2013.08.14

Updates:

1. .nets file can be generated.
   1. Usage
      1. addComponentCluster(…, net)
      2. addComponentCluster(…, netList)
   2. Functions
      1. Integration with Class FPNets
      2. Add, Removal from FPNets
   3. Limitation
      1. Only tested with basic testing setup on geogLayout
      2. netLength still needed to be validated
2. Next Step:
   1. Eliminate the limitations (validate with all other kinds of layouts)
   2. Extend the testcase to original layouts
   3. Compare the generated ParquetFP from .blocks/.nets with the ArchFP for optimization
3. Compare first with ParquetFP.
4. Generate an automated visualization with ArchFP.
   1. Draw nets
5. Test: 64 Processors firsts
   1. Then with cache in each tile
6. Randomization is bad for optimizating ordered-layout

Add: Parameters to add Cluster (fully connected, mash connected , ring network)

Subcomponent name: allow the router to be connected instead of the whole component.

2013.07.31

Updates:

1. Reviewed what was done and ensure what is on the to do list
2. Re-structure the FPNet definition with FPObject
   1. It makes more sense for user to define which net the element belongs to at the definition (ex. when we use AddComponentCluster)
   2. If the element needs to be in several nets, use an array of nets at the definition.
   3. I will ensure both are functional (with or without net definition)
   4. No need to be able to add wire specifically, which kind of require violating the access hierarchy.
3. Short-term to do list:
   1. Enable .nets file generation from the FPNet
   2. Compare the generated ParquetFP from .blocks/.nets with the ArchFP for optimization.

FP Project Meeting 2013.04.29

Completed tasks:

1. Basic structure for wire connection in ArchFP is done, explain and verify with Prof. Meyer.
   1. connWire FPObject::(c2, direction)
      1. addWire [to \*\*connItems]
      2. removeWire

FP Project Meeting 2013.04.22

Completed tasks:

1. Implemented a function in ArchFP that converts the examples into ParquetFP Flp (which can be done both before and after the layout).
2. Tested that the converted .blocks works with the shell script.

Questions:

1. How to specify .nets from ArchFP examples? The wire length calculation in ArchFP basically connects everything together after legalization, so for now we just bi-directionally connect all devices to each other?
   1. Specified by user at ArchFP. Design the wire connection ability in ArchFP.
2. So I will write something to automate the comparison process between ArchFP examples’ area and ParquetFP’s?
   1. Yes. Automate the comparison process between ArchFP and ParquetFP. The journal will be based on how ArchFP is better than ParquetFP in terms of structured designs.
3. Optimizing legalization with ParquetFP?
   1. No

To do:

1. Convert the .pl layout into the ArchFP layout (with coordinates already specified)
2. ~~Nets Class~~

FP Project Meeting 2013.03.25

Completed tasks:

1. Fixed bug on detectOverlap
2. Code cleanup and added test cases

To Do:

1. Optimize legalization process
   1. Experiment to prove the legalization is not optimal
      1. What to do? How?
      2. Comparing the area of an ArchFP floorplan with what ParquetFP produces. Use ArchFP (original one) and tweak it with AR constraints.
      3. **Learn ParquetFP, know what to be optimized.**
      4. **Write a function in ArchFP that converts the Hotspot flp into ParquetFP Flp.**
      5. **ParquetFP needs multiple files**
2. Tight integration between ParquetFP and ArchFP
   1. Q: Xinchi provided his code in freespace.main, no ParquetFP file is included. Shall we directly integrate it in the floorplan.cc?
   2. **A: Implement as a function in a basic container, internal member function**
3. Framework to call external floorplaner (ex. ParquetFP)
   1. Block Packer (Optimal way to pack rectangles in a bigger one)
   2. ParquetFP
4. Way to implement wires and measuring wire length?
   1. Inter-connects components
   2. Center Coordinate
   3. HPW (Half Perimeter Wire Length)
      1. Rectangle

1) a way to specify that things are connected,

2) a function to call to calculate the total wire length, and

3) a way to print wire length statistics in the output.

On this last point, the data should probably be "commented out" so that it doesn't interfere with HotSpot.  HotSpot doesn't make use of wiring information.

1. Legalization based on wirelength (Internal justify the decision)
   1. Q: It is done after legalization, and as long as the floorplan is legalized, the estimated wirelength is valid. What else?
   2. **Pay attention to Change in wirelength/Change in area**

FP Project Meeting 2013.02.28

Completed tasks:

1. Added TopBottomInversion
2. Added Overlap detection
3. Implemented Sort by decreasing area with legalization
4. Added the flag for legalization and all of its features

TO DO:

1. Expand the boundary based on the overlapping area.
2. Come out with the best expansion direction (TOP or RIGHT) and try it out in order.
3. Increase the area of the bounding box.
4. Always valid output.

ddCompleted Tasks:

1. Handled bagLayout (LeftRight/TopBottom)
2. Handled gridLayout (Mutliple elements)
3. ~~Changed the TopBottom order to BottomTop~~ (removed)
4. Added out-of-bound detection based on remaining area
   1. Different cases
      1. AR constraint (isGoodAR = false)
      2. Simply not enough space due to outer deadspace
      3. Note: Order of the components matters
   2. Question: Should we automatically swap the order in which the components are added or prompt user to do so in some out-of-bound cases?
   3. In other words, should we aim to build a more robust automating or user-customizable system?
5. Added optional overlap detection
   1. exception throwing
   2. not used at the moment

Questions:

1. How to properly design overall and suitable test cases? (+corner cases)
2. Does it matter if a few values from the original design are different?
   1. And it only happens when we optimize the compilation
   2. No difference when we compile normally

Next Week:

* Test basic 180 and Mirror cases.
* Deliver a working solution with non-recursive test cases for LeftRight/TopBottom/180/Mirror
* Deliver a working solution with non-recursive test cases for LeftRightMirror, TopBottomMirror and Center.

Later objective: Deliver an algorithm for recursive test cases