DAC 2012 Contest
Routability-Driven Placement

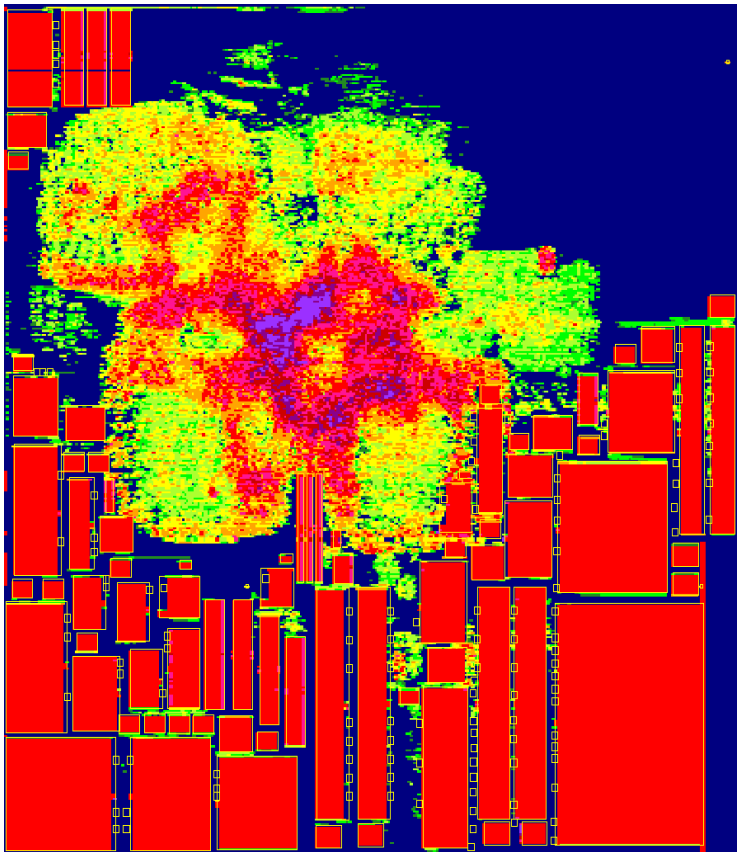
http://archive.sigda.org/dac2012/contest/dac2012_contest.html

Benchmark Description

Natarajan Viswanathan
IBM Corporation, Austin, TX
(nviswan@us.ibm.com)

Highlights

- Real industrial ASIC designs
- Information for placement and routing
- Design-density: 25% - 65%
- Placement blockages leading to a fragmented image space
- Routing blockages
- More metal layers with varying metal width and spacing across layers



Outline

- Benchmark File Format Description
- Special Features for Placement and Routing
- Utility Scripts

Benchmark File Format Description

Overview of Benchmark Files (1)

- Extend the Bookshelf format with information to perform placement and routing

- Each benchmark circuit will comprise of the following files
  - circuit.aux
  - circuit.nodes
  - circuit.nets
  - circuit.wts
  - circuit.pl
  - circuit.scl
  - circuit.shapes
  - circuit.route

Original Files in Bookshelf format with some extensions

New Files with extensions for both placement and routing

Overview of Benchmark Files (2)

- The output/solution of the placer should have the same format as the circuit.pl file
- Hence, placement output/solution file
  - <placement_solution>.pl
- For additional information:
  - http://vlsicad.ucsd.edu/GSRC/bookshelf/Slots/Placement/plFormats.html
  - http://archive.sigda.org/ispd2008/contests/ispd08rc.html

circuit.aux

- Auxiliary file listing all the files that describe/specify the benchmark
- The placer/router should parse the files listed in the "circuit.aux" file to get the benchmark information
- Single line giving all the file names

```
RowBasedPlacement : circuit.nodes  circuit.nets
circuit.wts  circuit.pl  circuit.scl  circuit.shapes
circuit.route
```

circuit.nodes (1)

- Specifies the node (object) name, dimensions (width and height) and movetype

- The nodes can have one of three movetypes
  - **movable:** Movable Node – the placer needs to obtain the locations of these nodes.
  - **terminal:** Fixed Node – the placer cannot move these nodes. Also, there should be no overlap between a movable and terminal node.
  - **terminal_NI:** Fixed "Not in Image" Node – the placer cannot move these nodes, but overlap is allowed with a terminal_NI node. (detailed description under special features)

circuit.nodes (2)

- NumNodes : Total number of nodes (movable + fixed)
- NumTerminals : Number of terminal (fixed) nodes
  - NumTerminals = #terminal + #terminal_NI
- For each node:

  node_name width height movetype
  - If a line does not specify a movetype, the associated node is a movable node

```
UCLA nodes 1.0
# File header with version information, etc.
# Anything following "#" is a comment, and should be ignored

NumNodes      :  5
NumTerminals  :  2

    o0      4       9                    # movable node
    o1      6       9
    o2     24       9
    o3    414    2007   terminal         # terminal node (fixed node)
    p0      1       1   terminal_NI      # terminal_NI node (fixed node, but
                                           overlap is allowed with this node)
```

circuit.nets (1)

- Specifies the circuit netlist – the set of nets or connections in the hypergraph
- Each net specification lists the pins that belong to the net
- A pin is specified by
  - The corresponding node
  - The offset of the pin with respect to the center of the node
- For wirelength driven placement, the pin direction can be ignored

circuit.nets (2)

- NumNets : Total number of nets in the circuit
- NumPins : Total number of pins in the netlist
- For each net:

NetDegree : number_of_pins_on_this_net [net_name]

node_name pin_direction : pin_Xoffset pin_Yoffset

- Pin offsets are from the center of the node

```
UCLA nets 1.0
# File header with version information, etc.
# Anything following "#" is a comment, and should be ignored

NumNets  :  2
NumPins  :  6

NetDegree  :  3    n0
        o0   I  :     0.0000     -1.5000
        o1   I  :    -2.5000      0.5000
        p1   O  :     0.0000      0.0000
NetDegree  :  3    n1
        o0   O  :     1.5000      3.0000
        o3   O  :    10.5000    -27.0000
        o2   I  :    -1.0000      0.5000
```

circuit.pl (1)

- Gives the coordinates (x,y) and orientation for each node
- The coordinates for all movable nodes will be (0,0) or undefined
- The placer should parse this file to obtain the coordinates for all the fixed nodes
- The default orientation is “vertical and face up” – **N** (North)
- NOTE: The output/solution of the placer should have the same format as the “circuit.pl” file

circuit.pl (2)

- For each node:

  node_name lowerleft_Xcoordinate lowerleft_Ycoordinate

  : orientation movetype

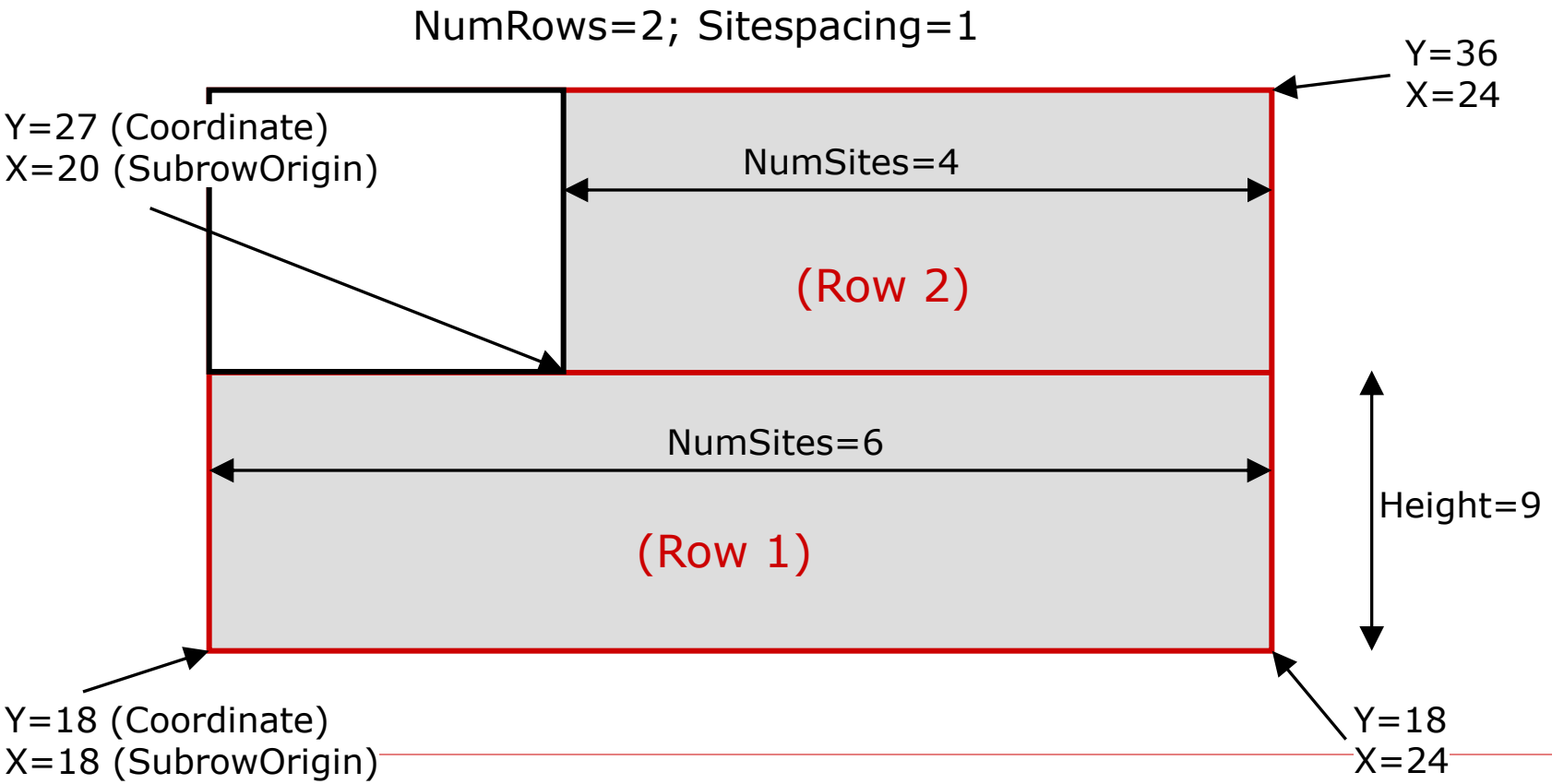
- Orientation of all the nodes will always be N (default)
  - No flipping / mirroring / rotation of the nodes is allowed
  - Use pin offsets directly as specified in the .nets file

```
UCLA pl 1.0
# File header with version information, etc.
# Anything following "#" is a comment, and should be ignored

# node_name  ll_Xcoord  ll_Ycoord       orientation    movetype
    o0             0           0    :          N
    o1             0           0    :          N
    o2             0           0    :          N
    o3          7831        7452    :          N         /FIXED
    p0          1215        7047    :          N         /FIXED_NI
```

circuit.scl (1)

- Specifies the placement image (individual circuit rows for standard-cell placement)
- Refer to the next slide for file format and definitions



circuit.scl (2)

- NumRows : Number of circuit rows for placement
- CoreRow – Horizontal circuit row followed by the row specification
  - Coordinate : **Y**-coordinate of the bottom edge of the circuit row
  - Height : Circuit row height (= standard-cell height)
  - Sitespacing : Absolute distance between neighboring placement sites in a row
  - SubrowOrigin : **X**-coordinate of the left edge of the subrow
  - NumSites : Number of placement sites in this subrow
  - Hence, **X**-coordinate of the right edge of the subrow = **SubrowOrigin + NumSites*Sitespacing**

```
UCLA scl 1.0
# File header with version information, etc.

NumRows  :  1

CoreRow Horizontal
   Coordinate     :  18
   Height         :  9
   Sitewidth      :  1                    # optional: equal to Sitespacing
   Sitespacing    :  1
   Siteorient     :  N                    # optional: can be ignored
   Sitesymmetry   :  Y                    # optional: can be ignored
   SubrowOrigin   :  18   NumSites  :  11605
End
```

circuit.shapes

- Specifies the component shapes for non-rectangular nodes
  (detailed description of non-rectangular nodes under special features)
- Any node not in this file is a regular rectangular node
- NumNonRectangularNodes : Number of non-rectangular nodes
- For each non-rectangular node:

  node_name : number_of_component_shapes

  shape_id lowerleft_Xcoord lowerleft_Ycoord width height

```
shapes 1.0
# File header with version information, etc.

NumNonRectangularNodes  :  2

o25 : 3                    # Non-rectangular node with three component shapes
 Shape_0   10    0    90   40
 Shape_1    0   40   100   10
 Shape_2   10   50    90   50
o32 : 4
 Shape_0   30  2259   963   9
 Shape_1   30  2268  1024   9
 Shape_2   30  2277  1024   9
 Shape_3   30  2286   963   9
```

circuit.route (1)

- Specifies information to perform global routing
- Example below specifies an instance with 9 metal layers

```
route 1.0
# File header with version information, etc.

Grid                 :  304  403  9
VerticalCapacity     :  0  80   0  80   0  80   0  80   0
HorizontalCapacity   :  0   0  80   0  80   0  80   0  80
MinWireWidth         :  1  1  1  1  2  2  2  4  4
MinWireSpacing       :  1  1  1  1  2  2  2  4  4
ViaSpacing           :  0  0  0  0  0  0  0  0  0
GridOrigin           :  18  18
TileSize             :  40  40
BlockagePorosity     :  0
```

Header Section

```
NumNiTerminals   :  2

  p0     4              # All the pins belonging to nodes p0/p1 are on
  p1     4                metal layer 4 for routing
```

Terminal_NI Section

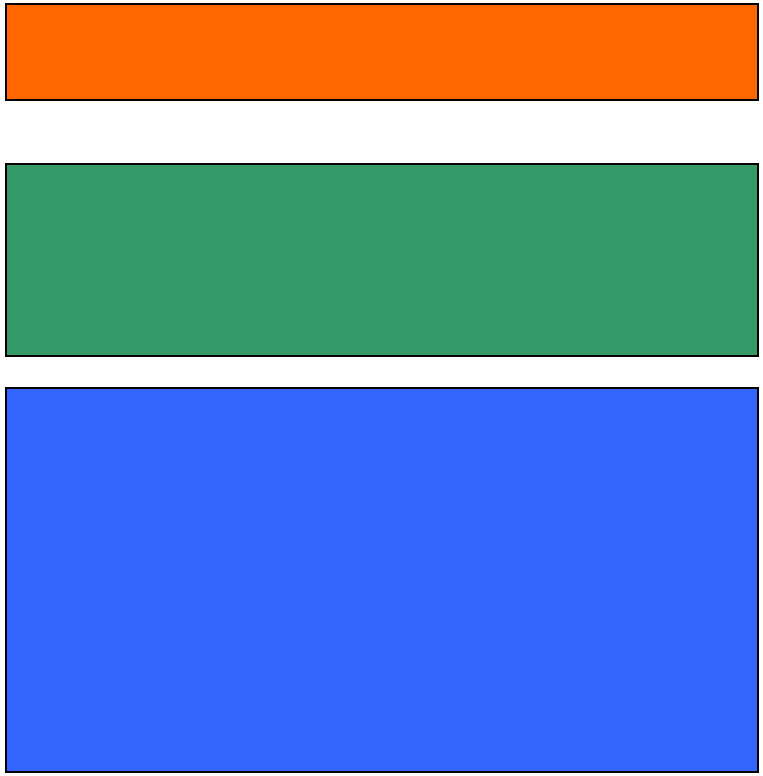
```
NumBlockageNodes   :  2

  o44   4   1 2 3 4   # o44/o2407 block 4 metal layers within all the routing
o2407   4   1 2 3 4     tiles that they overlap. These are layers 1,2,3,4.
```

Blockage Section

circuit.route (2)

Metal Stack for example in previous slide



- 9 metal layers
- M1-M4
  - 1x width and spacing
- M5-M7
  - 2x width and spacing
- M8-M9
  - 4x width and spacing

circuit.route (3)

- Similar to the ISPD 2008 routing contest format
  - http://archive.sigda.org/ispd2008/contests/ispd08rc.html

- Header Section

- Grid :	Global routing grid (num_X_grids num_Y_grids num_layers)
- VerticalCapacity :	Vertical capacity per tile edge on each layer
- HorizontalCapacity :	Horizontal capacity per tile edge on each layer (Preferred routing directions are indicated by a non-zero capacity value in that direction)
- MinWireWidth :	Minimum metal width on each layer
- MinWireSpacing :	Minimum spacing on each layer
- ViaSpacing :	Via spacing per layer
- GridOrigin :	Absolute coordinates of the origin of the grid (grid_lowerleft_X grid_lowerleft_Y)
- TileSize :	tile_width tile_height
- BlockagePorosity :	Porosity for routing blockages (Zero implies the blockage completely blocks overlapping routing tracks. Default = 0).

circuit.route (4)

- Terminal_NI section
  - NumNiTerminals : Number of terminal_NI nodes
  - For each node:

    `node_name  layer_id_for_all_node_pins`

- Blockage Section
  - NumBlockageNodes : Number of blockage nodes
  - For each blockage:

    `node_name  num_blocked_layers  list_of_blocked_layers`
  - The tiles overlapping with a blockage can be determined using placement information from the other files in the benchmark

Number of routing tracks per tile edge

How to determine the total number of routing tracks per tile edge?

The benchmark format follows the convention laid out in the ISPD 2008 routing contest. Essentially, for each tile edge, the "VerticalCapacity" or "HorizontalCapacity" values per layer give a measure of the total available space per tile edge. They are not the total number of global routing tracks per tile edge.

Hence, if the capacity for a particular layer is 80, and the minimum wire width and spacing are both 1, this corresponds to 80 / (1+1) = 40 minimum width tracks per tile edge.

For the following configuration:

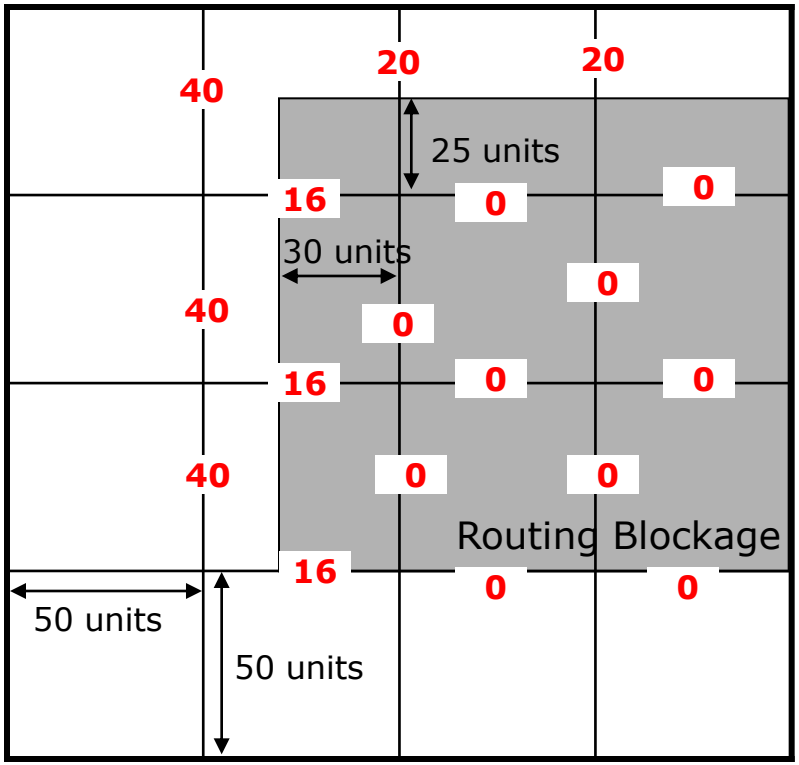
```
VerticalCapacity   : 0 80  0 80  0 80  0 80  0
HorizontalCapacity : 0  0 80  0 80  0 80  0 80
MinWireWidth       : 1  1  1  1  2  2  2  4  4
MinWireSpacing     : 1  1  1  1  2  2  2  4  4
```

Number of global routing tracks per tile edge:

```
M1:     0/(1+1) =  0
M2-M4: 80/(1+1) = 40 (for whichever capacity is not zero)
M5-M7: 80/(2+2) = 20 (for whichever capacity is not zero)
M8-M9: 80/(4+4) = 10 (for whichever capacity is not zero)
```

Example Routing Blockage Map

- The method to construct a routing blockage map for a particular layer is given below



Max H routing tracks : 40
Max V routing tracks : 40
Tile Width : 50 units
Tile Height : 50 units

Values in Red are the actual capacities in tracks of the edges

circuit.wts

- Currently unused
  - All nets have the same net-weight

Special Features for Placement and Routing

Non-rectangular Fixed Nodes (1)

- A subset of the fixed nodes in the design are not rectangular
- This affects placement density, routing capacity, etc.
- Non-rectangular nodes are represented as:
  - Enclosing rectangle – blue box in Fig. (b)
  - Set of rectangular component shapes – red hatched boxes in Fig. (b)

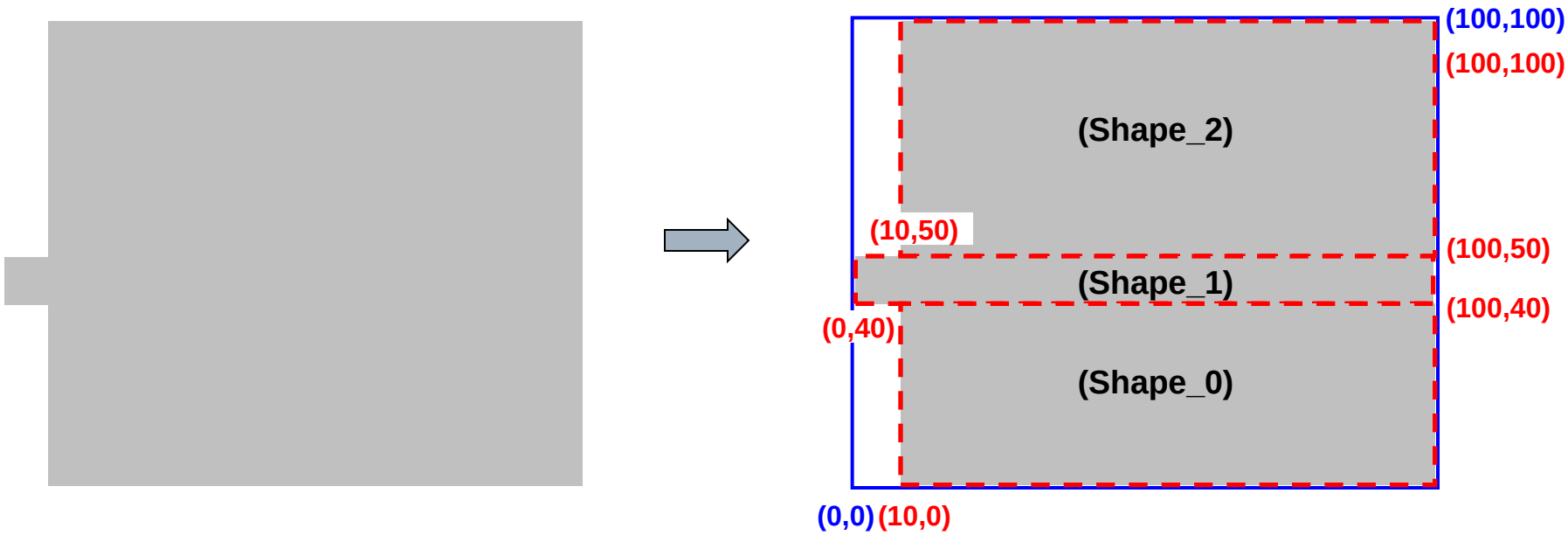
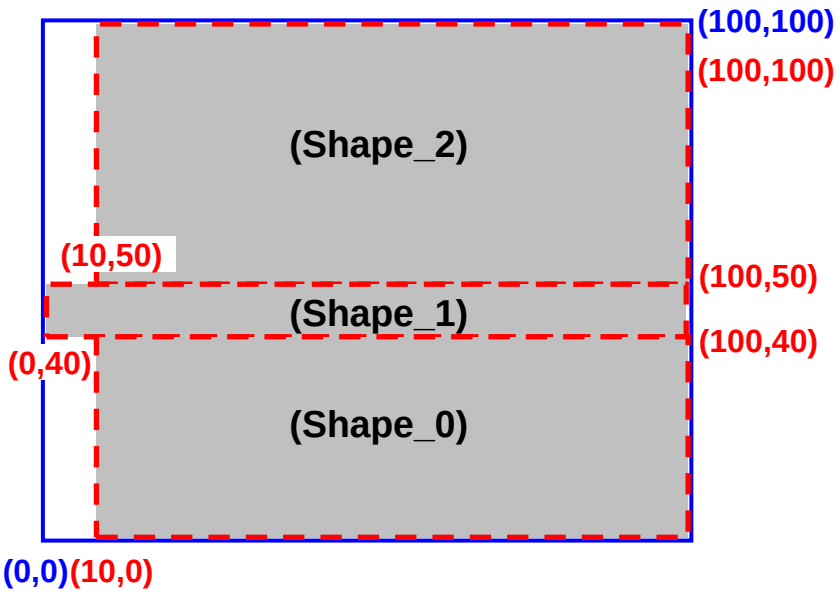


Fig (a): Non-rectangular node

Fig (b): Benchmark representation

Non-rectangular Fixed Nodes (2)



Blue: Enclosing rectangle of non-rectangular node
Red: Set of component shapes (3 in number)

```
== circuit.nodes ==
#node_name    width    height    movetype
    o25         100      100      terminal

== circuit.pl ==
#node_name   llx   lly   :   orient    movetype
    o25        0     0   :     N         /FIXED

== circuit.shapes ==
#node_name : NumComponentShapes
#Shape_id    llx    lly    width    height
o25 : 3
 Shape_0      10      0       90       40
 Shape_1       0     40      100       10
 Shape_2      10     50       90       50
```

Bookshelf Representation:

- **circuit.nodes** gives the dimensions of the **enclosing rectangle**
- **circuit.pl** gives the lower-left coordinate of the **enclosing rectangle**
- **circuit.shapes** gives the **component shape definitions** for the non-rectangular node
- **circuit.nets** gives the pin-offsets from the center of the **enclosing rectangle**

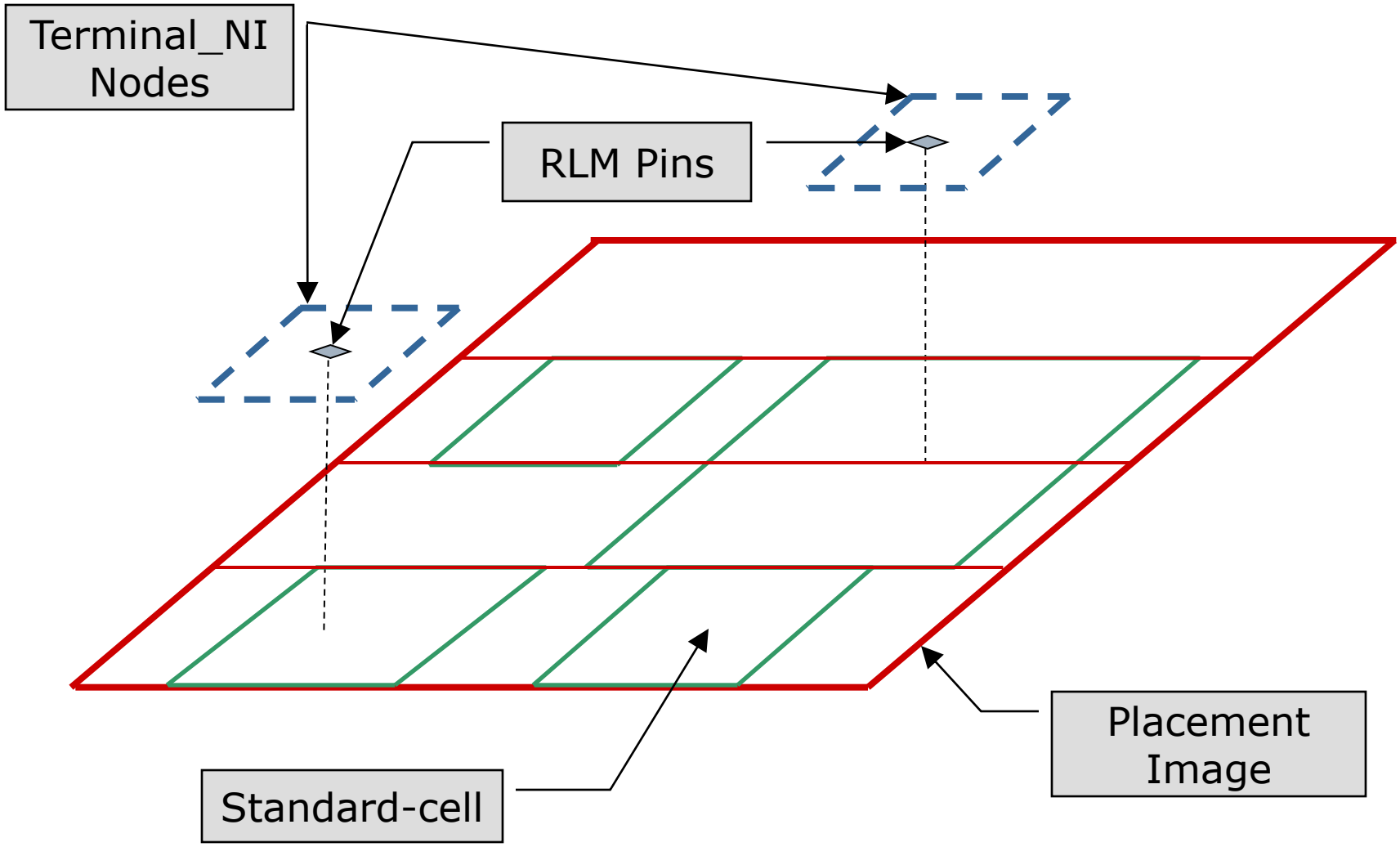
RLM Pins and Terminal_NI Nodes (1)

- RLM Pins
  - RLM pins are fixed pins that reside on a metal layer above the metal layer(s) used within a standard-cell for its pins or internal routing
  - All the RLM pins are associated with terminal_NI nodes

- **For placement**, the terminal_NI nodes are:
  - Fixed
  - Appear to reside "above" the placement image
    - In other words, standard-cells can be placed "below" the terminal_NI nodes without resulting in an overlap

- **For routing**, all pin(s) associated with the terminal_NI nodes will reside on a metal layer above M2

RLM Pins and Terminal_NI Nodes (2)



RLM Pins and Terminal_NI Nodes (3)

```
== circuit.nodes ==
#node_name    width    height     movetype
   p25          1        1       terminal_NI

== circuit.pl ==
#node_name    llx    lly    :  orientation    movetype
   p25         30     30    :       N          /FIXED_NI

== circuit.route ==
NumNiTerminals   :  Number_of_Terminal_NI_Nodes

#List of nodes with metal layer for ALL the pins on the node
#node_name    Layer_ID
   p25           3            # All the pins on node p25 reside on M3
```

Bookshelf Representation :

Placement:
- Movetype **terminal_NI** in **circuit.nodes** file (overlap is allowed with this node)
- Represented as **FIXED_NI** in **circuit.pl** file

Routing:
- Terminal_NI section in **circuit.route** file gives the metal layer for all the pins on such nodes
- The pins for any node not given in this section of circuit.route will be on layer M1

Utility Scripts

Script: dac2012_check_legality

- Perl script to check the legality of the placement solution
- Usage:
  dac2012_check_legality <circuit.aux> <solution.pl>
- This script checks the following conditions:
  - ERROR_TYPE 0: did a terminal or terminal_NI node move?
  - ERROR_TYPE 1: is a movable node placed outside the placement area?
  - ERROR_TYPE 2: is a movable node aligned to the circuit rows?
  - ERROR_TYPE 3: is a movable node placed on a multiple of Sitespacing?
  - ERROR_TYPE 4: are there any overlaps among the nodes (movable and/or fixed)?
- Can serve as a guideline to parse the benchmark files

Script: dac2012_get_hpwl

- Perl script to get the Half-Perimeter Wire Length (HPWL) of the placement solution
- Usage:
  dac2012_get_hpwl <circuit.aux> <solution.pl>
- Can serve as a guideline to parse the circuit.nets file and determine pin positions, etc.