ALX Symmetry Functional Specification

[0. Revisions 2](#_Toc353546954)

[1. Introduction 3](#_Toc353546955)

[1.1 Functional subsystem identifier 3](#_Toc353546956)

[1.2 Definition of terms and acronyms 3](#_Toc353546957)

[1.3 Problem statement 4](#_Toc353546958)

[1.4 Scope 4](#_Toc353546959)

[1.5 References 4](#_Toc353546960)

[2. ALX Symmetry 5](#_Toc353546961)

[2.1 Device Symmetry 5](#_Toc353546962)

[2.1.1 Device Symmetry Vertical 5](#_Toc353546963)

[2.1.2 Device Symmetry Horizontal 6](#_Toc353546964)

[2.1.3 Device Symmetry Composite 7](#_Toc353546965)

[2.1.4 Multiple Groups 8](#_Toc353546966)

[2.1.5 Device Symmetry Soft 9](#_Toc353546967)

[2.1.6 Device Symmetry Force 10](#_Toc353546968)

[2.1.7 Device Symmetry bBox 10](#_Toc353546969)

[2.1.8 Device Symmetry for Rotated Devices 11](#_Toc353546970)

[2.1.9 N or P Type Device Group 11](#_Toc353546971)

[2.2 Route Symmetry 12](#_Toc353546972)

[2.2.1 Route Symmetry Hard 12](#_Toc353546973)

[2.2.2 Route Symmetry Soft 13](#_Toc353546974)

[2.3 Route Alignment 15](#_Toc353546975)

[2.3.1 Route Alignment Hard 15](#_Toc353546976)

[2.3.1 Route Alignment Soft 17](#_Toc353546977)

[2.4 Abstract Device Symmetry 18](#_Toc353546978)

[2.4.1 Abstract Symmetry Hard 18](#_Toc353546979)

[2.4.1 Abstract Symmetry Composite Groups 20](#_Toc353546980)

[3. Command description 21](#_Toc353546981)

[3.1 Device Symmetry 21](#_Toc353546982)

[3.2 Route Symmetry 23](#_Toc353546983)

[3.3 Route Alignment 26](#_Toc353546984)

[3.4 Abstract Device Symmetry 28](#_Toc353546985)

[4. Future enhancements 29](#_Toc353546986)

[5. Questions and Answers 29](#_Toc353546987)

[6. Dependencies, assumptions, risks 29](#_Toc353546988)

# Revisions

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Date | Release | Comment | SCNs | Author |
| 4/9/2013 |  | Intial draft; Only device symmetry |  | Mohit |
| 4/12/2013 |  | First draft |  | Mohit |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

**Delete Me:** How to annotate functional specification document for the given release:

For each change:

* Updates in revision history table:
  + If the change will be implemented for the given release, turn row background to green.
  + If the change will not be implemented for the given release, turn row background to red.
* Update the Fspec during development according to the below color coding scheme
  + Add all substantive changes using green text if the change will be implemented for the given release (make formatting changes without tracking them)
  + Add all substantive changes using red text if the change will not be implemented for the given release (make formatting changes without tracking them)
  + *Note: Text in black means the change already is in the system*

If the fspec is completely new for E-2010.09 release use **black** text. Green/red text should be used to indicate incremental changes in the specs from release to release.

# Introduction

## Functional subsystem identifier

This document describes auto constraint generation for preserving symmetry in the ALX output. This is done for:

* NP devices
* Abstract devices
* Wire Routes

## Definition of terms and acronyms

|  |  |
| --- | --- |
| Titan | Mixed signal platform |
| ALX | Analog Layout Accelerator |
| LP | Linear Programming |
| molCell | Intermediate ALX physical cell view having all shapes from the flattened input |
| Bonds | ALX construct to create relationship between intermediate physical shapes |
| cellId | User level handle for a design opened in Titan |
| objectId | User level handle for a objects in Titan database |
| bBox | Bounding Box |
| Point | Pair of Coordinates |
| OD | Diffusion Layer |
| UU | User Units |
|  |  |

## Problem statement

To maintain beauty of the ALX output additional bonds are required which capture symmetry of shapes in a design. This will be done for NP devices, abstract devices and wire routes.

## Scope

Functionality expectations, user level commands, output, flow integration steps will be covered. Target users are Titan PE, AE, Documentation teams.

## References

1. ALX User Guide
2. Titan User Guide

# ALX Symmetry

Symmetry extraction will be done on the ALX molCell. User level commands provide an option to generate basic bonds or high level constraints. The bonds or constraints are dumped to an ascii file which can be specified by the user. The bonds generated are required to be fed to the ALX LP engine. User can review/edit constraints.

## Device Symmetry

Symmetry constraints are generated for NP devices.

### Device Symmetry Vertical

* For devices stacked vertically.
* OD bBox is used for symmetry computation.
* Devices are symmetrical if the median axis along the direction of gate is same.
* OD width of devices need not be same.
* Different rows in the stack can be of N or P type.
* Hard symmetry constraints are generated for such groups.

Figure 1: Vertical symmetrical group

### Device Symmetry Horizontal

* OD bBox is used for symmetry computation.
* OD bBoxes (and hence the device) are reflection of each other about the combined median axis along the direction of gate.
* A single device will be self-symmetric about its media axis along direction of gate.
* Two devices will for a symmetric pair.
* Only same type of devices can be horizontally symmetrical. Hence a horizontal symmetric group will have all N devices or all P devices.
* Hard symmetry constraints are generated for such groups.

Figure 2: Horizontal symmetrical group.

### Device Symmetry Composite

* The basic symmetrical pairs can combine to create a larger symmetric group.
* Groups are created to contain maximum number of devices.

Figure 3: Example of composite group.

Figure 4: Example of composite group.

Figure 5: Example of a mixed composite group.

### Multiple Groups

* At times devices can belong to multiple symmetry groups.
* For such devices, multiple constraints will be generated.

Figure 6: Example of device in multiple groups.

### Device Symmetry Soft

* At times it is required to maintain symmetry for devices which not exactly symmetrical.
* User can specify the “tolerance” for which the devices will be considered symmetrical.
* Soft symmetry constraints generated for such groups.
* For soft symmetry constraints, user specified “force” value is used.
* “Soft” constraints will not cause infeasibilties/conflicts during migration and is preferred when a constraint is not a must-have requirement, a.k.a, soft bond.

#### Device Symmetry Soft Vertical

* Tolerance here is the separation between the device median axes.
* Please specify units for tol

tol

Figure 7: Vertical Soft Symmetry Group.

#### Device Symmetry Soft Horizontal

* Tolerance here is the difference of device median axis from the combined group media axis.

tol

Figure 8: Horizontal Soft Symmetry Group.

#### Device Symmetry Soft Hard Constraints

* A special user specified force value of “-1” generates hard constraints also for devices symmetrical under tolerance.

### Device Symmetry Force

* User can specify the force value to be used for soft symmetry constraints.
* Force value of 1000 will make the constraint stronger, that is it will be given higher priority by the ALX LP solver.
* Maybe we should list the default force value(s) for other bonds so that the user knows what value to use for higher/lower priority

### Device Symmetry bBox

* By default symmetry extraction is done for the complete cell.
* If required, user can specify the bBox for which symmetry extraction is done.

### Device Symmetry for Rotated Devices

* Symmetry extraction done independently for vertical and horizontal gate directions.
* Such groups are not mixed, or form composite groups.
* Constraints generated capture the different directions.

### N or P Type Device Group

* Devices having NDIFF for diffusion layer are clubbed into N type groups.
* Devices having PDIFF for diffusion layer are clubbed into P type groups.
* What if there is only OD in the design (no specific ndiff/pdiff? I presume that we internally convert everything into ndiff/pdiff by looking at PP/NP layers to figure out which type of OD it is. Maybe we can mention this here.
* The “-note” field in the generated constraints also capture N or P type group.

## Route Symmetry

Constraints generated for paths and wires symmetrical about devices symmetry axis.

Are device groups considered for route symmetry or only single device fingers, or device OD only?

### Route Symmetry Hard

* Symmetry extraction done for path, wire, pathSeg, rectangle and polygon objects.
* Symmetry identified w.r.t. to route shape median axis in the direction of device gate.
* Constraints only capture the route shapes. Hence exclusive constraints for device and routes about the same symmetry axis.
* Done only for shapes on metal layers.
* The symmetrical pair shapes are on same metal layer.
* Metal layers are picked from ALX tag layers.

Figure 9: Route Symmetry Group.

### Route Symmetry Soft

* Constraints for wire route not perfectly symmetrical.
* User can specify “tolerance” for which shapes will be considered symmetrical.
* Soft symmetry constraints generated for such groups.
* For soft symmetry constraints, user specified “force” value used.

#### Route Symmetry Soft Same Path Width

* Path width of routes is same but median axis is not symmetrical.

tol

Figure 10: Route Symmetry Soft Group.

#### Route Symmetry Soft Different Path Width

* Path width of routes is different.

**pw2**

w2w2

**pw1**

w2w2

**tol = pw2 – pw1**

w2w2

Figure 11: Route Symmetry Soft Group.

## Route Alignment

Alignment constraints generated for symmetrical shapes, when top and bottom coordinates are aligned.

### Route Alignment Hard

* For shapes perfectly aligned.
* Constraint can be generated for either top or bottom edge also.
* Hard constraints generated.

Figure 12: Route Alignment Group.

Figure 12: Route Alignment Group with Top edge aligned.

### Route Alignment Soft

* Constraints for shapes not perfectly aligned.
* User can specify “tolerance” for which shapes will be considered aligned.
* Soft alignment constraints generated for such groups.
* For soft alignment constraints, user specified “force” value used.

**tol**

Figure 13: Route Alignment Soft Group

## Abstract Device Symmetry

Symmetry generation for abstract blocks.

### Abstract Symmetry Hard

* Hard symmetry constraints symmetrical abstract blocks.
* Blocks are of same dimensions.
  + This is for source block dimensions, right? I think the molcell has source block dimensions. Lets state that here.

Figure 14: Abstract symmetry y-axis

Figure 15: Abstract symmetry x-axis

### Abstract Symmetry Composite Groups

* Single grouping for blocks of different dimension but symmetrical about common axis.

Figure 16: Abstract Symmetry Composite Group.

# Command description

## Device Symmetry

Synopsis

alx gen devSymBonds <molCellId> [alx gen devSymBonds options]

 alx gen devSymBonds options:

    -symBondFileName <string> (default: airDevSymBonds.tcl)

    -force <float> (default: 500)

    -tolerance <float> (default: 0)

-keepAlign

    -bBox        <ll> <ur>

        ll: <point>

        ur: <point>

Only device symmetry related arguments listed above.

Description

This is a user level command to get device symmetry constraints. Constraints are dumped to the file specified by user. Currently this is a hidden command.

Arguments

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Type | Optional | Default | Description |
| molCellId | Int | No | NA | molCell cellId |
| symBondFileName | String | Yes | airDevSymBonds.tcl | Output file name |
| tolerance | Float | Yes | 0 | Distance in user units (UU). This is used to create symmetry constraints for devices which are not exactly symmetrical |
| force | Float | Yes | 500 | Force value to be used for soft bonds. For the special value of “-1”, all bonds are created hard |
| keepAlign | Bool | Yes | 0 | ?? |
| bBox | {point point} | Yes | None | When specified, symmetry extraction done for the specified bBox |

Sample Output

alx gen devSymBonds 3 –symBondFileName sample.tcl –tolerance 0.1

cat sample.tcl

# <libName/cellName/ViewName>

# Device Symmetry Bonds

alx add constraint symmetry 3 { 512 513 514 515 } -dir x -note autoSymDeviceVM\_141

alx add constraint symmetry 3 { 515 516 } -dir x -soft -force 500 -note autoSymDeviceHP\_150.845-151.480

Aren’t we changing above to use native bonds instead of custom bonds? I guess you will update this section at that time?

Above output shows 2 symmetry constraints:

* First one is a hard constraint, capturing 4 objects.
* dir “x” implies direction of gate is in vertical direction.
* Second is a soft symmetry bond.
* note field has additional constraint information. It following syntax:

autoSymDevice<VorH><MorPorN>\_<axis>

* V indicates vertical group, this can also be
* H for horizontal group.
* M denotes mixed group, this can also be
* N for n device group.
* P for p device group.
* axis is the physical coordinate of median axis in UU. For soft symmetry bonds this can be a range.

## Route Symmetry

Synopsis

alx gen devSymBonds <molCellId> [alx gen devSymBonds options]

 alx gen devSymBonds options:

    -symBondFileName <string> (default: airDevSymBonds.tcl)

    -bBox        <ll> <ur>

        ll: <point>

        ur: <point>

-noRoute

-routeSymmetryTolerance <float> (default: 0)

-routeForce <float> (default: 300)

-routeWidthMatchTolerance <float> (default: 0)

-routeLayers <list>...

-keepRouteAlign

Only route symmetry related arguments listed above.

Description

This is a user level command to get route symmetry constraints. Constraints are dumped to the file specified by user. Currently this is a hidden command.

Arguments

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Type | Optional | Default | Description |
| molCellId | Int | No | NA | molCell cellId |
| symBondFileName | String | Yes | airDevSymBonds.tcl | Output file name |
| bBox | {point point} | Yes | None | When specified, symmetry extraction done for the specified bBox |
| noRoute | Bool | Yes | 0 | When set route symmetry constraints not generated |
| routeSymmetryTolerance | Float | Yes | 0 | Distance in user units (UU). This is used to create symmetry constraints for shapes which are not exactly symmetrical |
| routeWidthMatchTolerance | Float | Yes | 0 | Distance in user units (UU). This is used to create symmetry constraints for shapes which are not of same width |
| routeSymmetryForce | Float | Yes | 300 | Force value to be used for soft bonds. For the special value of “-1”, all bonds are created hard |
| routeLayers | List | Yes | None | List of layers for which route symmetry constraints are generated |
| routeKeepAlign | Bool | Yes | 0 | ?? |

Sample Output

alx gen devSymBonds 3 –symBondFileName sample.tcl –tolerance 0.1 –routeSymmetryTolerance 0.1

cat sample.tcl

# <libName/cellName/ViewName>

# Route Symmetry Bonds

alx add constraint symmetry 3 { 1050 1081 } -dir x -keep\_aligned

-note autoSymWireV\_M2\_8.250

alx add constraint symmetry 3 { 744 756 749 757 753 753 } -dir x

-keep\_aligned –soft –force 300 -note autoSymWireV\_M1\_8.252-8.262

Above output shows 2 route symmetry constraints:

* First one is a hard constraint capturing 2 objects.
* dir “x” implies direction of gate is in vertical direction.
* Second is a soft symmetry bond.
* note field has additional constraint information. It following syntax:

autoSymWire<VorH>\_<metalLayerName>\_<axis>

* V indicates vertical group, this can also be
* H for horizontal group.
* metalLayerName is the name of metal layer as described in technology.
* axis is the physical coordinate of median axis in UU. For soft symmetry bonds this can be a range.

## Route Alignment

Synopsis

alx gen devSymBonds <molCellId> [alx gen devSymBonds options]

 alx gen devSymBonds options:

-noRouteAlign

-routeAlignForce <float> (default: 1000)

-routeAlignTolerance <float> (default: 0)

-routeAlignKeepAlign

Only route alignment related arguments listed above.

Description

This is a user level command to get route alignment constraints. Constraints are dumped to the file specified by user. Currently this is a hidden command.

Arguments

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Type | Optional | Default | Description |
| molCellId | Int | No | NA | molCell cellId |
| symBondFileName | String | Yes | airDevSymBonds.tcl | Output file name |
| bBox | {point point} | Yes | None | When specified, symmetry extraction done for the specified bBox |
| noRouteAlign | Bool | Yes | 0 | When set route alignment constraints not generated |
| routeAlignTolerance | Float | Yes | 0 | Distance in user units (UU). This is used to create alignment constraints for shapes which are not exactly aligned |
| routeAlignmentForce | Float | Yes | 300 | Force value to be used for soft bonds. For the special value of “-1”, all bonds are created hard |
| routeAlignKeepAlign | Bool | Yes | 0 | ?? |

Sample Output

alx gen devSymBonds 3 –symBondFileName sample.tcl –tolerance 0.1 –routeSymmetryTolerance 0.1

cat sample.tcl

# <libName/cellName/ViewName>

alx add constraint alignment 8 { 635 634 } -dir x -keep\_aligned

-line both -note autoAlignWireV\_M1\_0.690\_L:0.375\_R:0.785

alx add constraint alignment 8 { 637 635 } -dir x -keep\_aligned

-line left -note autoAlignWireV\_M1\_0.690\_L:0.370

Above output shows 2 route alignment constraints:

* First one is a hard constraint, capturing 2 objects
* dir “x” implies direction of gate is in vertical direction.
* Second one has only bottom edge aligned.
* -line <type> tells which edges are aligned.
* both when both edges aligned.
* left when bottom edge only aligned for shapes symmetrical about y axis.
* right when top edge only aligned for shapes symmetrical about y axis.
* note field has additional constraint information. It following syntax:

autoAlignWire<VorH><metalLayerName>\_<axis>\_L:<lEdge>\_R:<rEdge>

* V indicates vertical group, this can also be
* H for horizontal group.
* metalLayerName is the name of metal layer as described in technology.
* axis is the physical coordinate of median axis in UU. For soft symmetry bonds this can be a range.
* lEdge is the left edge coordinate.
* rEdge is the right edge coordinate.

## Abstract Device Symmetry

Synopsis

alx gen absSymBonds <molCellId> [alx gen absSymBonds options]

 alx gen absSymBonds options:

    -symBondFileName <string> (default: airAbsSymBonds.tcl)

Description

This is a user level command to get device symmetry constraints. Constraints are dumped to the file specified by user. Currently this is a hidden command.

Arguments

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Type | Optional | Default | Description |
| molCellId | Int | No | NA | molCell cellId |
| symBondFileName | String | Yes | airAbsSymBonds.tcl | Output file name |

Sample Output

alx gen absSymBonds 3 –symBondFileName sample.tcl

cat sample.tcl

# <libName/cellName/ViewName>

# Device Symmetry Bonds

alx add constraint symmetry 3 { 41194 41191 41192 41204 41208 41210 }

-dir y -keep\_aligned -note autoSymAbstractH

alx add constraint symmetry 3 { 41169 41170 } -dir x -keep\_aligned

–note autoSymAbstractV

Above output shows 2 symmetry constraints

* First one is a group of 6 objects symmetrical about y axis.
* Second group has 2 objects symmetrical about x axis.
* note field has additional constraint information. It following syntax:

autoSymAbstract<VorH>

* V indicates vertical group, this can also be
* H for horizontal group

# Future enhancements

* Sizing information from output is to be corroborated before constraint generation.
* Basic bond generation inplace of high level user constraints.
* bBox argument to be removed. For all shapes, parent cell chain information to be used.
* Use of device bBox edge instead of median axis for symmetry computation? Ex for wire routes, constraints can be generated only for inside edges.
* Usage of soft symmetry, soft alignment constraints to maintain relative device positions.
* Support for “Instance” symmetry.
* Currently a hidden command.
* Additional alignment commands?
* Symmetry extraction to be moved to input instead of molCell?
* Auto feeding of bonds?
* Tolerance support for Abstract devices.
  + We should probably increase the priority of this one. Blocks are often off by a grid unit or two.
  + Btw, we have not tested starc with abstract dev symmetry feature. Currently we are using custom bonds to achieve the abs symmetry. This should be part of the test plan if time permits.
* Resizing information usage for abstract devices.

# Questions and Answers

This section contains pertinent questions and answers encountered during the development of the specifications or during tool implementation.

# Dependencies, assumptions, risks

* Uses ALX derived layer dev\_mos.
* The constituent shapes of dev\_mos are at times not symmetrical themselves.
* For device symmetry, constraints are generated only for OD shapes. Fingers are attached to diffusion by other bonds. Their symmetry hence is dependent on these other bonds.
* N or P type device distinction based on NDIFF, PDIFF layer usage in design.
* For abstract device symmetry, instance master information is not used, hence instances of same size but different masters not distinguished.
* For abstract device symmetry, instance transformation information is not used. Hence constraints generated for non reflective blocks also.