General Quadratic Placer

Generated by Doxygen 1.8.11

Fri May 13 2016 08:29:07

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 coo_matrix Class Reference

`#include <solver.h>`

Public Member Functions

- void read_coo_matrix (const char *fname)
- void matvec (const valarray< double > &x, valarray< double > &y)
- void solve (const valarray< double > &b, valarray< double > &x)

Public Attributes

- int n
- int nnz
- valarray< int > row
- valarray< int > col
- valarray< double > dat

3.1.1 Member Function Documentation

3.1.1.1 void coo_matrix::matvec (const valarray< double > & *x*, valarray< double > & *y*)

3.1.1.2 void coo_matrix::read_coo_matrix (const char * *fname*)

3.1.1.3 void coo_matrix::solve (const valarray< double > & *b*, valarray< double > & *x*)

3.1.2 Member Data Documentation

3.1.2.1 valarray<int> coo_matrix::col

3.1.2.2 valarray<double> coo_matrix::dat

3.1.2.3 int coo_matrix::n

3.1.2.4 int coo_matrix::nnz

3.1.2.5 valarray<int> coo_matrix::row

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

- solver.h
- solver.cpp

3.2 mothercore Class Reference

Public Member Functions

- mothercore ()
- int get_numG ()
- vi get_gateKeys ()
- vd get_gateCoords (int gateNum)
- vi get_gateconnections (int gateNum)
- void add_gate (int gateNum, vi listofconnections)
- int get_numP ()
- vi get_padKeys ()
- vd get_padCoords (int padNum)
- void add_pad (int padNum, vd netandlocation)
- int get_numN ()
- vi get_netKeys ()
- int get_numNetConns (int netNum)
- vi get_netGateConns (int netNum)
- vi get_netPadConns (int netNum)
- void add_net (int netNum, int connection, int gateorpad)
- bool add_location (vd x, vd y, vi gatekeys, int bound[4])
- vvd get_locations (vi gatekeys)
- void print_all_locations ()
- void print_all_pads ()

Private Attributes

- int numG
- int numP
- int numN
- map< int, vi > gate
- map< int, vd > pad
- map< int, vvi > nets
- map< int, double > gateX
- map< int, double > gateY

3.2.1 Detailed Description

Class which defines an ASIC and its components.

3.2.2 Constructor & Destructor Documentation

3.2.2.1 mothercore::mothercore ( ) [inline]

Constructor of the class 'mothercore'

3.2.3 Member Function Documentation

3.2.3.1 void mothercore::add_gate (int *gateNum,* vi *listofconnections*) [inline]

Helper function which makes a new gate and adds list of connections.

Parameters

gateNum	The gate-id of the gate to be added.
listofconnections	A vector of net-ids connected to the gate to be added.

See also

add_pad(int padNum, vd netandlocation)
add_net(int netNum, int connection, int gateorpad)

3.2.3.2 bool mothercore::add_location (vd *x,* vd *y,* vi *gatekeys,* int *bound[4]*) [inline]

Helper function which adds location values for given gate keys

Parameters

x	A vector containing x-coordinates of gates in the same order as the gate-ids in the vector 'gatekeys'
y	A vector containing y-coordinates of gates in the same order as the gate-ids in the vector 'gatekeys'
gatekeys	A vector containing gate-ids of the gates for which location is given and to be updated.
bound	The minimum and maximum values of the x, y coordinates desired. It is an array of 4 numbers, [x_min, x_max, y_min, y_max]. The argument is not necessarily used.

See also

get_locations(vi gatekeys)

Returns

True if the operation is successful, false otherwise. It is false if there is a mismatch amongst sizes of 'x', 'y', and 'gatekeys'.

3.2.3.3 void mothercore::add_net (int *netNum,* int *connection,* int *gateorpad*) [inline]

Helper function which makes a new net, if needed, and appends a connection to the net 'netnum'

Parameters

netNum	The net-id of the net to be added.
connection	The gate-id/ pad-id of the gate/pad to which the net is connected to.
gateorpad	It shows if the given connection is a gate or a pad. It is 0 for gate, 1 for pad.

See also

add_gate(int gateNum, vi listofconnections)
add_pad(int padNum, vd netandlocation)

3.2.3.4 void mothercore::add_pad (int *padNum*, vd *netandlocation*) [inline]

Helper function which makes a new pad and adds its connections and location

Parameters

padNum	The pad-id of the pad to be added.
netandlocation	A vector with net-id connected to the pad to be added, and its x, y coordinate.

See also

add_gate(int gateNum, vi listofconnections)
add_net(int netNum, int connection, int gateorpad)

3.2.3.5 vi mothercore::get_gateconnections (int *gateNum*) [inline]

Helper function to return connections of a gate.

Parameters

gateNum	The gate-id of the gate for which connections are needed.

Returns

A vector of the net-ids connected to the gate with gate-id 'gateNum'.

3.2.3.6 vd mothercore::get_gateCoords (int *gateNum*) [inline]

Helper function to return gate coordinates.

Parameters

gateNum	The gate-id for which coordinates are needed.

See also

get_padCoords(int padNum)

Returns

The x, y coordinates of the gate with gate-id 'gateNum'.

3.2.3.7 vi mothercore::get_gateKeys ( ) [inline]

Helper function to return gate keys.

See also

get_padKeys()
get_netKeys()

Returns

A vector with the gate-id of gates in the ASIC.

3.2.3.8 vvd mothercore::get_locations (vi *gatekeys*) [inline]

Helper function which gets the location values for given gate keys

Parameters

gatekeys	A vector containing gate-ids of the gates for which location data is needed.

See also

add_location(vd x, vd y, vi gatekeys, int bound[4])

Returns

A vector of x, y coordinates for each gate-id in vector 'gatekeys' , in the same order as the gate-ids in 'gatekeys'.

3.2.3.9 vi mothercore::get_netGateConns (int *netNum*) [inline]

Helper function to return gate connections to a net

Parameters

netNum	The net-id of the net for which the information is desired.

See also

get_netPadConns(int netNum)

Returns

A vector of gate-ids of the gates connected to the net of net-id 'netNum'.

3.2.3.10 vi mothercore::get_netKeys ( ) [inline]

Helper function to return net keys.

See also

get_gateKeys()
get_padKeys()

Returns

A vector with the net-id of nets in the ASIC.

3.2.3.11 vi mothercore::get_netPadConns (int *netNum*) [inline]

Helper function to return pad connections to a net

Parameters

netNum	The net-id of the net for which the information is desired.

See also

get_netGateConns(int netNum)

Returns

A vector of pad-ids of the pads connected to the net of net-id 'netNum'.

3.2.3.12 int mothercore::get_numG ( ) [inline]

Helper function to return the number of gates.

See also

get_numP()
get_numN()

Returns

The number of gates in the ASIC.

3.2.3.13 int mothercore::get_numN ( ) [inline]

Helper function to return the number of nets.

See also

get_numG()
get_numP()

Returns

The number of nets in the ASIC.

3.2.3.14 int mothercore::get_numNetConns (int *netNum*) [inline]

Helper function to return number of net connections

Parameters

netNum	The net-id of the net for which the information is desired.

Returns

Total number of gates and pads, combined, connected to a net of net-id 'netNum'.

3.2.3.15 int mothercore::get_numP ( ) [inline]

Helper function to return the number of pads.

See also

get_numG()
get_numN()

Returns

The number of pads in the ASIC.

3.2.3.16 vd mothercore::get_padCoords (int *padNum*) [inline]

Helper function to return pad coordinates.

Parameters

padNum	The pad-id of the pad for which coordinates are needed.

See also

get_gateCoords(int gateNum)

Returns

The x, y coordinates of the pad with pad-id 'padNum'.

3.2.3.17 vi mothercore::get_padKeys ( ) [inline]

Helper function to return pad keys.

See also

get_gateKeys()
get_netKeys()

Returns

A vector with the pad-id of pads in the ASIC.

3.2.3.18 void mothercore::print_all_locations ( ) [inline]

Helper function which prints the locations of all gates in the present core.

See also

print_all_pads()

3.2.3.19 void mothercore::print_all_pads ( ) [inline]

Helper function which prints the pads in the present core.

See also

print_all_locations()

3.2.4 Member Data Documentation

3.2.4.1 map<int, vi > mothercore::gate [private]

A Hash Table storing informations about gates in the ASIC. It lists the net ids connected to each gate.

3.2.4.2 map<int, double> mothercore::gateX [private]

A Hash Table storing x-coordinates of gates in the ASIC.

3.2.4.3 map<int, double> mothercore::gateY [private]

A Hash Table storing y-coordinates of gates in the ASIC.

3.2.4.4 map<int, vvi> mothercore::nets [private]

A Hash Table storing informations about nets in the ASIC. One vector consists of the gate ids connected by this net. Other vector consists of the net ids connected by this net.

3.2.4.5 int mothercore::numG [private]

Number of gates in the ASIC.

3.2.4.6 int mothercore::numN [private]

Number of nets in the ASIC.

3.2.4.7 int mothercore::numP [private]

Number of pads in the ASIC.

3.2.4.8 map<int, vd > mothercore::pad [private]

A Hash Table storing informations about pads in the ASIC. It lists the net connected to each pad, x and y coordinate of the pad. Assuming each pad is connected to only one net.

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

- qplacer.cpp

Chapter 4

File Documentation

4.1 qplacer.cpp File Reference

```
#include <iostream>
#include <fstream>
#include <cstdlib>
#include <string>
#include <map>
#include <vector>
#include <cmath>
#include <valarray>
#include <cstdio>
#include "solver.h"
```

Classes

- class mothercore

Typedefs

- typedef vector< vector< int > > vvi
- typedef vector< vector< double > > vvd
- typedef vector< int > vi
- typedef vector< double > vd

Functions

- mothercore create (char *filename)
- void writeback (mothercore *core, char *filename)
- void writebackpads (mothercore *core, char *filename)
- vd solve (vi R, vi C, vd V, vd ba)
- bool solveforx (mothercore *core, int bound[4])
- vvi assign (mothercore *core, vi gatekeys, int hORv)
- void update_coordinates (vd *padlocation, int bound[4])
- vvd containNrun (mothercore *core, vi gatekeys, int bound[4], int hORv, int lORr)
- void place (mothercore *core, vi gatekeys, int bound[4], int n)
- int main (int argc, char *argv[ ])

4.1.1 Typedef Documentation

4.1.1.1 typedef vector<double> vd

4.1.1.2 typedef vector<int> vi

4.1.1.3 typedef vector<vector<double> > vvd

4.1.1.4 typedef vector<vector<int> > vvi

4.1.2 Function Documentation

4.1.2.1 vvi assign (mothercore * *core,* vi *gatekeys,* int *hORv*)

Function which sorts given gates according to their locations, horizontally or vertically. Horizontally if hORv = 0; Vertically if hORv = 1

Parameters

core	Pointer to an object of class "mothercore".
gatekeys	Vector of gate-ids of gates that need to be sorted.
hORv	It is used to decide if the sorting is done based on x-coordinate or y-coordinate. If hORv = 0, x-coordinate is used, and if it is 1, y-coordinate is used.

Returns

A vector of 2 vectors. First vector contains the gate-ids which are on the lower values of the sorting coordinate. The second vector contains the gate-ids which are on the higher values of the sorting coordinate.

4.1.2.2 vvd containNrun (mothercore * *core,* vi *gatekeys,* int *bound[4],* int *hORv,* int *lORr*)

Function which contains the given gate-ids within the given bound, creates virtual pads, and runs the resulting mothercore object.

Parameters

core	Pointer to an object of class "mothercore".
gatekeys	Vector of gate-ids of gates that need to be contained within the given bound.
bound	The minimum and maximum values of the x, y coordinates desired. It is an array of 4 numbers, [x_min, x_max, y_min, y_max].
hORv	It is used to correct the coordinate of some virtual pads. hORv = 0 means that the present bound is the result of a horizontal cut. hORv = 1 means that the present bound is the result of a vertical cut.
lORr	It is used to correct the coordinate of some virtual pads. lORr = 0 means that the present gate-ids are present in the lower part of a cut. lORr = 1 means that the present gate-ids are present in the higher part of a cut.

Returns

The new locations of the gates whose gate-ids are in "gatekeys".

4.1.2.3 mothercore create (char * *filename*)

Function which creates an object of class "mothercore" from a given file.

Parameters

filename	Pointer to the name of a file where the input is located.

Returns

An object of class mothercore.

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

The main function: It creates a new mothercore object from a file and runs place for recursive placing. Final placement is written back to an output file using writeback and writebackpads.

4.1.2.5 void place (mothercore * *core*, vi *gatekeys*, int *bound[4]*, int *n*)

Function which recursively calls itself to place the given gates within the bound. The bound keeps shortening as the depth of the recursion increases. Also the number of gates in each bound decreases as the depth of the recursion increases. It aims to find a uniform distribution of the gates in all the divisions of the ASIC.

Parameters

core	Pointer to an object of class "mothercore".
gatekeys	Vector of gate-ids of gates that need to be placed within the given bound.
bound	The minimum and maximum values of the x, y coordinates desired. It is an array of 4 numbers, [x_min, x_max, y_min, y_max].
n	The level of the iteration. A nth iteration means that there are 2^n divisions in the ASIC.

4.1.2.6 vd solve (vi *R*, vi *C*, vd *V*, vd *ba*)

Function which solves a sparse matrix, of the form Ax=b, using coo_matrix class from solver.h.

Parameters

R	Vector containing non-zero row values of the matrix A, in order.
C	Vector containing non-zero column values of the matrix A, in order.
V	Vector containing non-zero values of the matrix A, in order.
ba	Vector containing the b vector in the matrix form Ax=b.

Returns

A vector containing values of the solved vector x in the form Ax=b.

4.1.2.7 bool solveforx (mothercore * *core*, int *bound[4]*)

Function which generates the A matrix and b vector for each coordinate, from an object of the class mothercore and sends it to solve for solving. The result is written back to the mothercore object.

Parameters

core	Pointer to an object of class "mothercore".

Parameters

bound	The minimum and maximum values of the x, y coordinates desired. It is an array of 4 numbers, [x_min, x_max, y_min, y_max].

See also

solve(vi R, vi C, vd V, vd ba)

Returns

True if there are no errors, false otherwise.

4.1.2.8 void update_coordinates (vd * *padlocation*, int *bound[4]*)

Function which updates the coordinates of the given x, y coordinates according to the given bound.

Parameters

padlocation	Pointer to a vector containing x, y coordinates.
bound	The minimum and maximum values of the x, y coordinates desired. It is an array of 4 numbers, [x_min, x_max, y_min, y_max].

4.1.2.9 void writeback (mothercore * *core*, char * *filename*)

Function which writes the location of gates in an object of class "mothercore" to a file.

Parameters

core	Pointer to an object of class "mothercore".
filename	Pointer to the name of a file where the output is to be written.

See also

writebackpads(mothercore *core, char *filename)

4.1.2.10 void writebackpads (mothercore * *core*, char * *filename*)

Function which writes the location of pads in an object of class "mothercore" to a file.

Parameters

core	Pointer to an object of class "mothercore".
filename	Pointer to the name of a file where the output is to be written.

See also

writeback(mothercore *core, char *filename)

4.2 solver.cpp File Reference

```
#include "solver.h"
```

Functions

- double dot (const valarray< double > &x, const valarray< double > &y)

4.2.1 Function Documentation

4.2.1.1 double dot (const valarray< double > & *x*, const valarray< double > & *y*)

4.3 solver.h File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <valarray>
#include <algorithm>
#include <iostream>
#include <fstream>
```

Classes

- class coo_matrix

Functions

- template<typename T >
  void print_valarray (valarray< T > &v)

4.3.1 Function Documentation

4.3.1.1 template<typename T > void print_valarray (valarray< T > & *v*)

Index