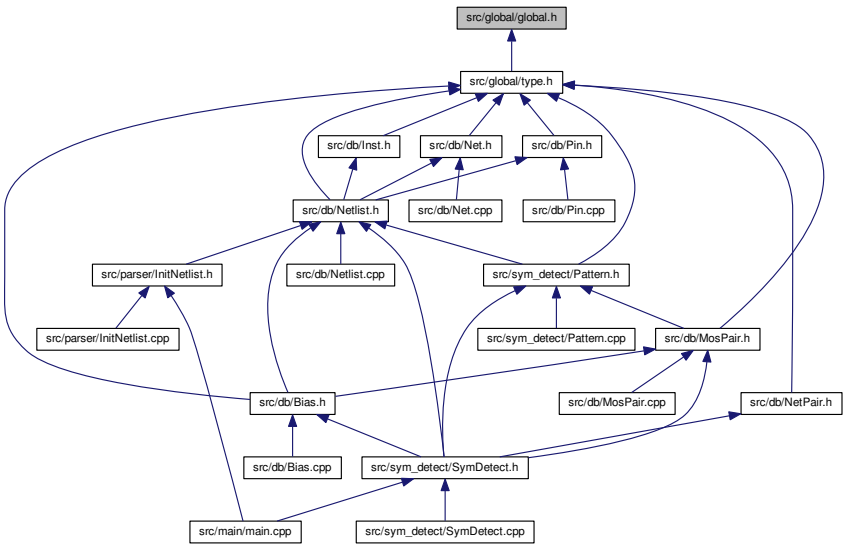
Global header file.

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

Include dependency graph for global.h:



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



4.13.1 Detailed Description

Global header file.

Author

Mingjie Liu

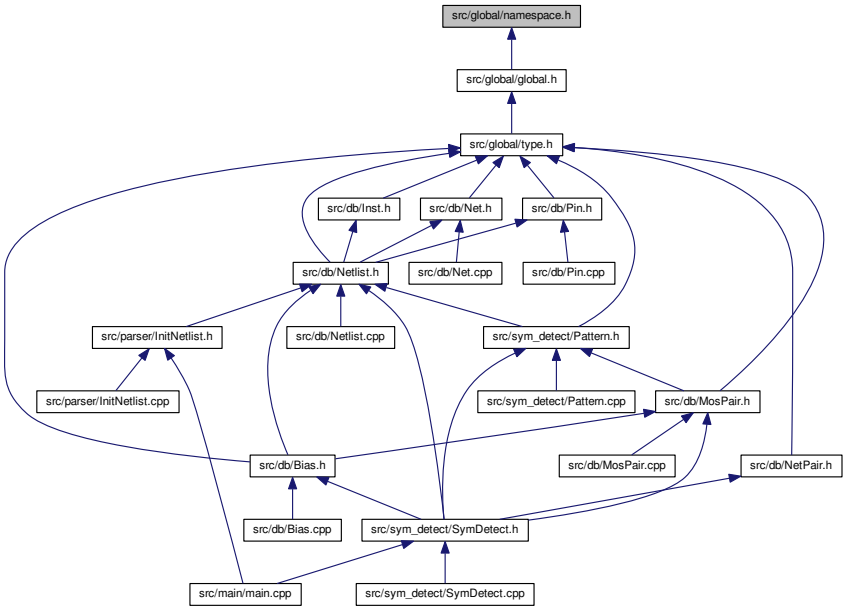
Date

11/24/2018

4.14 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 }

4.14.1 Detailed Description

Namespace header file.

Author

Mingjie Liu

Date

11/24/2018

4.14.2 Macro Definition Documentation

4.14.2.1 PROJECT_NAMESPACE

```
#define PROJECT_NAMESPACE SFA
```

4.14.2.2 PROJECT_NAMESPACE_BEGIN

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

4.14.2.3 PROJECT_NAMESPACE_END

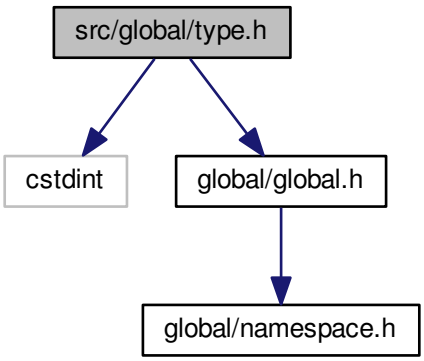
```
#define PROJECT_NAMESPACE_END }
```

4.15 src/global/type.h File Reference

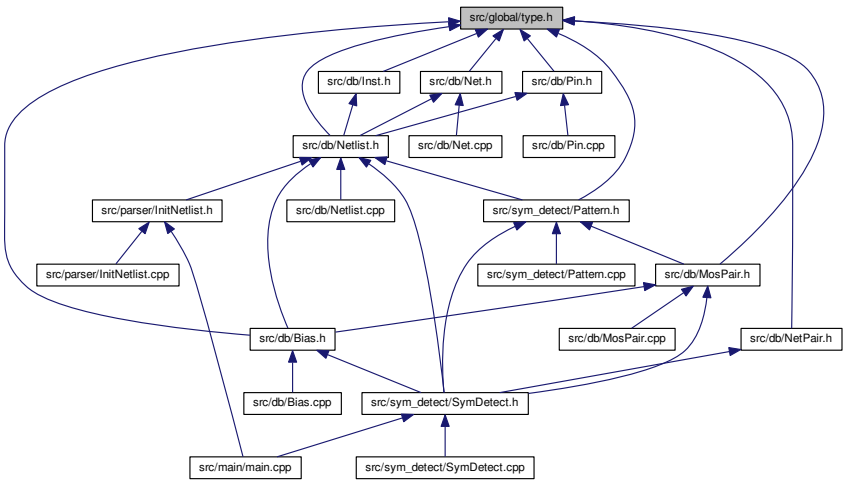
Type header file.

```
#include <cstdint>
#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_CASCODE, MosPattern::CASCODE, MosPattern::LOAD,
  MosPattern::CROSS_CASCODE, MosPattern::CROSS_LOAD, MosPattern::PASSIVE, MosPattern::SELF,
  MosPattern::BIAS, 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

4.15.1 Detailed Description

Type header file.

Author

Mingjie Liu

Date

11/24/2018

4.15.2 Typedef Documentation

4.15.2.1 Byte

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

4.15.2.2 IndexType

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

4.15.2.3 IntType

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

4.15.2.4 RealType

```
using RealType = double
```

4.15.3 Enumeration Type Documentation

4.15.3.1 InstType

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

Type of Inst.

Enumerator

RES	Resistor
PMOS	PMos
NMOS	NMos
CAP	Capacitor
OTHER	Other

4.15.3.2 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 Inst.
BIAS	Bias symmetry pair.
INVALID	No pattern detected.

4.15.3.3 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

4.15.3.4 NetType

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

Type of Net.

Enumerator

POWER	Power
GROUND	Ground
SIGNAL	Signal

4.15.3.5 PinType

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

Type of Pin.

Enumerator

SOURCE	Inst is Mosfet
DRAIN	Inst is Mosfet
GATE	Inst is Mosfet
BULK	Inst is Mosfet
THIS	Inst is Passive
THAT	Inst is Passive
OTHER	Other

4.15.4 Variable Documentation

4.15.4.1 INDEX_TYPE_MAX

```
constexpr IndexType INDEX_TYPE_MAX = 1000000000
```

4.15.4.2 INT_TYPE_MAX

```
constexpr IntType INT_TYPE_MAX = 1000000000
```

4.15.4.3 INT_TYPE_MIN

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

4.15.4.4 REAL_TYPE_MAX

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

4.15.4.5 REAL_TYPE_MIN

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

4.15.4.6 REAL_TYPE_TOL

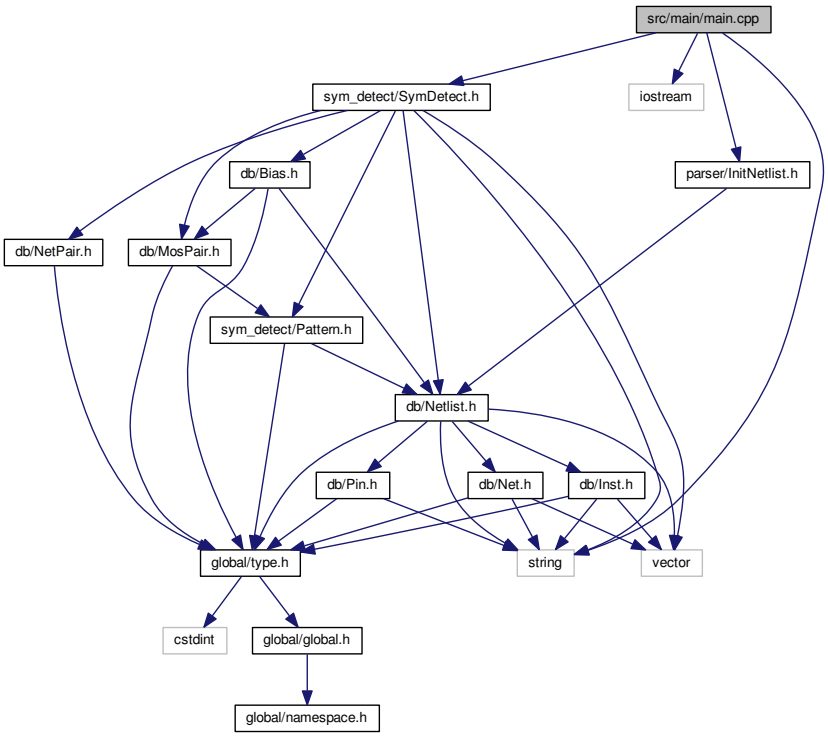
```
constexpr RealType REAL_TYPE_TOL = 1e-6
```

4.16 src/main/main.cpp File Reference

main.cpp

```
#include <string>
#include <iostream>
#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[ ])

4.16.1 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.

4.16.2 Macro Definition Documentation

4.16.2.1 __SFA_TEST__

```
#define __SFA_TEST__
```

4.16.3 Function Documentation

4.16.3.1 main()

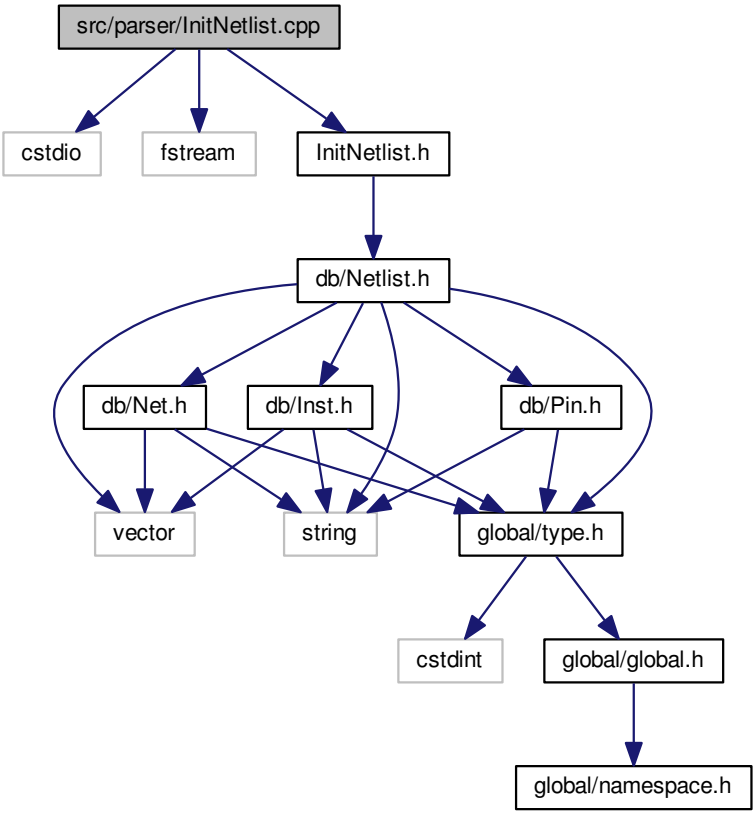
```
int main (
            int argc,
            char * argv[] )
```

4.17 src/parser/InitNetlist.cpp File Reference

Parser implementation.

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

Include dependency graph for InitNetlist.cpp:



4.17.1 Detailed Description

Parser implementation.

Author

Mingjie Liu

Date

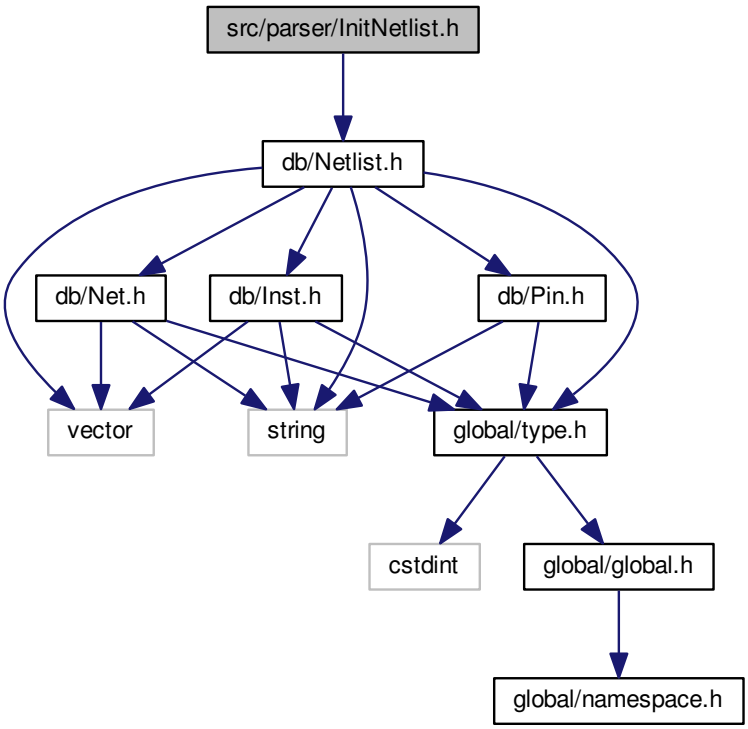
11/24/2018

4.18 src/parser/InitNetlist.h File Reference

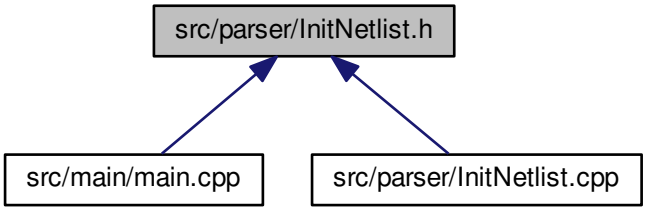
Parser to initialize netlist.

```
#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.

4.18.1 Detailed Description

Parser to initialize netlist.

Author

Mingjie Liu

Date

11/24/2018

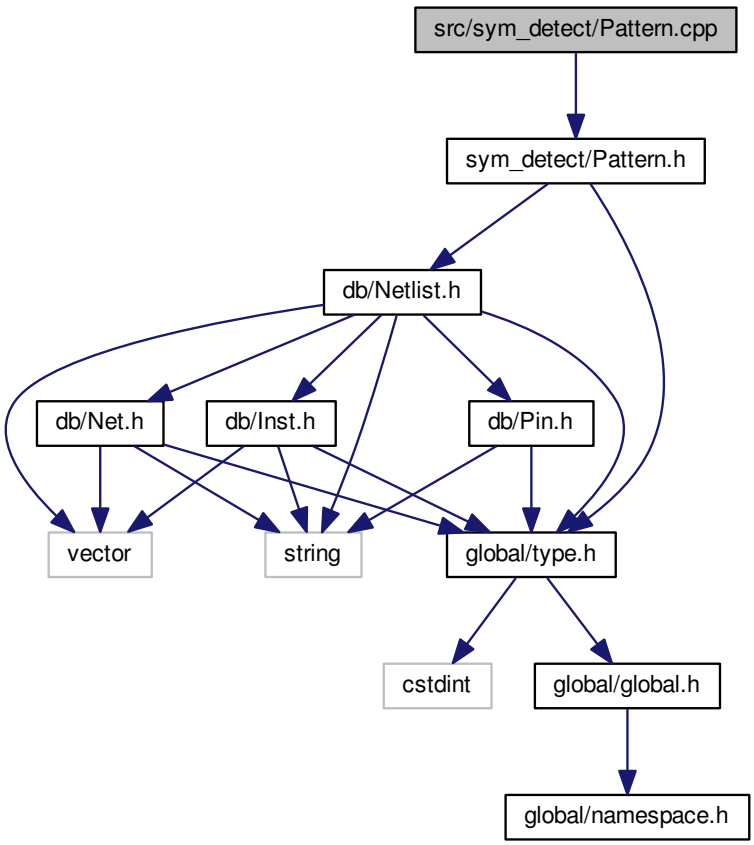
Input file should follow same format generated through scripts/create_init_obj.py. The python scripts take standardized hspice/spectre netlist files as inputs. Sample input files for c++ are under benchmarks.

4.19 src/sym_detect/Pattern.cpp File Reference

Pattern definitions.

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

Include dependency graph for Pattern.cpp:



4.19.1 Detailed Description

Pattern definitions.

Author

Mingjie Liu

Date

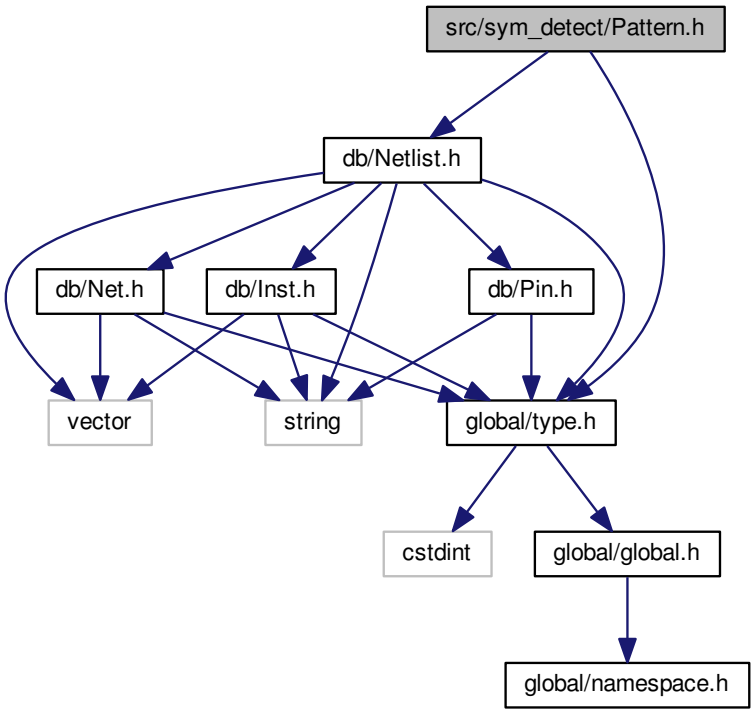
11/24/2018

4.20 src/sym_detect/Pattern.h File Reference

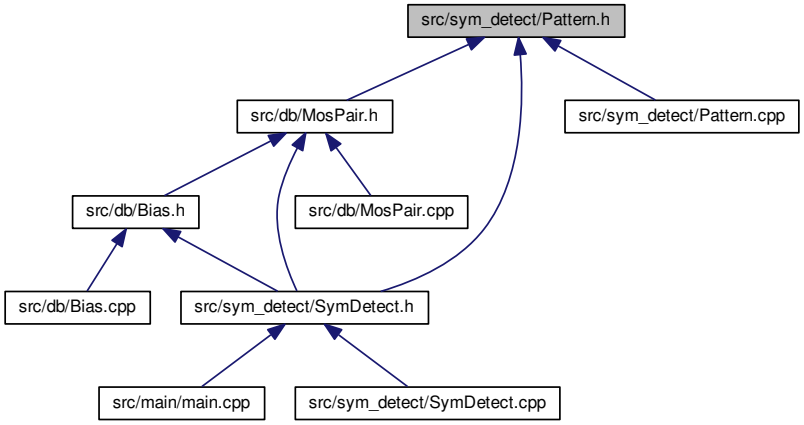
Mosfet pair patterns.

```
#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.

4.20.1 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

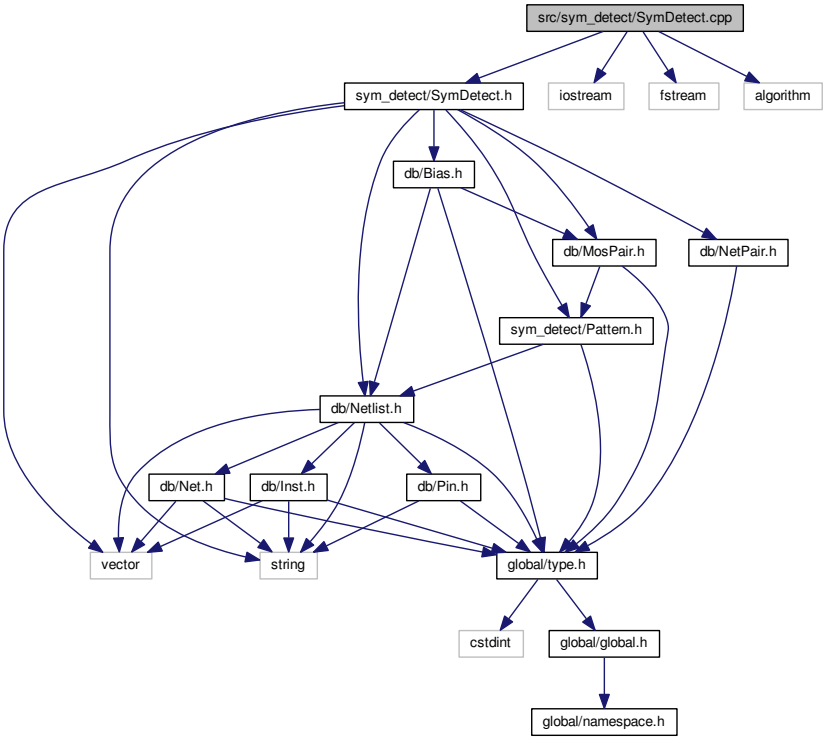
4.21 src/sym_detect/SymDetect.cpp File Reference

Detect symmetric patterns.

```
#include "sym_detect/SymDetect.h"
#include <iostream>
#include <fstream>
```

```
#include <algorithm>
```

Include dependency graph for SymDetect.cpp:



4.21.1 Detailed Description

Detect symmetric patterns.

Author

Mingjie Liu

Date

11/24/2018

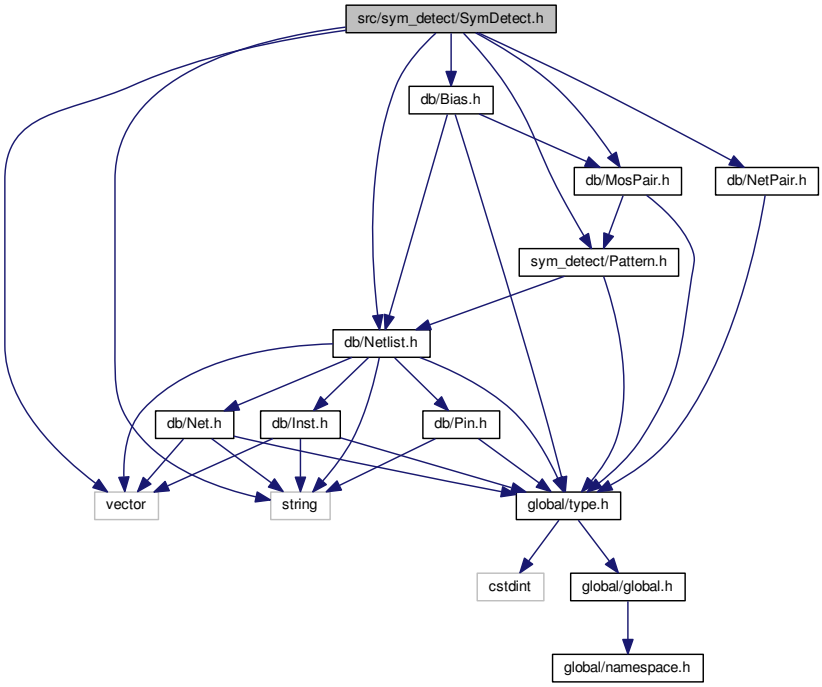
4.22 src/sym_detect/SymDetect.h File Reference

Detect symmetric patterns.

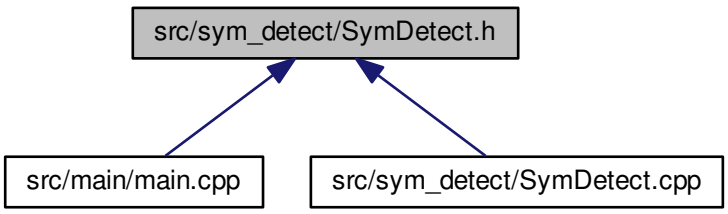
```
#include "db/Netlist.h"
#include "db/MosPair.h"
#include "db/NetPair.h"
#include "db/Bias.h"
#include "sym_detect/Pattern.h"
```

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

Include dependency graph for SymDetect.h:



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



Classes

- class SymDetect

  SymDetect class.

4.22.1 Detailed Description

Detect symmetric patterns.

Author

Mingjie Liu

Date

11/24/2018

Index