Methodology for Repeater Insertion Management in the RTL, Layout, Floorplan and Fullchip Timing Databases of the Itanium™ Microprocessor

Rory McInerney
Intel Corp.
SC12-408
408-765-6341
rory.mcinerney@intel.com

Kurt Leeper
Intel Corp.
SC12-405
408-765-8512
kurt.leeper@intel.com

Troy Hill
Intel Corp.
RA2-350
503-613-3389
troy.hille@intel.com

Heming Chan
Intel Corp.
AN1
512-314-0058
heming.chan@intel.com

Bulent Basaran
Intel Corp.
SC12-601
408-653-7245
bulent.basaran@intel.com

Lance McQuiddy
Intel Corp.
SC12-405
408-765-6948
lance.mcquiddy@intel.com

ABSTRACT

In this paper, we describe a methodology for inserting repeaters into the RTL, Layout, Floorplan and Fullchip timing databases of the Itanium™ processor.

Keywords

Microprocessors, floorplan, repeaters, routing, timing, estimation, RC delay.

1. INTRODUCTION

Most microprocessors designed at Intel require repeating signals at the Fullchip level to some degree. On the Itanium™ processor, 85% of the Fullchip level routes require one or more repeaters. Previous papers on repeater insertion primarily concerned themselves with the proper placement of repeaters on a single net using various algorithms that optimized such electrical constraints as RC delay, power and placement in a repeater station among others.

This paper will discuss the Itanium™ processor repeater estimation techniques and will also address new aspects of repeater insertion. It will develop a repeater solution for a net that is electrically sound and will then deal with the problem of implementing thousands of such solutions in the microprocessor's floorplan, RTL, schematic, layout and timing databases. As will be shown the sheer magnitude of the number of nets requiring repeaters requires a dramatic modification to some of the design flows.

ISPD 2000, San Diego, CA.
ACM 1-58113-191-7/00/0004.

There were six challenges that need to be addressed in order to be able to successfully manage the repeater solution of the Itanium™ processor.

- Early in the design phase the designer needs to have a good repeater estimation capability. Repeaters are needed to improve the speed of a net and to improve signal slopes. The estimation tool on the Itanium™ processor was able to deal with both of these constraints in addition to the constraint of where the repeater stations are placed. It also helped the designer specify where the best location of a repeater station would be for that particular signal.
- Traditional works on repeater insertion deal with cases where the signal is not functionally changed by the repeater. On the Itanium™ processor many of the nets require solutions that involve clocked and enabled elements such as latches and flip-flops. A methodology that incorporates these types of repeaters is needed.
- The actual design environment into which the repeaters are being inserted is not stable. Typically you can generate reasonably accurate specifications to an LBF (Layout Bridging Fub: an Intel® acronym for a layout building block that corresponds to a functional block in RTL) for implementation. While RC specifications can also be generated, the repeater implementation to meet those specifications can have multiple solutions depending on actual route, metal layers used, additional electrical constraints such as robustness and others that are not always well understood at the beginning of the project. Therefore there is a need to have a repeater implementation methodology that is highly flexible and isn't frozen until the end of the project.
- The design flow for repeaters is fundamentally backwards. Typically a design starts with an RTL functional description and then it migrates to schematics, layout and finally into the floorplan. In repeater design, the flow is reverse. The repeater solution is dependent on the floorplan being routed and the fixes are then back propagated into the layout, schematic and

RTL databases. Given the inter-connected nature of design tools, this back-propagation is not trivial.

- The analysis loop for repeater analysis is long. The analysis requires a full set of fullchip netcells, which is the product of a Physical Integration (explained further in the paper). A netcell is the electrical equivalent of a fullchip route. It is a distributed network of resistances and capacitance. Typically once these netcells are generated they are fed back into the timing model. However when repeaters are introduced this critical feedback loop can be delayed. The repeater methodology needs to have a good estimation approach and also needs to feed real data into the timing model on a regular basis.
- The repeater solution freezes late in the design stage. Therefore the repeater stations themselves tend to be the latest LBFs to be finished in the design. The schematic and physical implementation therefore needs to be fast and should not cause any timing surprises once completed.

The Itanium™ processor team developed a comprehensive repeater methodology that addresses all of the challenges listed. This paper will discuss each aspect of the solution, will present results on how successful the methodology was implemented and finally will summarize the key findings that can be applied to other projects.

2. Outline of paper

Five aspects of the Itanium™ processor repeater methodology will be discussed.

- A description of the repeater estimation tools will be given. These tools were used from the early stages of the design to help plan the routing and the placement of repeater stations.
- A description of the repeater station design will be reviewed in addition to how the repeater stations are incorporated into the floorplan and how the repeated nets are implemented in the floorplan.
- A description of how the repeaters are incorporated into the RTL model.
- A description of the physical layout implementation of the repeater stations and the schematic generation
- A description of the floorplan-timing interaction that surrounds repeater insertion.

3. Repeater Estimation

There are two types of estimation used on the Itanium™ processor, unconstrained and constrained based. In the early phase of the design the floorplan is very immature and has no repeater stations. A tool was developed that places the repeaters in their ideal location on a net based on the input slope on the receiver. This is the first pass repeater assignment and with appropriate guard-banding it will suffice until a constrained based solution can be developed.

As the floorplan matures and repeater stations are inserted it is necessary to constrain the repeater solutions to actual repeater stations. A software tool called MARS (Merced™ Automatic Repeater Solution) was developed that used the location of the repeater stations to generate a solution that met the signal timing and quality requirements with the minimum number of repeaters. Keeping the number of repeaters low is good as it minimizes the number of times the route has to drop down through all the different metal layers thus adding via resistance and impacting the routability of the lower level metals.

4. Floorplan and Repeater station design

The basis of the repeater station (RPS) is the repeater standard cell library and the standard RPS template design. The repeater standard cells all fit into a standard layout template. Regular structures and arrays set groundwork for automation. In repeaters, automation can be taken advantage of and increase productivity.

Repeater standard cells must be designed robust enough to handle electrical and physical design constraints. Regular standard cell libraries deal with leaf cell design problem. Repeater Standard cells need to be designed for Full Chip design problems such as noise and inductance. The cells are pre-characterized for noise concerns and the P/N ratio is skewed to deliver an equal rise and fall time.

The library supports inverters and buffers in both static and domino designs. It also has Flip-Flops, Latches and some other specialized cells. There is a standard selection of cell strengths that are required for driving Full Chip nets. The ability to create custom designed cells is easy; the only requirements being that they conform to the standard repeater cell template.

All repeater standard cells follow the standard repeater cell template. Standard cells are placed in the RPSs by flip arraying, or mirroring each cell with respect to the cell next to it.

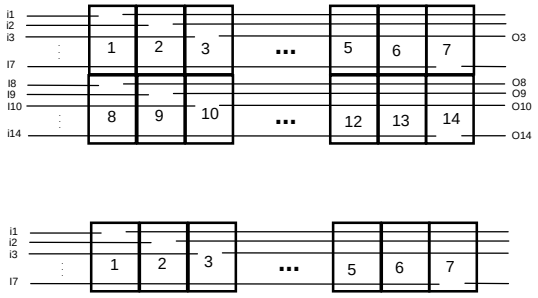

Figure 1: Repeater station template

RPSs are mostly abutted to Unit boundaries. Because of the M2 and M3 signals that come out of Units, the Repeater standard cells need to have a high level of porosity for feed-throughs. This is done by predefining the I/O, Power, and Ground, Enable and Clock tracks for each repeater standard cell in the repeater standard cell template. The RPSs then have regular patterns of feed through such that unit physical designer can create M2 and M3 pins on their units and not collide with internal repeater routing. The level of porosity is 70%, or 21/30, Metal three tracks, and 50%, or 9/18, Metal two tracks. This high level of porosity allows the repeaters to be inserted anywhere in the floorplan and not constrict the M2 or M3 routing in that area. This template works for both Horizontal and Vertical RPSs.

The need for the repeater standard cells to fit into a standard template is evident when the design of the repeater station itself is reviewed. The fundamental concept behind the RPS station

design is that ANY signal in Metal4, Metal 5 that crosses over the station can be repeated. This means there needs to be enough repeater standard cells in the station to accommodate repeaters on every wire.

In the example of a Vertical repeater station slice, the wires are bundled into groups of fourteen signals between power rails. Each repeater standard cell (marked 1 through 14) is designed to be the width of fourteen wires. Therefore a stack of seven by two repeater standard cells is needed in order to repeat all of the Metal 4 wires in a metal 4-power grid pitch. The RPS station can be as long as needed as the template just repeats itself. The strong advantage of this design concept is that the repeatability of the wires crossing the station is guaranteed thus allowing the possibility of late changes in the design to be accommodated.

The RPS is also designed to be able to accommodate repeaters for metal layer 4 (m4); m5 and m6 interconnect layers. On Itanium(tm) processor™ the m4 and m6 signals run in the same direction. Software determines which cells the repeater solution needs to ensure there are no m4/m6 collisions, that is to ensure a m6 wire isn't "double-booking" a repeater location reserved for the m4 directly beneath it. To avoid such cases the software has the concept of "borrowing". It looks for a free repeater standard cell within a defined distance of the collision and uses that location instead. Once again the repeater standard cell library needs no adjustments in order to fit stations that run in the vertical or horizontal directions on the floorplan.

The repeater stations themselves are placed in the floorplan at the top level of the hierarchy. They are not placed inside units, as was the case in previous designs. This is done for two reasons.

- The repeater station does not finalize at the same time as the unit; therefore the unit level completion does not become gated by the repeater design. By keeping the RPS stations at the top level, this problem is avoided and the station design can remain flexible for a longer period.

- An Electrical Change Order (ECO) doesn't require a unit to be opened up. In many cases on Itanium™ a wire needs to use multiple repeater stations, if the repeater solution were to change late in the game or the wire needs to be moved, all the units that have the RPS stations embedded would have to be opened up.

On the Itanium™ processor there are forty five-RPS. Most were pre-placed on the floorplan based on a manual signal plan study of the chip and use of the estimation tools. There are distributed evenly throughout the design and are usually adjacent to unit edges. Figure 2 has a detailed plot of the Itanium™ processor with the RPS shown. While the stations reside next to channels their area is not for free because of the internal wiring requirements in hooking up the repeaters prevents extensive use of metal tracks as feed-through.

Since the RPS reside at the top level it is relatively easy to add, move or resize stations. This has proved to be invaluable as the design progresses and there is a greater emphasis on ensuring all the wires that need repeaters can be implemented. The Itanium™ processor accommodated changes in RPS up-to 3 months before tape release. Through constant iterations of the repeater insertion tools, RPS have been modified and added to where the present placement meets the design demand.

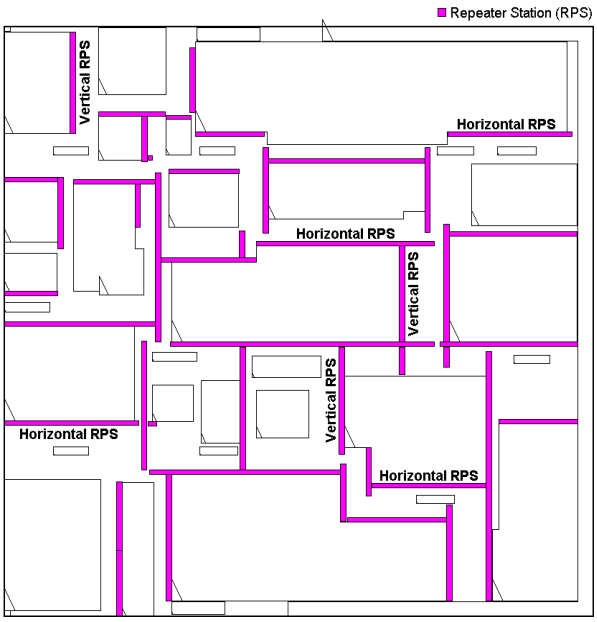

Figure 2: Itanium™ processor floorplan

A fundamental feature of the Itanium™ processor methodology is that the wires in the floorplan are not actually connected to the repeaters until late in the design stage. This allows the wiring design to proceed in the floorplan without the overhead of dealing with number of wires that result from inserting repeaters. A floorplanner can move wires at will, only having to ensure that the resultant move still has the wire crossing a RPS, thereby ensuring repeatability. The estimation software will recalculate the optimal repeater size and location based on the new route. An extensive suite of software has been developed to automatically connect the wires to the inserted repeaters at any point in time in order that netcells can be extracted and sent to the timing model.

Bus breaking automates the repeater insertion into the floorplan database. The tool operates on the layout database, which is maintained in ChPPR, the full-chip layout tool used at Intel. Bus breaker has three major modules, which help automate repeater insertion into floorplan:

- *Bus breaking*: The bus breaking module processes nets one at a time. Given a net and the coordinates where it needs repeaters, first it finds the buses that route over these coordinates. Each such bus is split into two at the specified coordinate. Note that, depending on the coordinate points, a bus may have to be split more than once.

- *Net Renaming*: Once a repeater is inserted in a net, all the layout data that belongs to the downstream portion of this net requires a name change. There are three components to this renaming: *1)* renaming the net in the broken bus, *2)* renaming the net in all the buses tapped into the broken bus, *3)* renaming the terminals on the receivers, if any, on the downstream portion of this net.

- *Repeater interface generation*: At the repeater station full-chip interface, we create pins for IO of each repeater on the RPS. The required schematic connectivity and also the actual layout data for each pin are created. Once the pins and ports are created as such, the ICC router from Cadence is run within each RPS to connect ports to the pins. The interface generator also has novel heuristic to maximize number of nets that can be repeated over a repeater station while minimize the routing distance to the repeater cell.

5. Repeaters and RTL

There are two types of repeaters, those that change the functionality of the signal, such as latches or flops, and those that don't, such as buffers. Any functional change needs to be reflected in the RTL for proper logical operation but it is questionable as to whether it is necessary to insert repeaters in a non-functional case. The Itanium™ processor team chose to insert all repeaters into RTL. The primary reason is that on the Itanium™ processor the RTL is the primary source of connectivity information for all design tools and flows. In particular the floorplan and timing teams receive a set of .sch files (connectivity files) from the RTL integration team and it was decided that these .sch files should have the repeater information so as to ensure consistency between the databases.

The methodology requirement to insert all repeaters has resulted in a mixed approach of manually inserting some types of repeaters and automatically inserting others. A decision flow diagram outlining which approach to use for a particular repeater solution is shown in the figure below.

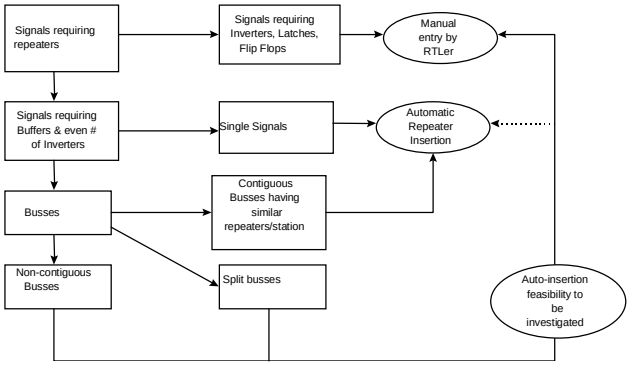

Figure 3: RTL insertion decision flow

The fundamental rule is that any repeater which changes the functionality or polarity of the signal needs to be manually coded in RTL. All other repeaters can be automatically inserted, unless the automation fails, in which case it is manually coded. In general it was estimated that about 10% of the signals would require manual insertion and of the remaining 90%, approximately 4/5 of them could be automatically inserted with the remaining 1/5 needing to use the manual insertion flow.

The essential elements of an RTL insertion flow are: -

- A robust naming convention
- RTL and grafting changes to support insertion. Grafting is the process of integrating an incremental change to the RTL model.

Naming convention: Since a signal may pass through several repeater stations on the way to the receiver it is important to ensure that the propagation history is maintained in the final name. In other words as the designer attempts to trace to origin of the signal he or she needs to be able to derive not only the original driving LBF but also the repeater stations that were used. The naming convention used on the Itanium™ processor is: -

Signalname_[##F]C

- Signalname is the original signal name that is output of the driving unit.
- ## is the Number of the repeater station
- F is the Repeater function where B=Buffer, I=Inverter, F=Flipflop, O=Other
- C is the coloring information (timing phase information)
- [##F] is added to the signal names at each repeater station.

RTL and grafting changes: All unit level input signals must appear in the unit's lbf.param file (a file that describes the I/O). In order to support the lbf.param file and also to enable easier renaming in the future these signals need to be scoped (defined) in the unit.sig (a file that describes the unit I/O) file. To enable unit level grafting, the RPS needs to be at the unit level. This allows easier grafting, removes dependencies of signals from other units and aids RTL simulation performance. When generating connectivity the designers would then use a specially generated xxx_external.sig file, which contains the new signal names and ensures that all the repeated signals are connected correctly.

With the naming convention in place and the support infrastructure in the RTL, it becomes an automation challenge to do the automatic renaming. The Itanium™ processor team developed the necessary software to successfully complete this task. The manually and automatically inserted repeaters are both coded the same way and into the same repeater stations. The insertion flow is shown in figure 4.

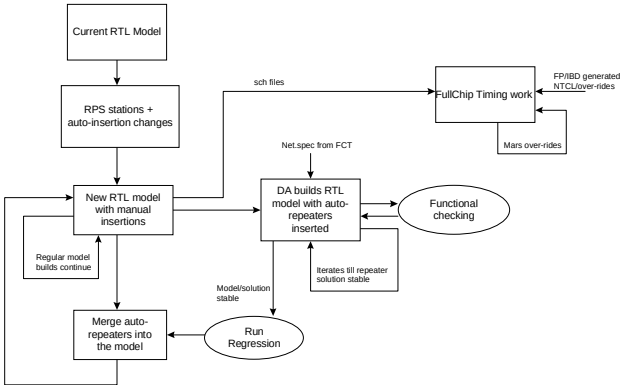

Figure 4: RTL insertion flow

The flow chart shows how the repeater stations are added to the RTL, and the manually coded repeaters are inserted into the "live"

model. This is needed as the manually inserted nets change the functionally of the signal and therefore the full regression test suite needs to be run. The auto-inserted repeaters are inserted into a parallel model to allow several iterations and changes to occur without destabilizing the production model. At a given point when the RTL has stabilized and the repeater solution is also stable both models are merged and this becomes the production model. The connectivity information of the chip comes from this production model.

It should be noted that the auto-inserted repeaters can be completely removed from RTL and the code is returned to the exact state as it was before insertion.

6. Physical implementation of repeater stations

In order to physically implement the repeater stations and generate timing data, a schematic file needs to be generated. No visual schematics for repeater stations are drawn so the file is generated from the RTL model. During the insertion process the repeater size and cell name is coded. This is similar to a physical-block implementation in synthesis. With this information it is relatively simple to generate a .sch file with the relevant information to generate a file that can be fed into an auto-place and route tool.

The Itanium™ processor team developed its own placement tool. The reason was the desire to have a deterministic placement of cells in order to ensure there was no additional routing inside the repeater station. This would ensure there was no surprise parasitic data that would impact the timing of the signal. The tool reads in the repeater information and populates the station. The cells that are not used are filled with decoupling capacitors. Once the cells are placed, the CCT router is used to complete the hookup. The layout is then ready for parasitic extraction.

The circuit design loop for the stations is relatively straightforward. Since the cells themselves are not too complicated the LBF doesn't have too much trouble passing all the design checks such as Static Electrical Rule Checks (ERC), Decoupling capacitor checks, Reliability checks (EM and SH) etc. The only major constraint of the repeater station design is that it gets finalized right before tape release. Therefore the design needs to be fast and require little rework. This process has been automated such that 1 designer can spin all 45 repeater stations through the entire circuit verification flow (including extraction) in 24 hours using distributed computing

7. Incorporation of repeaters into the physical integration flow

While pure estimation works fine for a long period of time, there comes a point where there is a need to insert the repeaters into the actual floorplan and to extract a new set of netcells that have been connected to the repeater stations. A Physical Integration (PI) is the process of rolling up the layout data of the chip into the fullchip floorplan, completing top level routing, extracting the netcells and building a fullchip timing model. A PI needs to be completed in order to generate a complete set of netcells for repeater analysis. Mars is then used to analyze the netcells and generate a set of repeater solutions.

This repeater solution then needs to be applied to both the floorplan and RTL to see if it can be automatically inserted. There can be many reasons why this insertion process can fail and there is extensive sharing of data between the RTL and Timing and floorplan teams on what signals were inserted and what ones were not. Several rounds of debugging are needed to ensure the highest success percentage. The complexity and amount of interaction between these groups rises significantly during the repeater insertion phase. Once the insertion has been completed in both the RTL and floorplan, a new set of netcells needs to be extracted.

The process outlined above means it takes additional time to generate an accurate set of netcells for timing data.

The penalty for repeater insertion is not paid every PI. Since the PI is a significant feedback mechanism for the designers it is important to make the turnaround time as fast as possible but on the other hand it is also important to give the design community the most accurate data. The Itanium™ processor team devised a way of meeting both requirements by inserting repeaters in a parallel floorplan model thereby allowing the basic PI loop to continue but at the same time inserting repeaters and giving that data to the timing model a little later. The model for this is shown.

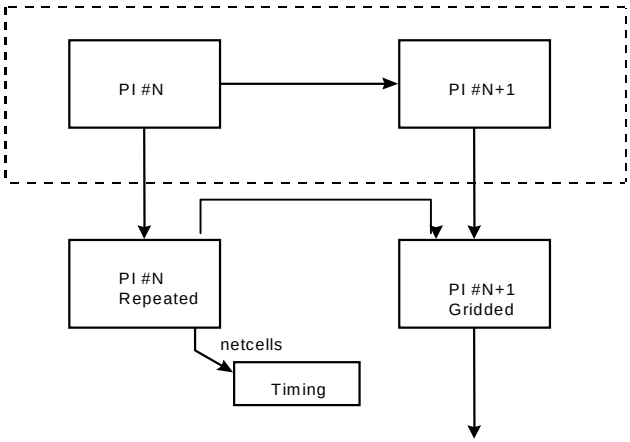

Figure 5: PI plus repeater insertion

The solution is that once a PI is completed the floorplan database is copied to a directory where it is post-processed to get the netcell data. The floorplan and the design are free to continue its work on the "live" database in preparation for the next PI. Net cells are extracted from the frozen database and sent to the timing model. The same netcells are used to determine the repeater placement and they are inserted in the frozen floorplan. Netcells are once again extracted and sent to the timing model. At time of the next PI, repeater solutions that haven't changed can be reused again which gives the timing model a level of stability. Over a period of several PIs the timing model gets to see all the nets inserted into the floorplan.

The final problem that needs to be solved is the case where the designer wants a different repeater solution but doesn't change the functionality of the signal. The new solution could be because of a tight timing requirement that Mars doesn't seem to be able to meet. The methodology allows the designer to generate a user specification of a repeater solution (a uspec) and this spec overwrites any solution that Mars generates. The floorplan and

RTL insertion tools will accept these uspecs and they are treated just like an automatically generated specification. The uspec capability is very powerful as it gives the ability to custom design every net on the chip if necessary and still uses the insertion automation in the floorplan and timing. It therefore gives a high degree of flexibility in design to incorporate last minute changes due to final timing tweaks or robustness fixes. An important limitation on uspecs is that they can only be used on signals whose functionality is not being changed.

8. Results

The methodology described in the preceding pages has been used on the Itanium™ processor

There are about 13k nets routed at the top level of Itanium(tm) processor™. Of these 11k require a repeater of some type, that is they route to a repeater station. Of these 11k, about 7.5k need a buffered repeater and can therefore be implemented in the automated flow. So over 50% of the nets on the chip will have an automatic RTL and floorplan insertion flow applied to them. There are about 3.5k nets that used the manual RTL insertion flow and these are inserted and working in the design today. The average number of repeater drop points on a net is about 1.5, which means that the number of wires that result once the repeaters are hooked up is around 3x. That would mean the final count of nets at the top level of the Itanium™ processor would run in the order of 25k.

The number of RPS on the chip is 45 and these are evenly divided between vertical and horizontal stations on the floorplan. The total area consumed by these repeater stations is about 6% of total die area.

The % utilization of the RPS is also an interesting number and on the Itanium™ processor it never rises about 80%. The average utilization for RPS on the chip is around 40%. While this number is low, it is not clear that area could be given back to the units as in many cases once the RPS is created it can be lengthened without impacting the die area.

9. Summary

Repeater insertion is becoming a critical activity for today's microprocessor design. The Itanium™ processor team has developed a comprehensive methodology and tool suite that allows for insertion into the RTL, Layout, Floorplan and Fullchip timing databases. The methodology is fully integrated with all other aspects of the chip design. The advantages of the Itanium™ processor approach are:-

- Hides the repeater problem from the LBF design.
- Allows for the LBF and floorplan designs to proceed concurrently for most of the design cycle before the repeater solution gets locked in.
- Allows the designer to craft his or her own solution that can be easily integrated into the automation flows
- Functional elements such as latches can be inserted along side buffers.
- Ensures the connectivity information that all design tools use is current and has repeaters inserted.
- Ensures the RPS stations can be easily modified up until the last change in routing gets completed and then the circuit and physical implementation of the stations can be quickly done.
- Successfully pipelines the repeater analysis and insertion flows in such a way as to avoid slowing down the top-level floorplan and timing work. This is a critical

10. ACKNOWLEDGMENTS

The authors would like to recognize the following people for their invaluable contributions, John Benoit, Jim Bergman, Wenliang Chen, Ravi Eakambaram, Mamun Rashid, David Kidd and Sudhakar Bhat in contributing to the development of the methodology.