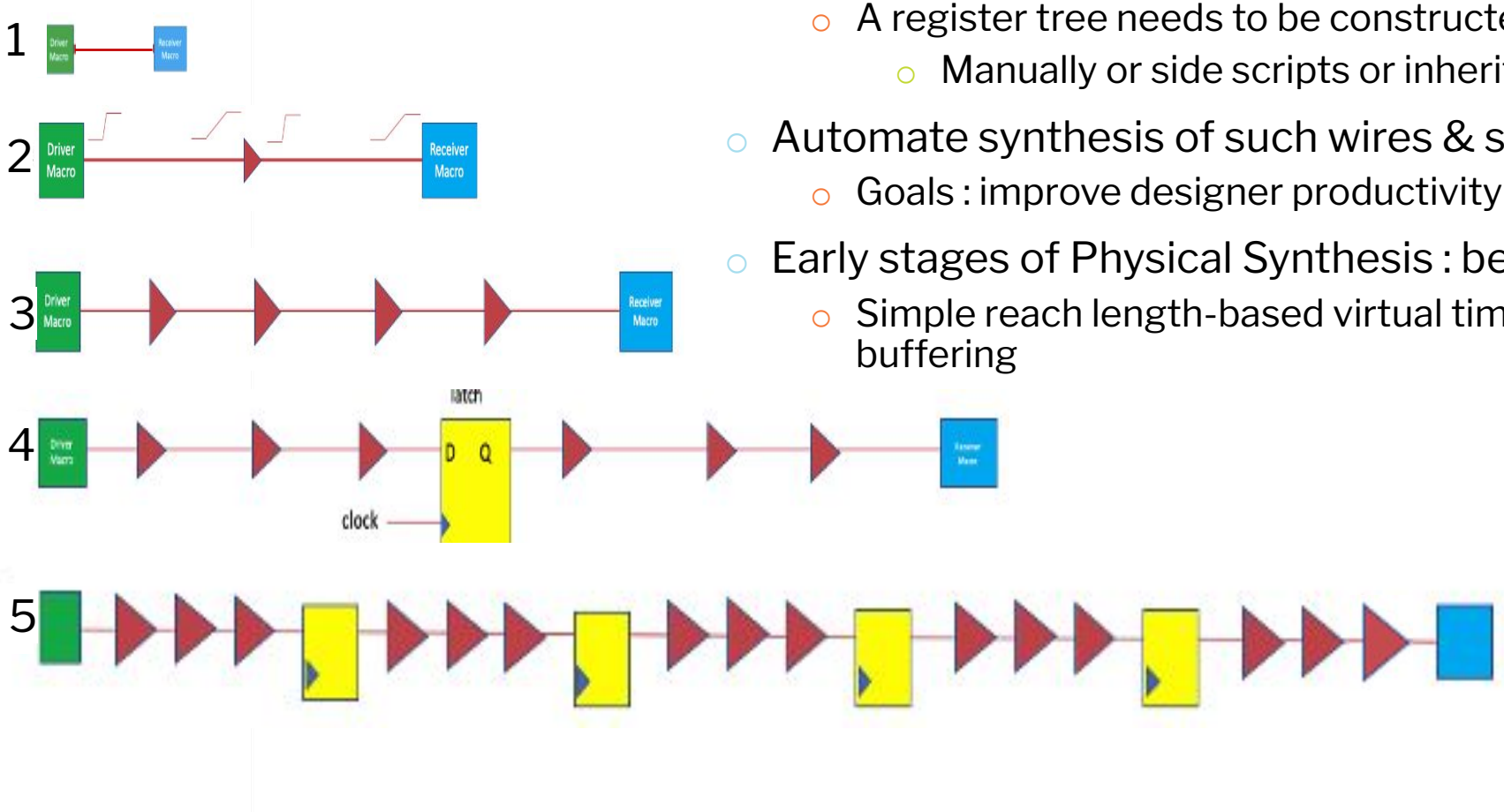

# Multi-Cycle Interconnect Synthesis (MCIS) : (data) register tree construction

Nicolas Degors, Josiah Hamilton, Jinwook Jung*, Nany Kollesar, Arjen Mets, Gi-Joon Nam*, Lakshmi Reddy*, Greg Schaeffer, Paul Villarrubia, Nancy Zhou

*IBM Research, IBM Systems

# Motivation

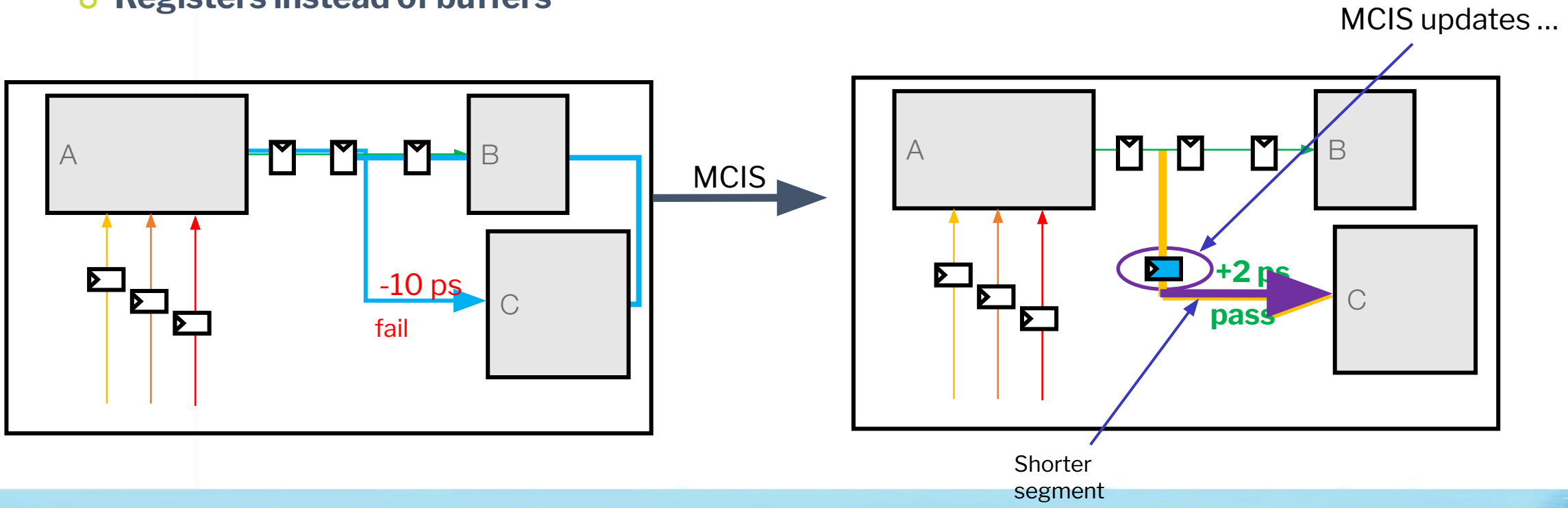
- At chip-integration level or in very large designs:
  - Some data signals travel very long distances
  - Need multiple clock cycles to get to their destinations
  - A register tree needs to be constructed to synthesize such wires
    - Manually or side scripts or inherited from legacy designs …
- Automate synthesis of such wires & satisfy the given constraints
  - Goals : improve designer productivity & QoR
- Early stages of Physical Synthesis : before CTS & buffering
  - Simple reach length-based virtual timing model that assumes ideal buffering




Technology progression

# Multi-Cycle Interconnect Synthesis (MCIS)

- Given:
  - Root net/driver & cycle delay constraints for each sink
  - Register type & routing layer resource available to build the register tree
- **MCIS goal:**
  - **Build *efficient* multi-cycle register tree & satisfy given constraints**
  - **Analogous to buffering**
    - **Registers instead of buffers**

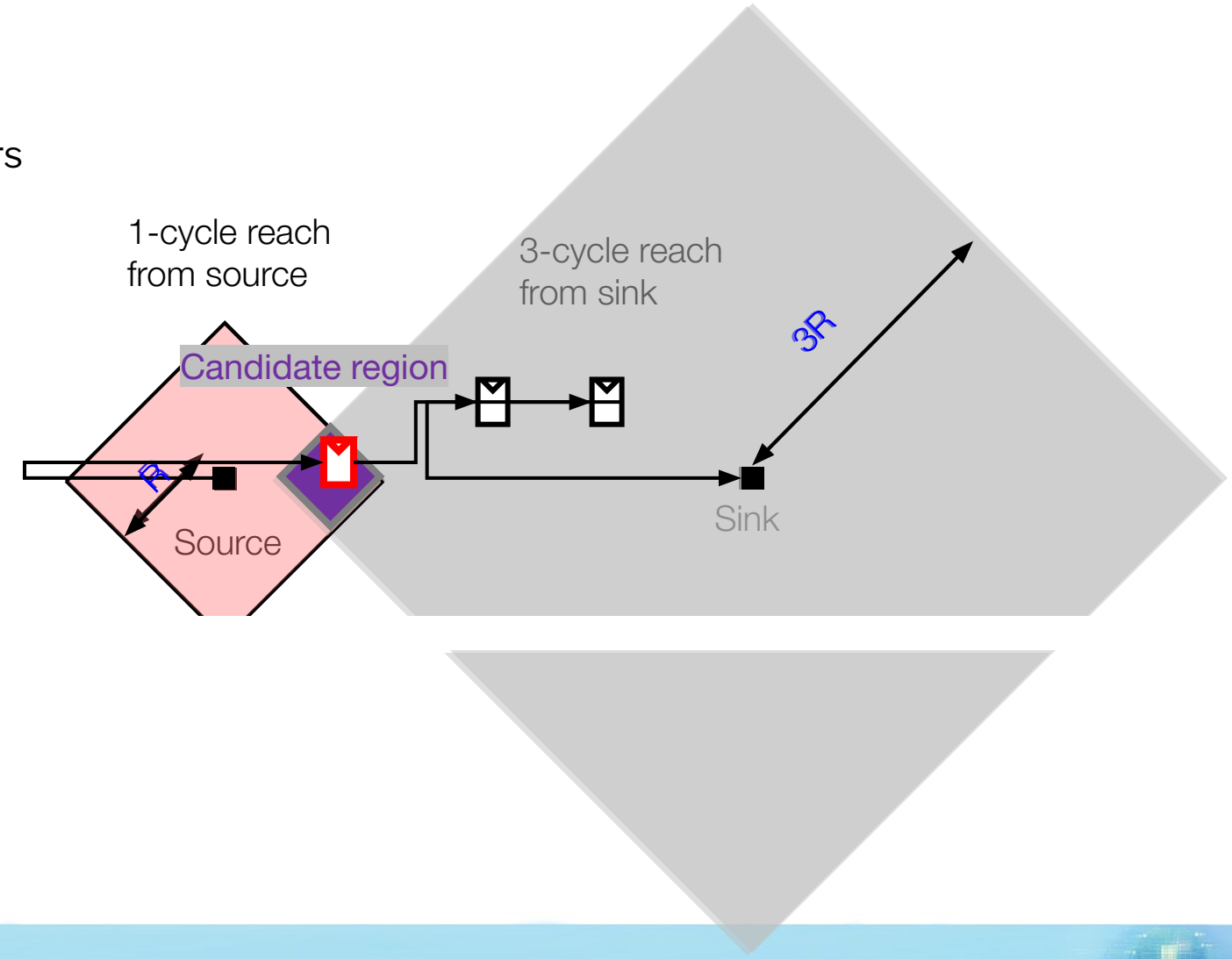


# Register Tree Construction

- *Layer-trait* : specific routing layer, width, & spacing
- *Cycle Reach Length*: ***cr***
  - Given a layer-trait:
    - max distance covered in a single cycle
      - ideal buffering is assumed
  - Tree is built with a single type of layer-trait …
- **Step 1** : Layer-trait selection (from a library)
  - *minimum-reach(**mr**) needed:*
    - over all source/sink pairs :
      - max of { source-sink Distance *D* / number of cycles constraint for the sink }
  - select a layer-trait with min cycle reach ***R*** : ***R*** > ***mr***

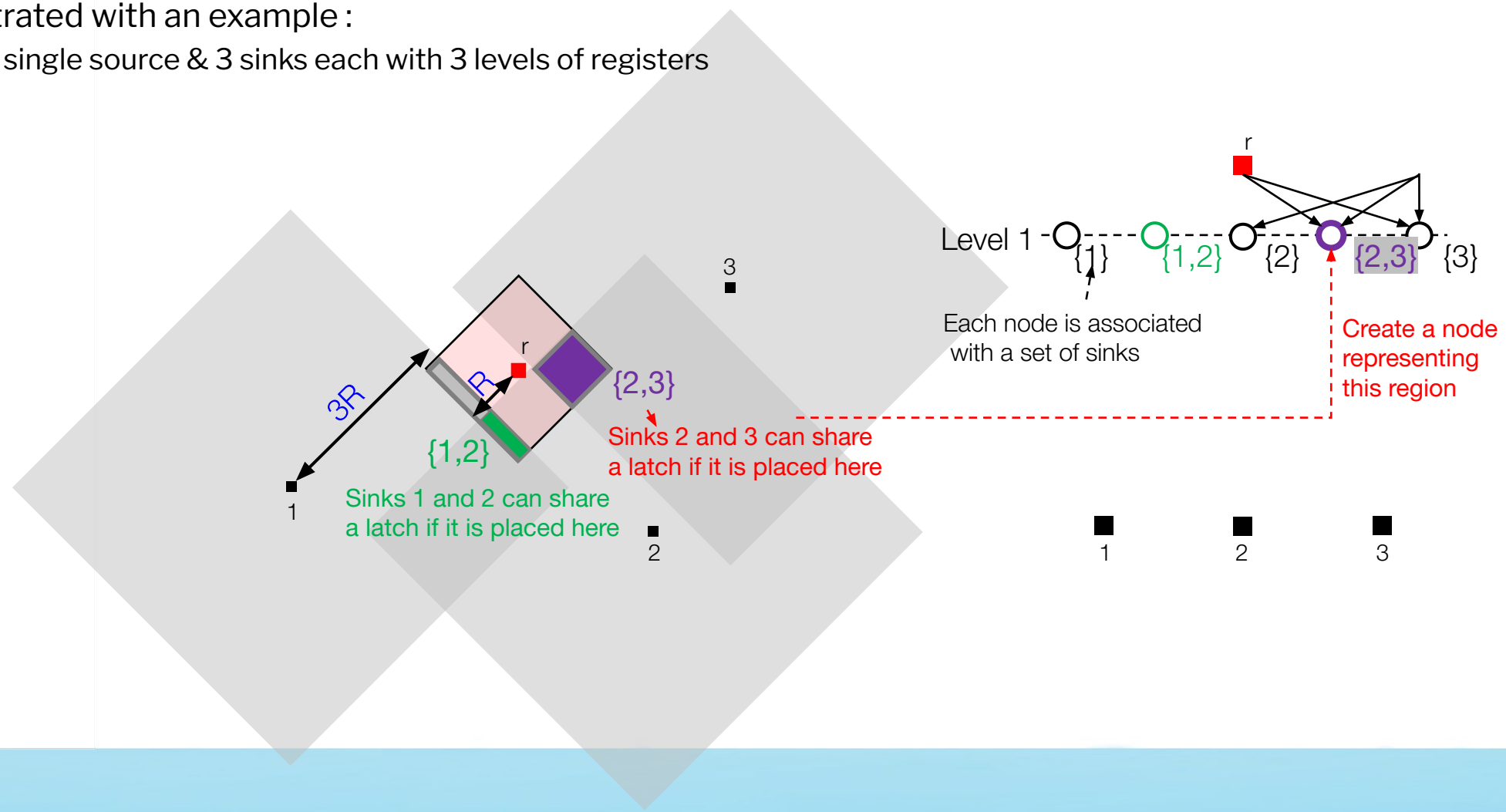
# Register Tree Construction

- Computing 1st level candidate register location
  - Feasible region
  - Illustrated with an example :
    - single source/sink with 3 levels of registers
    - assume source/sink register bounded
  - diamond shapes (Manhattan circles)



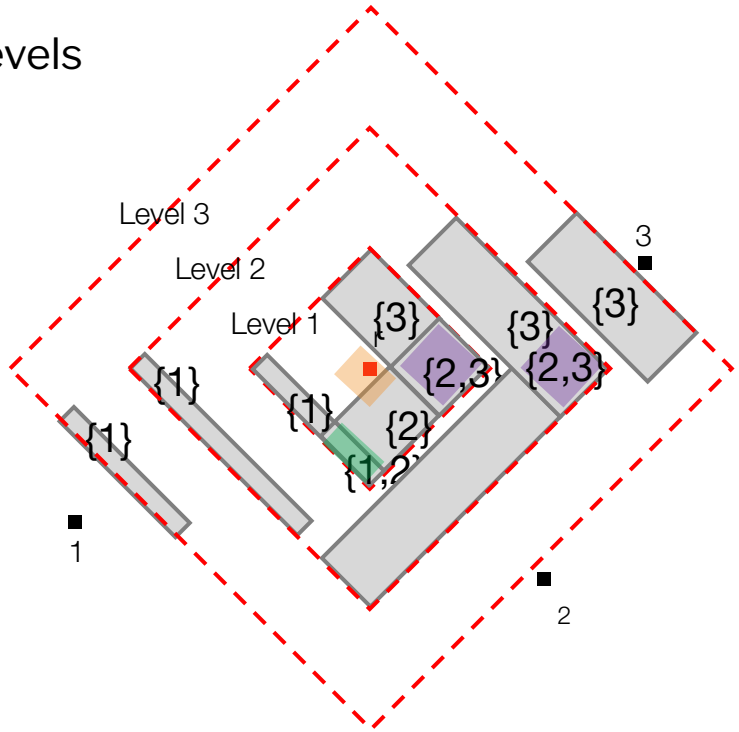
# Finding Movable Regions – Registers for Level 1

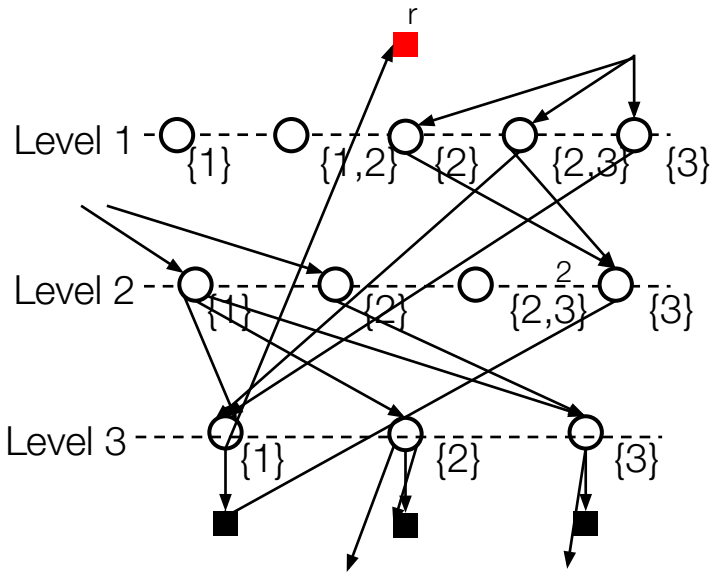
- Draw diamond shapes (Manhattan circles) to cover source and sinks
  - At source (r) with radius R, at each sink with radius 3R
- Create nodes for each identified candidate region (overlaps)
  - Illustrated with an example :
    - single source & 3 sinks each with 3 levels of registers



# Step 2: Topology Search Graph Generation

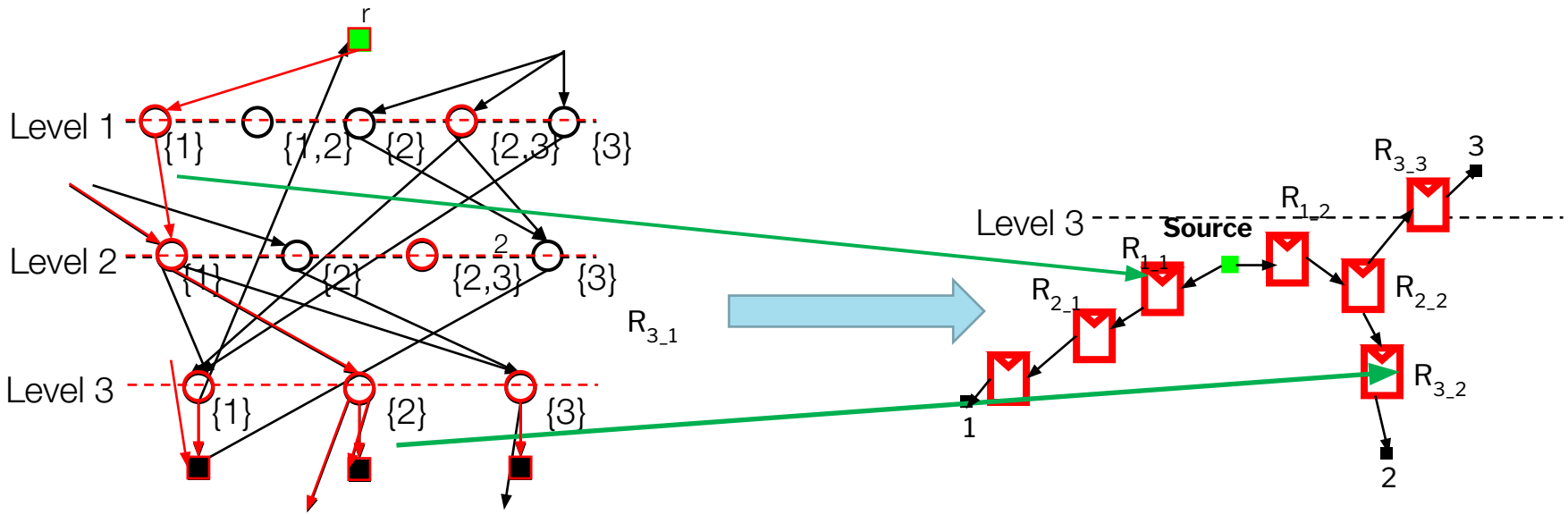
- At every level, compute candidate regions for registers
  - Draw Manhattan circles (diamond shapes) for sources & sinks
    - Radius depends on the level and is a multiple of ***R***
  - Create a node for each overlapping region of diamond shapes
    - Gives the region where a register may be placed connecting source to covered sinks …
  - Complete Graph by processing all levels



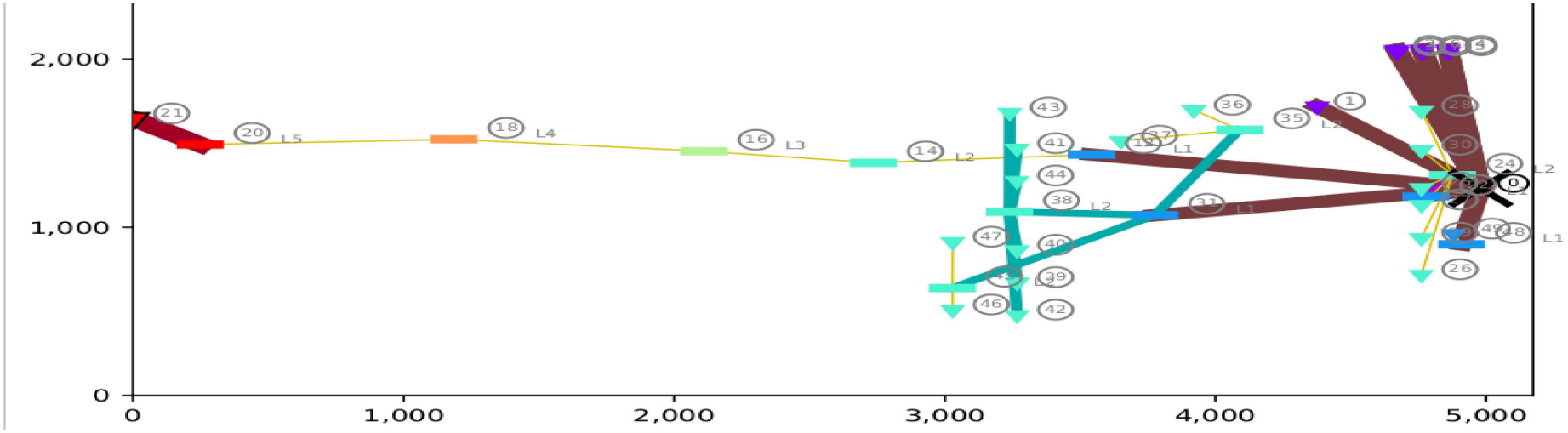
# Step 3 : Register Tree Creation

- At each level, select nodes to cover all sinks
- Goal : minimize number of nodes (registers) selected
- Exact cover NP-complete : use greedy heuristics
  - Example : nodes with higher overlap area & so on …



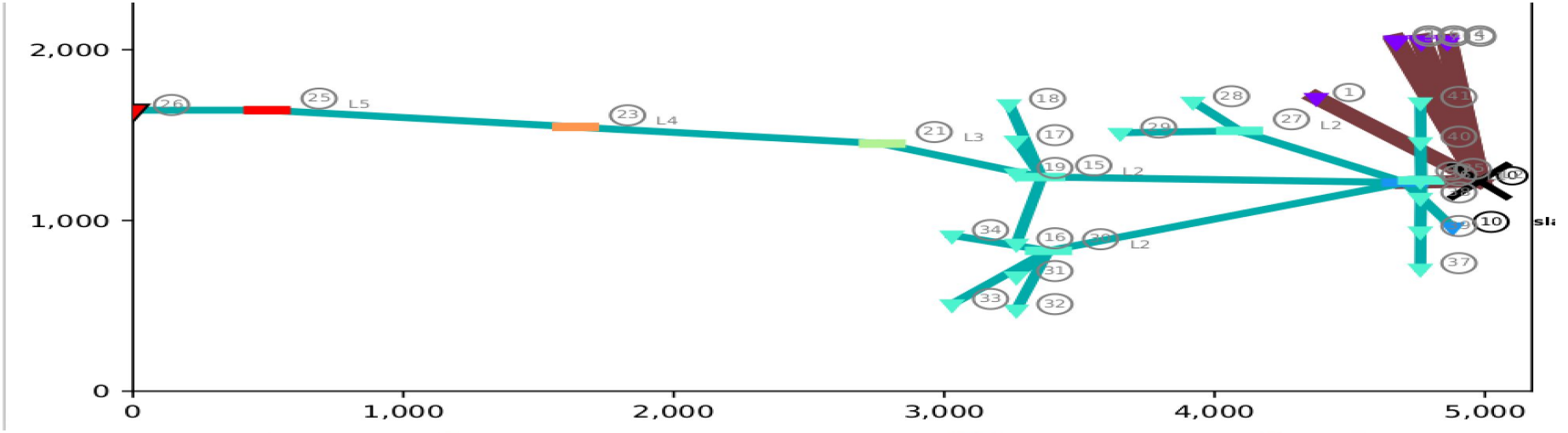
# Tree Example :

***Before:*** WL: 13146, WSLK: 16.4, Latch: 11



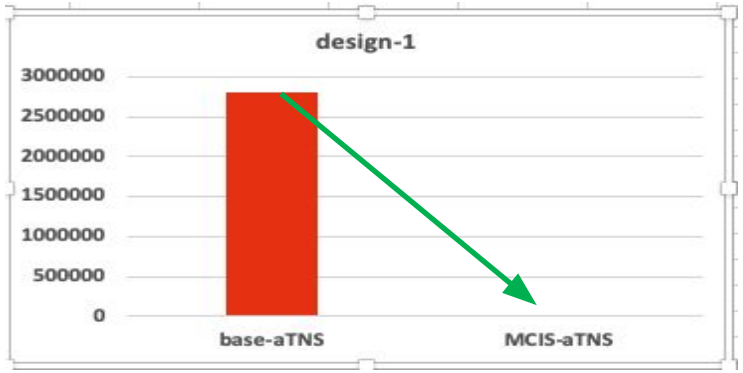

*Wire colors due to subsequent routing layer opt. Please ignore.*

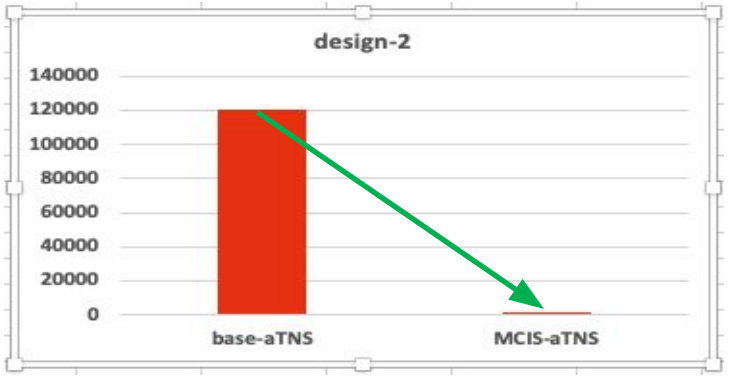
***After:*** WL: 11212, WSLK: 0.3, Latch: 7

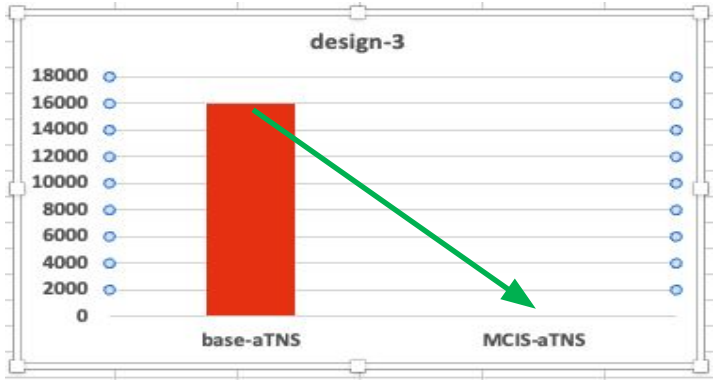


# Experimental Results

- Sample results for 3 designs:
  - Targeted signals with initial skeleton register trees
    - only 1 register per level
  - All MCIS run times under an hour / added 1000+ latches



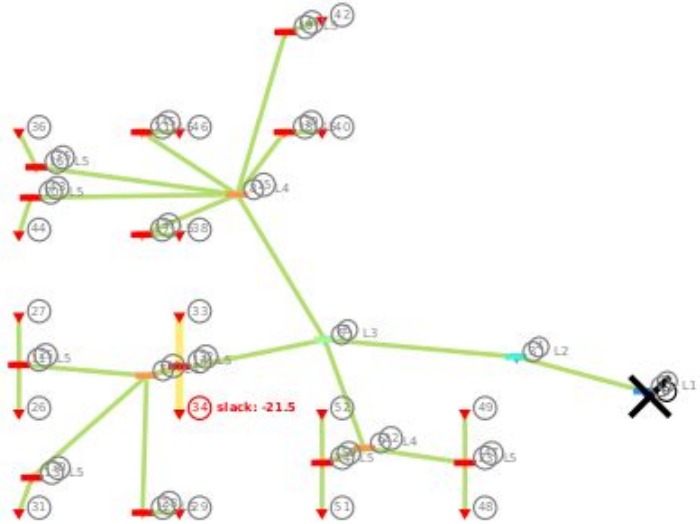

- **Some designer feedback:**

***aTNS : absolute value of Total Negative Slack in pico-seconds***

*"… made a significant difference in timing as well as … help (reduce) congestion. ….having it all automated to run at the chip (level) is nice vs having … to do manually."*

*"the benefit for me is I don't have to care anymore about the manual cloning and maintenance … I used to do on <design-xyz>. Moreover, the tool provides a good solution .. which is directly applicable. If any changes … a simple update … and I am good. I like it!"*

# Summary




- At chip-integration level or in large designs:
  - Some data wires need to travel large distances over multiple clock-cycles
- Multi-Cycle Interconnect Synthesis (MCIS) feature
  - Automates register-tree construction to synthesize such wires to:
    - Improve designer productivity & QoR
- Proposed method involves:
  - Generates a topology-search-tree that captures possible solutions for a register tree
  - Uses a simple greedy heuristic to create a register tree from the search-tree
  - Post-processing transformations performed to further improve QoR
- Results:
  - Viable register trees : quality comparable to manually built versions / Fast run times
- Future work:
  - Currently, MCIS honors existing number of registers to each sink
    - Designers may specify number levels based on estimates / trial & error / legacy designs
  - Taking automation to the next level:
    - MCIS decides number of register stages to each sink
      - Based on the latest physical information
- Acknowledgements : Thanks to Adam Matheny & Walter Nop for the support/advice