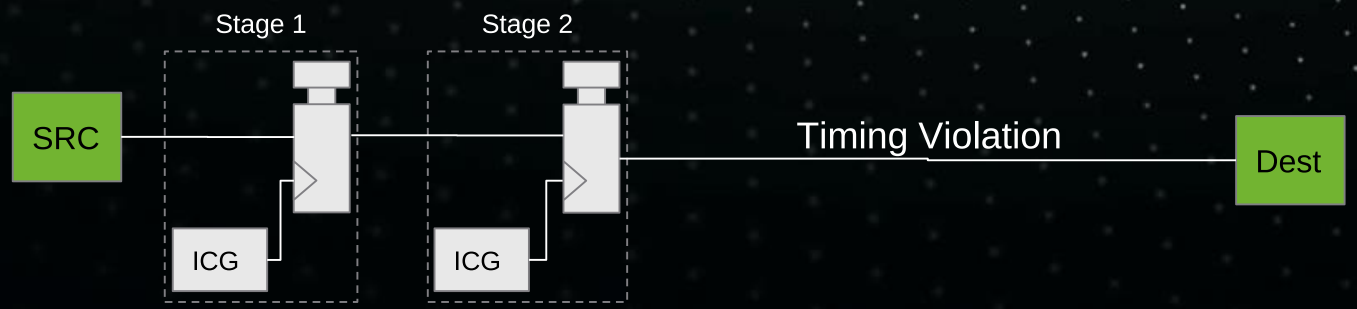
Tile-Based Automated Pipeline Floorplanning

Greg Ford - greg.ford@averasemi.com

averasemi

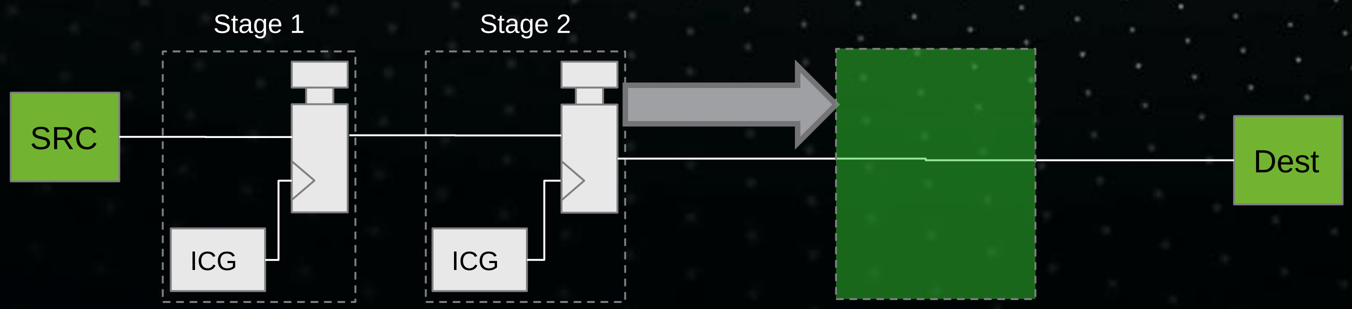
Background

- As designs get larger, there is an increased need to pipeline signals traversing longer distances
  - Pipeline stages can become a system level performance bottleneck, so being able to insert an optimal amount is critical
- General placement algorithms often produce unbalanced pipe placement
  - Some stages with large negative slack, and others with large positive slack
- Most pipeline planning requires fixed start/end points, and pre-placement



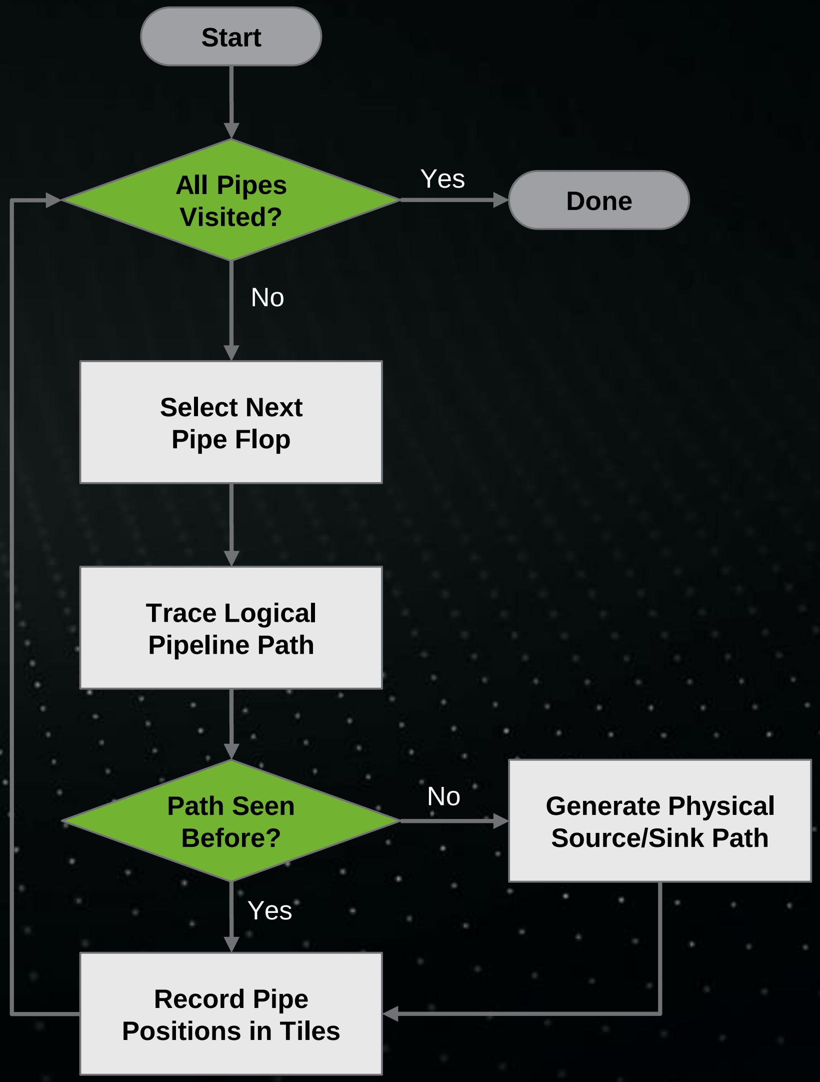
Goals

- Guide placement so pipe stages are well spaced along path
  - Pipeline paths will need to be logically traced to understand the association between source/destination and interceding pipe stages
  - Design will need to be placed (though not optimized), in order to know source and destination locations to guide placement along the physical path
- Assess pipeline path depth to ensure optimal stage count
  - Gauged by maximum stage distance, based on clock frequency and ps/mm characteristics of repowering.



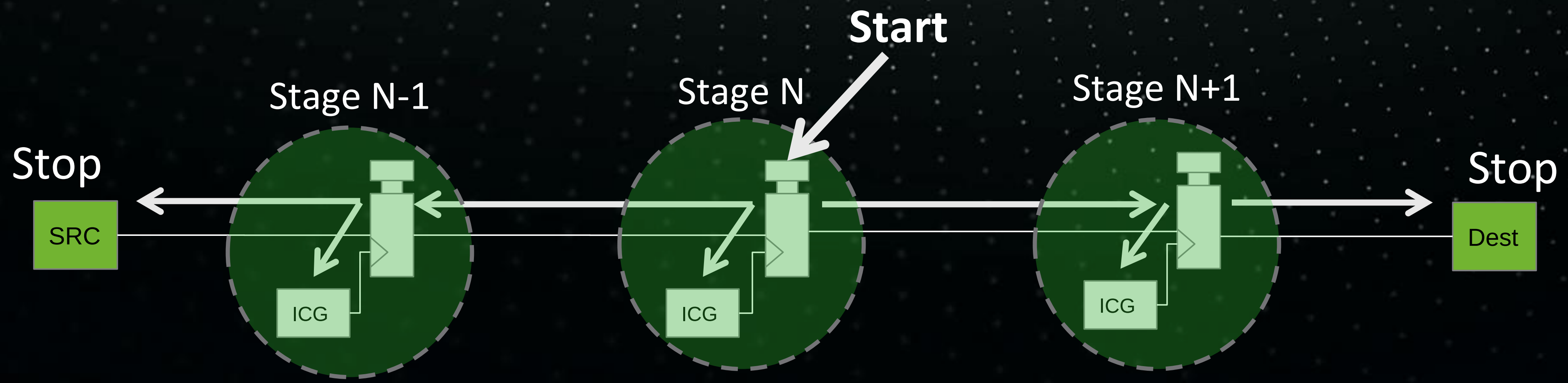
Pipeline Extraction and Tile Assignment

- Start with coarse global placement
  - No assumption that locations for source / destination of pipeline paths are known
- Iterate through all pipeline flops by name
  - Trace forward / backward to overall source / destination for the pipelined path
  - Generate a physical placement path between source and destination
    - Store result for entire path; no need to process the same path multiple times
  - Select balanced positions along physical path
    - Abstract positions into local tiles



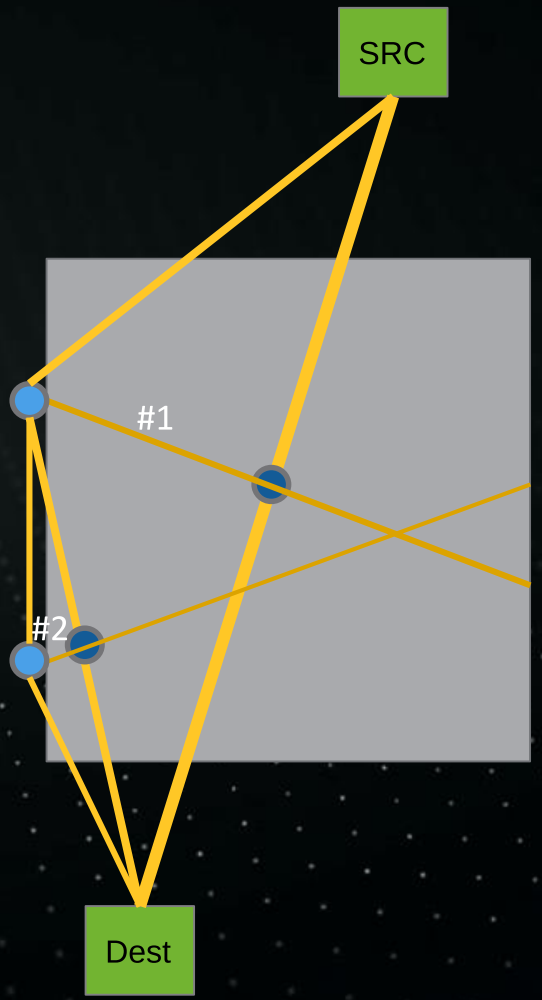
Logical Pipeline Tracing

- Start tracing at an arbitrary pipe stage flop (found by name pattern)
  - If flop's clock is gated by an ICG, include the ICG in the current stage group
  - Trace backward to next pipe stage flop and repeat
  - Trace forward to next pipe stage flop and repeat
    - Ignore scan path connections
  - Stop when no further pipe stages are found; record source/destination
- Repeat for each pipe stage flop in the design
  - If a starting pipe stage flop is recorded as already being traced, skip



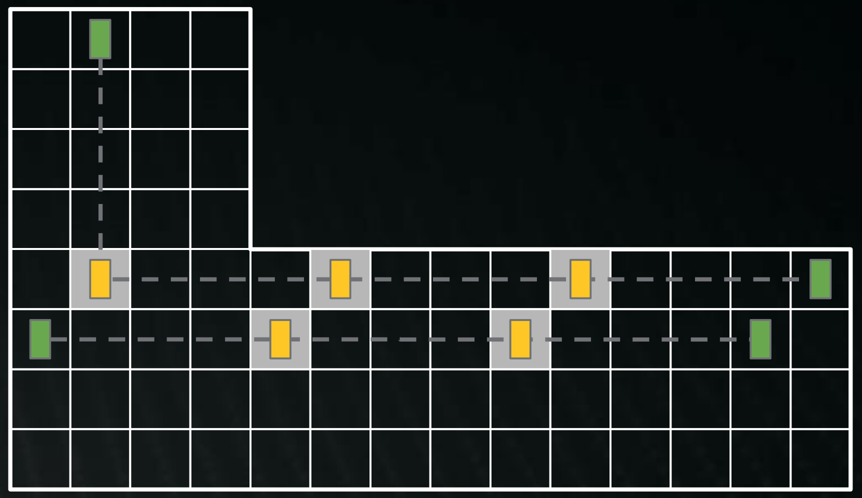
Physical Path Generation

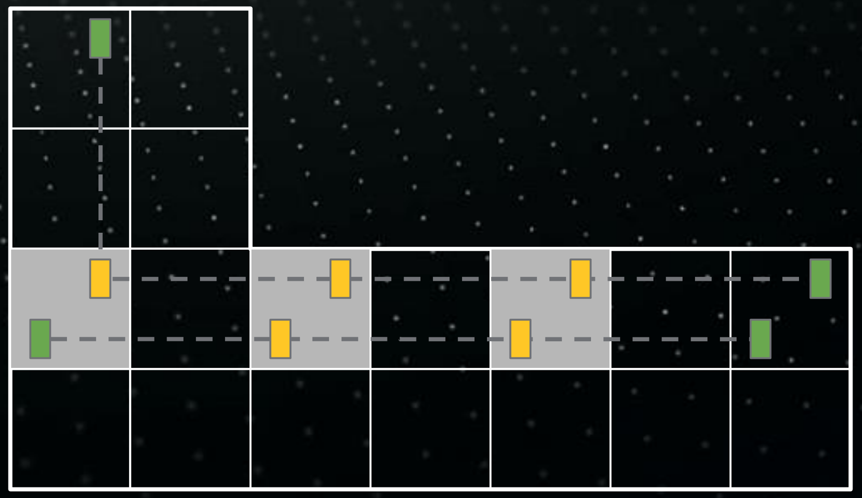
- Start with a straight line from source to destination
- Select midpoint of current segment and legalize
  - If point is blocked, draw a perpendicular line and trace to edge of blockage; select the point closest to original location
- Recurse for new sub-segments, until sub-segment length is less than threshold
- Further detail available in DAC2015 presentation “Fast Custom Repowering”



Physical Paths and Tile Efficiency

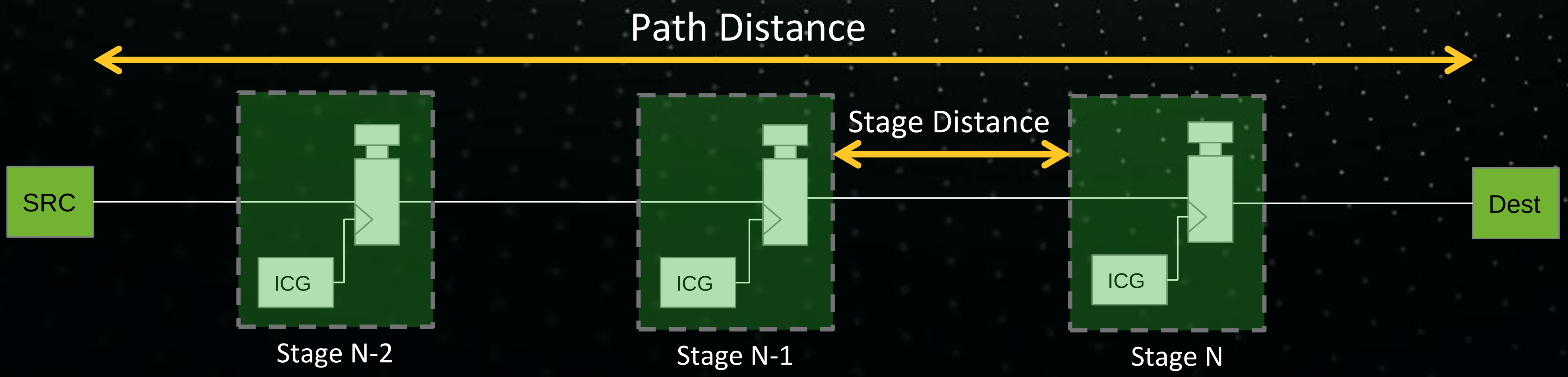
- Local tiles are used as placement constraints to give global placement a degree of flexibility in its solution
- Tile size allows for a tradeoff in flexibility
  - More tiles add to placement complexity and increase tool runtime
  - Scale up tiles based on overall path lengths
    - Deeper paths need less granularity
- Tiles simplify physical path generation
  - Once a path has been generated between two tiles, all other paths between the same tiles reuse the same physical path





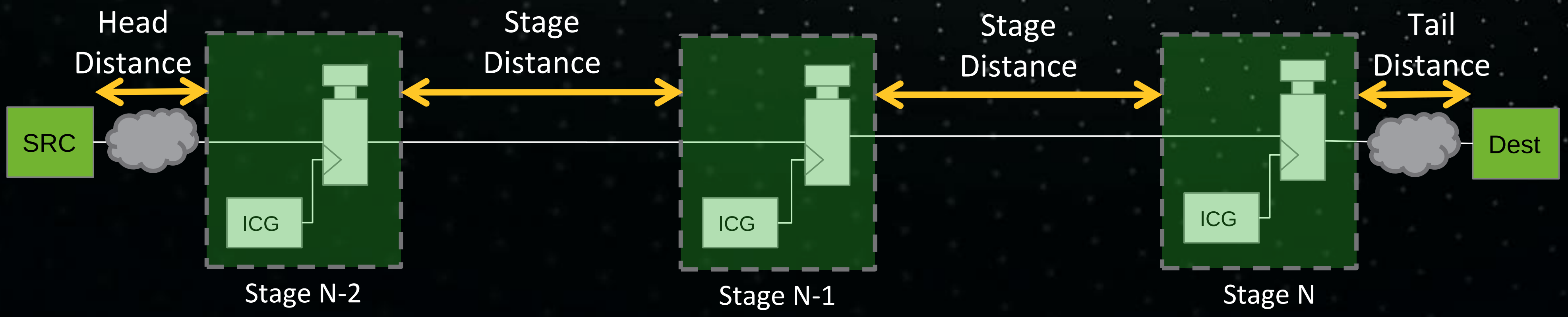
Optimizing Pipeline Depth

- The optimal number of pipe stages for each traced source / destination path can be computed
  - Start with the total physical distance traced from source to destination
  - Determine maximum stage distance from clock frequency and repowering ps/mm
  - **Stages = ceil(path_dist / max_stage_dist) - 1**
- Pipe stage distance can be audited once placed, to ensure it remains bounded by the assumptions used to compute optimal depth



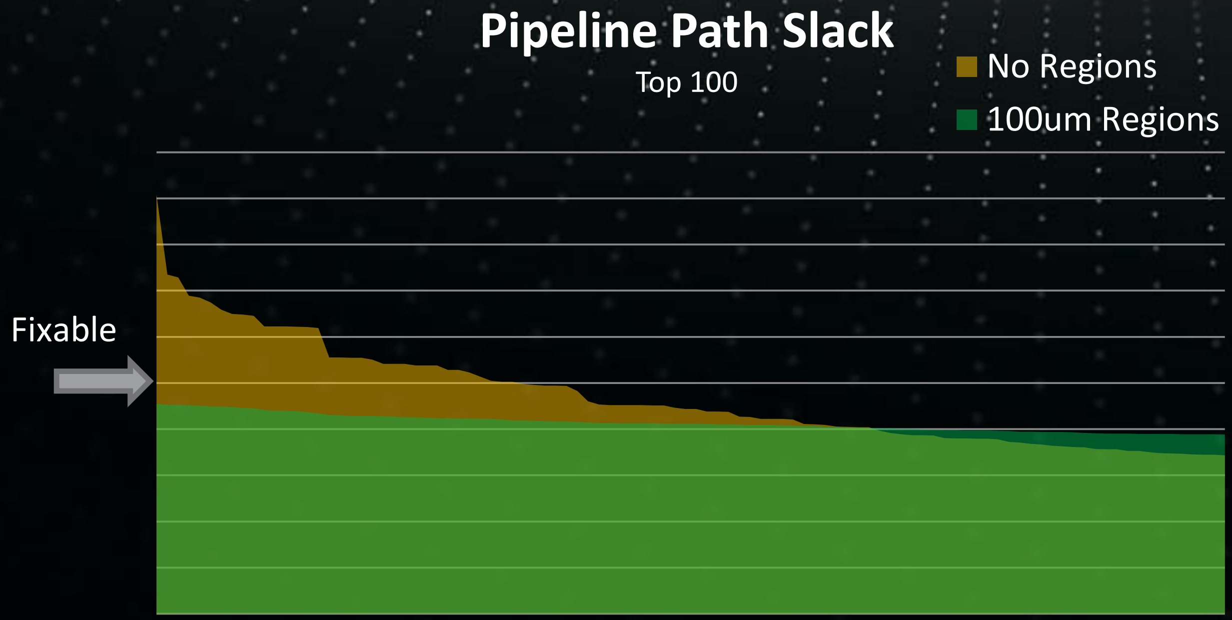
Optimizing Pipeline Stage Spacing

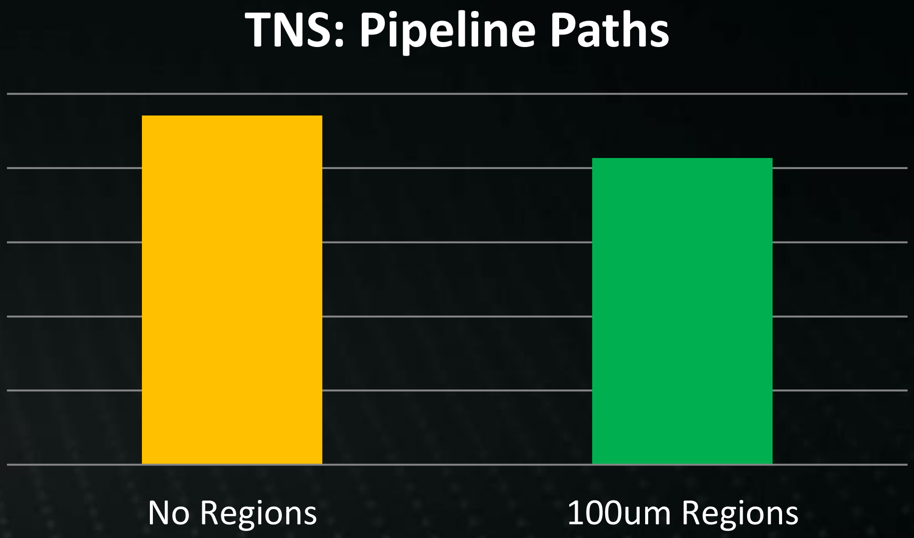
- Even spacing of pipe stages from source to sink allows for the lowest possible number of pipe stages in the path
- Often, there will be combinational logic in the path from the source and/or the path to the destination
  - In this case, even spacing may not be possible, as source and destination segments may not be able to meet timing
  - Use maximum stage distance between pipe stages to cover distance as efficiently as possible, and use a lower head/tail distance to give extra time for logic depth

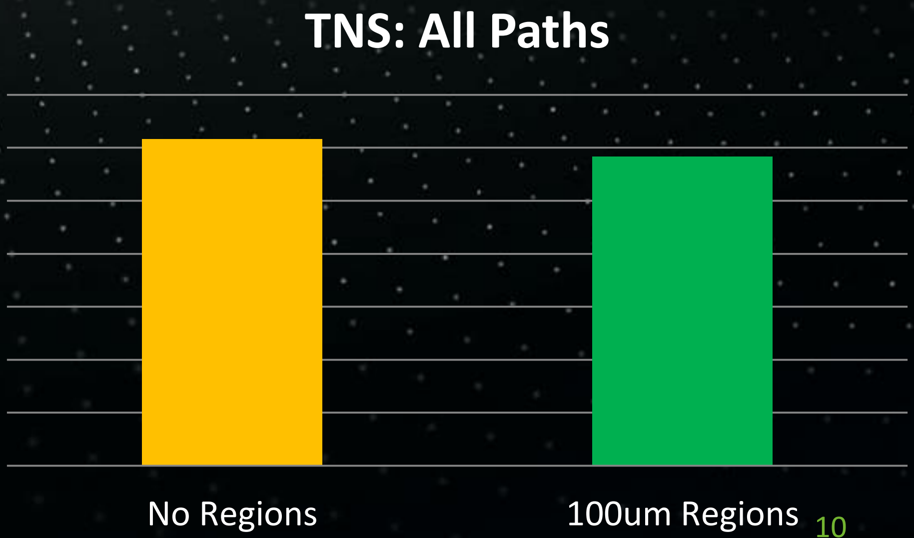


PreCTS Timing Results

- Runs completed using 100um pipe tile regions
  - 10% higher runtime with regions, in global placement phase
  - WNS cut in half; pushed below fixable threshold
  - TNS for pipeline paths reduced by 12%
  - TNS for all paths reduced by 5%

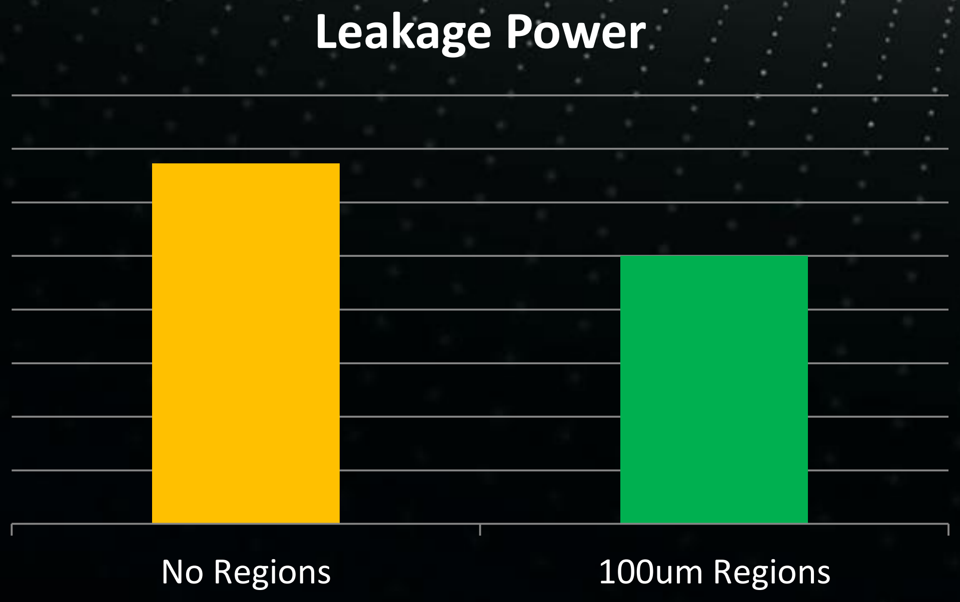


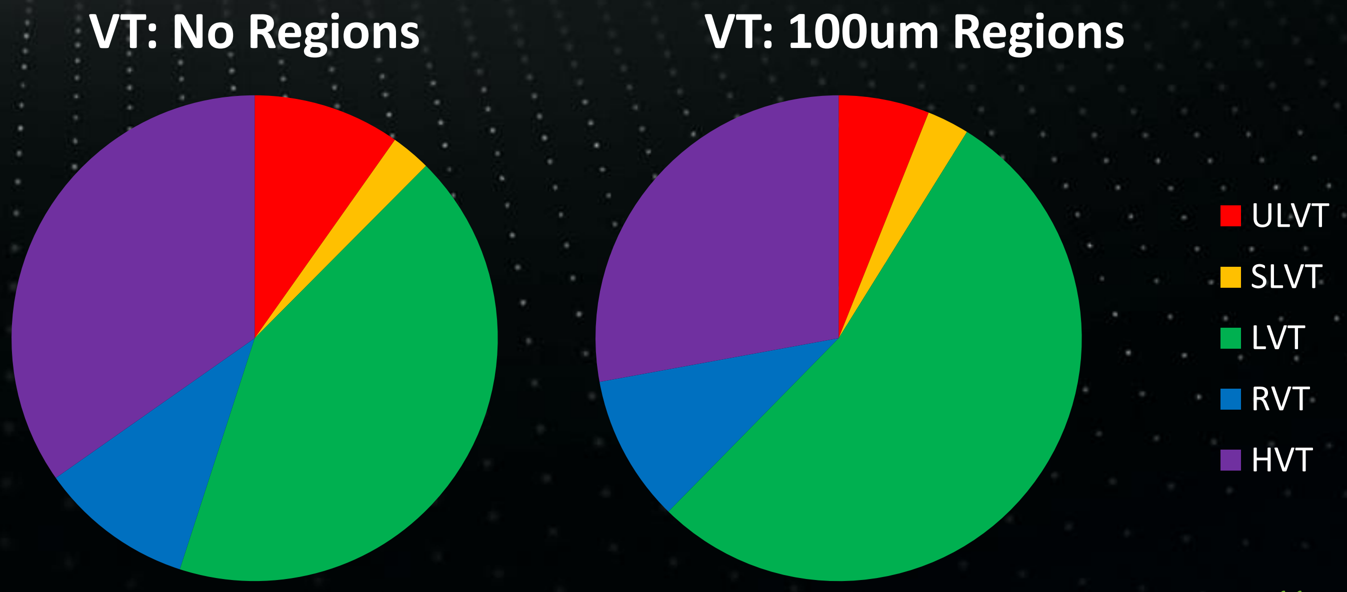




Post-Flow VT / Power Results

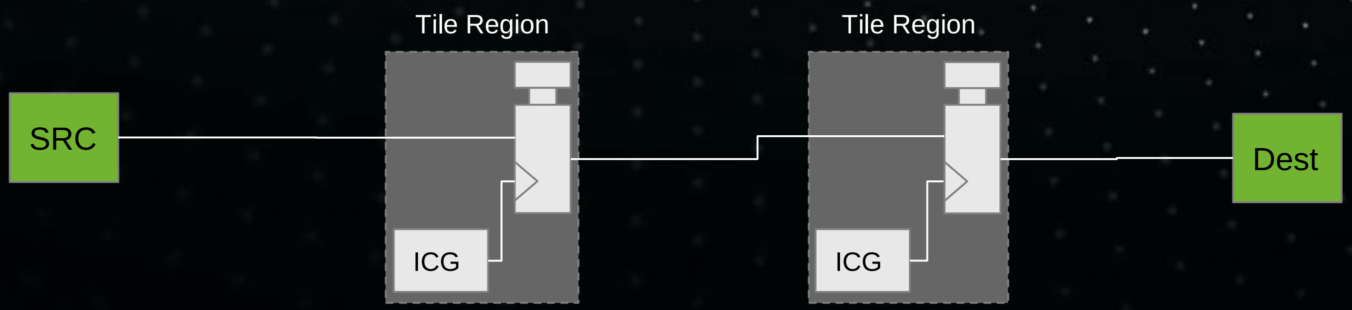
- Runs with and without regions taken through CTS/Routing flow for final optimization comparison
  - Using pipeline regions reduced leakage power by 26%
    - 20% Lower HVT usage compared to baseline run
    - 38% Lower ULVT usage compared to baseline run





Summary

- Use of pipelining for timing closure is becoming more common as designs continue to get larger
- Placing pipeline stages balanced along their path allows for closure with an optimal number of pipeline stages
  - Using excess pipeline stages can impact system performance
- Tracing pipeline physical paths, and pre-assigning pipeline stages to tiled placement regions allows for an automated balanced solution
  - Reductions in WNS/TNS/Power, with a slightly higher initial placement runtime



Thank You