
# Delay-driven Layer Assignment for Advanced Technology Nodes

Szu-Yuan Han[1], Wen-Hao Liu[2], Rickard Ewetz[3], Cheng-Kok Koh[4], Kai-Yuan Chao[5], and Ting-Chi Wang[1]

[1]National Tsing Hua University, Hsinchu, Taiwan
[2]Cadence Design Systems, Austin, TX, USA
[3]University of Central Florida, Orlando, FL, USA
[4]Purdue University, West Lafayette, IN, USA
[5]Intel Corporation, Hillsboro, OR, USA

# Outline

- Introduction
- Problem Formulation
- Parasitic Extraction
- Our Algorithm
- Experimental Results
- Conclusion

# Introduction

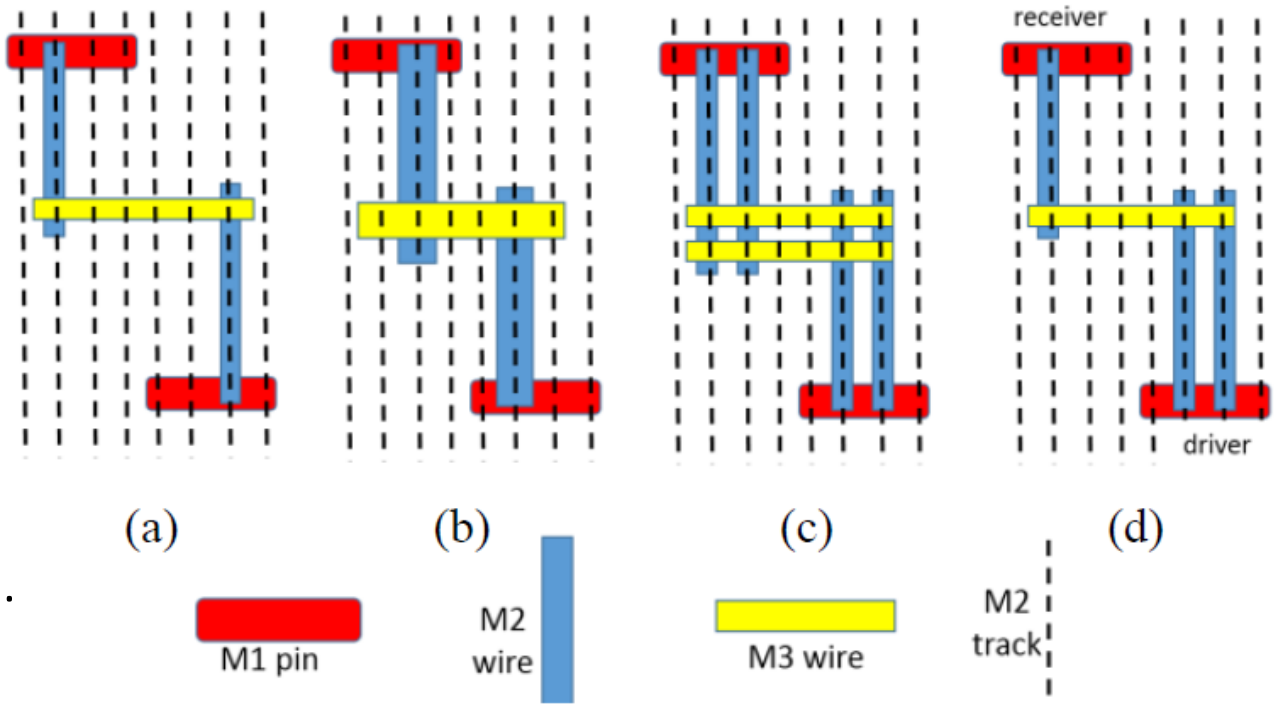
- In advanced technology nodes, interconnects are realized in a multi-tier layer structure, where each tier of layers has a different default wire width and spacing.
- Layer assignment is one of the critical steps during global routing to affect interconnect delay.
- A challenge for layer assignment is how to effectively assign wires of timing-critical nets to upper layers while avoiding routing congestion.
- In addition, layer assignment also needs to consider the timing impact from vias and wire coupling effect.

# Non-default-rule Wires and Parallel Wires

- Due to manufacturing limitations, wires in advanced technology nodes cannot have arbitrary widths, and can only be of certain pre-defined special widths.
- Wires that use a special pre-defined width are called non-default-rule (NDR) wires.
- Moreover, lower layers are manufactured by multiple patterning lithography such that NDR wires have to be realized by parallel wires instead of wide wires.
- The parallel wire technique uses multiple and parallel default-width wires to route a connection, which is similar to using a wide wire.

# Examples of Using Default-width Wires, NDR Wires, and Parallel Wires

- Examples of routing a net using
  - (a) Default-width wires
  - (b) NDR (double-width) wires
  - (c) Parallel default-width wires
  - (d) Parallel wires partially

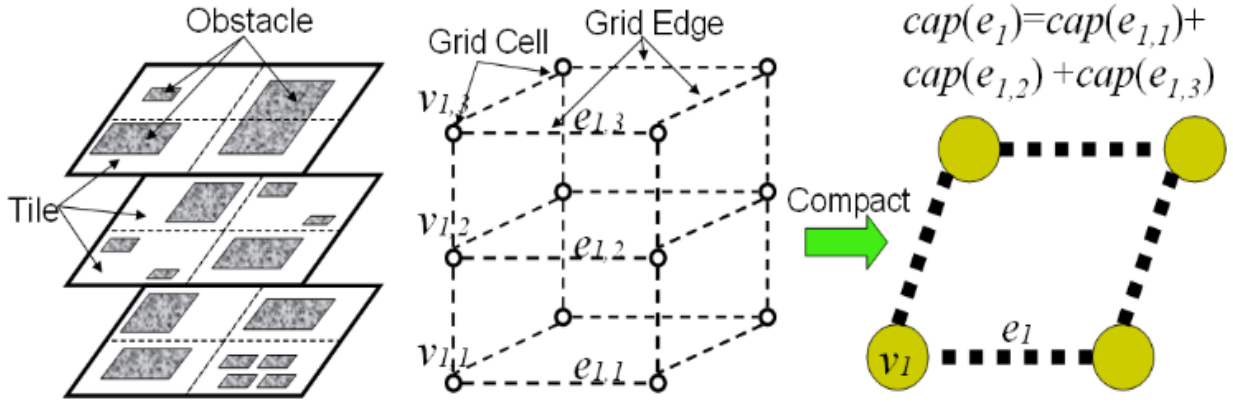


# Contributions

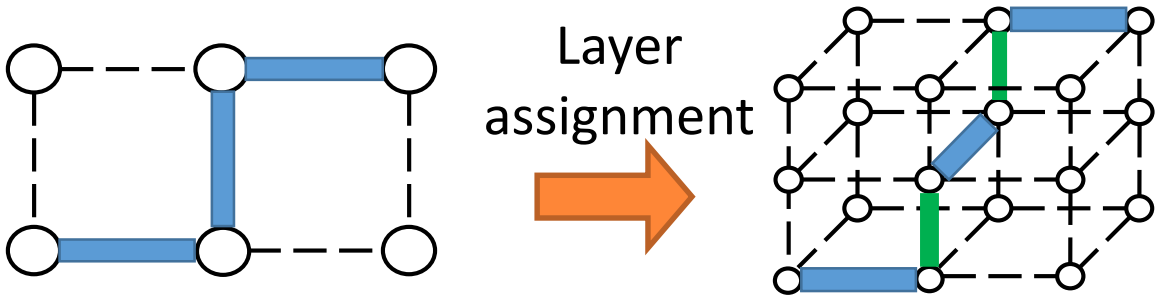
- This paper addresses a delay-driven layer assignment problem that considers via delay and wire capacitive coupling as well in the global routing stage.
- A probabilistic model is proposed to estimate wire capacitance including coupling effect.
- A negotiation-based layer assignment algorithm is developed to strike a good balance between delay, congestion, and via count.
- The proposed layer assignment algorithm also effectively utilizes NDR wires and parallel wires for further delay reduction.

# Problem Formulation

- Input:
  - A 2D grid graph, which is compressed from a *k*-layer 3D grid graph
  - A 2D global routing *S*




- Output: a 3D global routing $S^k$ through layer assignment

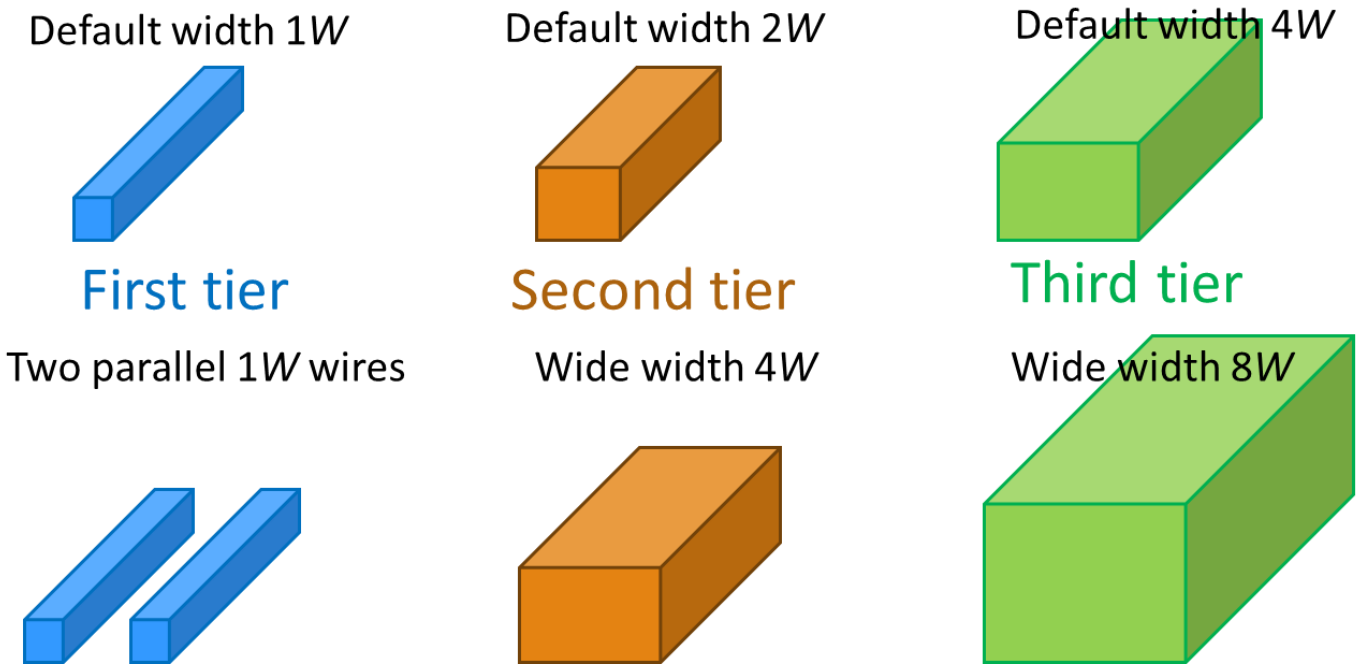


# Problem Formulation

- Wire congestion constraints:
  - $Total\_Overflow(S^k) = Total\_Overflow(S)$
  - $Max\_Overflow(S^k) = \lceil Max\_Overflow(S) \times (2 / k) \rceil$
- Objective:
  - Minimizing the top 5% worst net delays and the total net delay
- Delay Model:
  - Interconnect delay is estimated using the Elmore delay model.
  - Net delay: $d(N) = \Sigma_{i \in Sink\_Set(N)} \alpha_i \times d(i)$
  - $\alpha_i$ is set to $1/|Sink\_Set(N)|$

# Problem Formulation

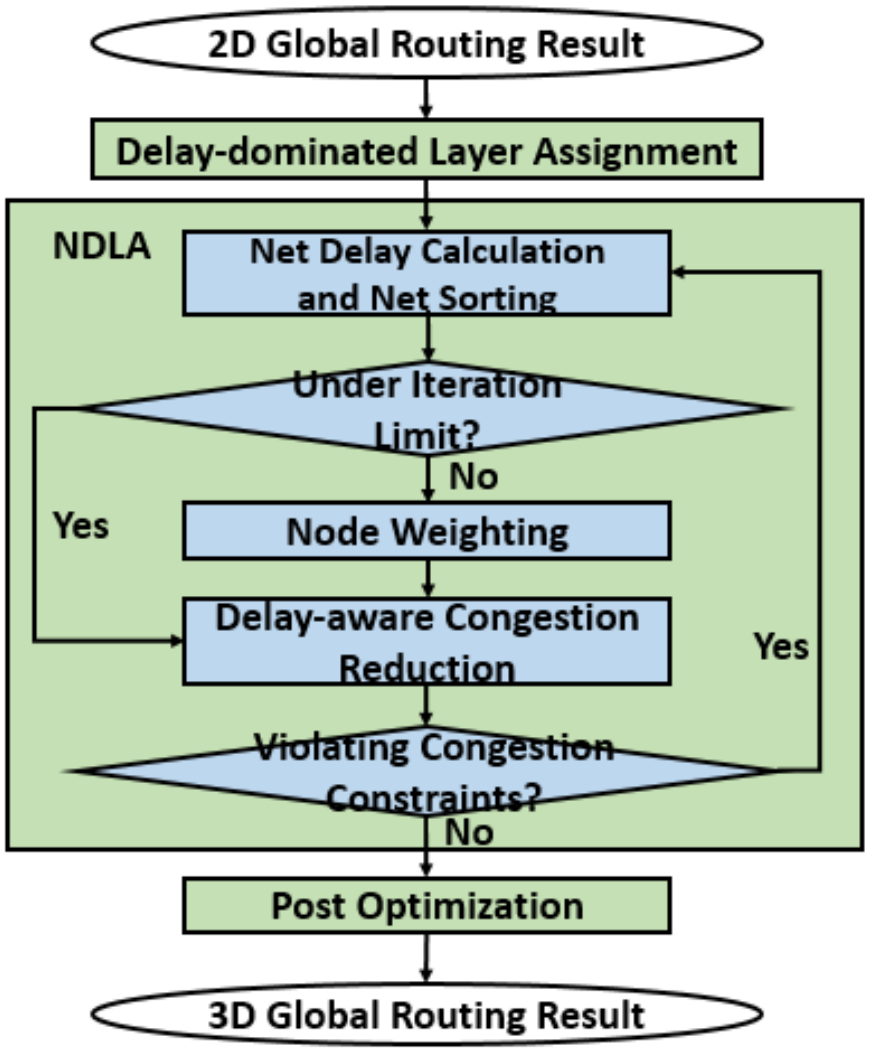
- Multi-tier layer structure
  - Three tiers of metal layers respectively from layers 1 to 4, 5 to 7, and 8 to 9.
  - Default widths of the first, second, and third tiers of layers are respectively 1*W*, 2*W*, and 4*W*.
  - Parallel and NDR wires



# Parasitic Extraction

- The unit resistance of a default-width wire is set according to the ITRS roadmap.
- The unit resistances of parallel wires and NDR wires are set based on the roadmap under the assumption that every wire has the same lining thickness.
- The probabilistic unit capacitance for wires are pre-calculated and stored in a lookup table (LUT).
- Wire capacitance extraction
  - Each entry (w.r.t. a wire segment of type *t* assigned to a 3D grid edge of density *d* on layer *z*) in the LUT is computed in a probabilistic manner.
  - Wire type could be default-width wire, two parallel wires, or NDR wire

# Our Algorithm




- Delay-dominated layer assignment (DLA)
  - Initial layer assignment without considering wire congestion constraints
- Negotiation-based delay-driven layer assignment (NDLA)
  - Iterative layer re-assignment to fix the violation of wire congestion constraints
- Post optimization (PO)
  - Re-assigning each net once for further quality improvement without violating wire congestion constraints
- A single net layer assignment algorithm is adopted in all three stages but with different objective functions.

# First Stage: DLA

- Each net *N* is assigned the best layers to minimize the following objective function without considering the wire congestion constraints.

$$\beta \times d(N) + \#via_N$$

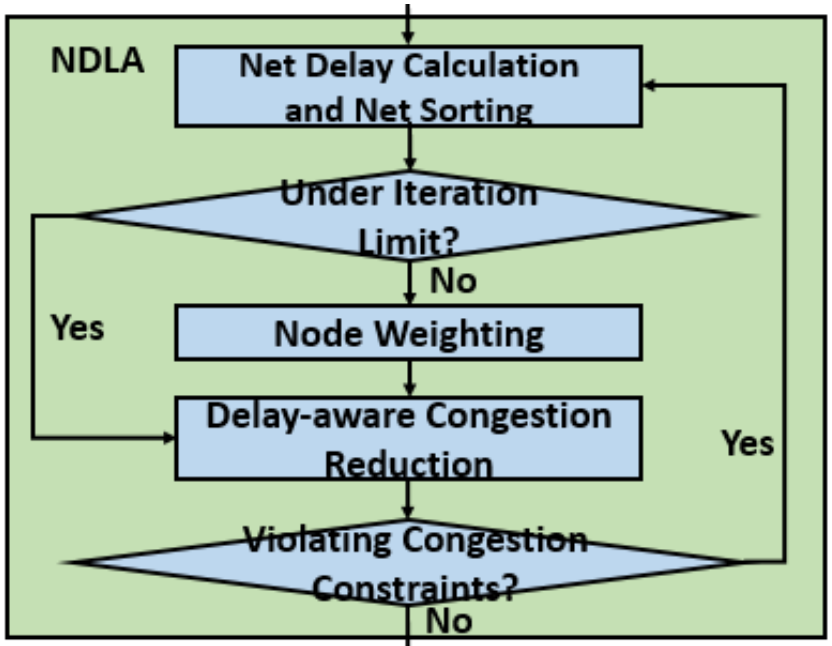
- Parallel wires and NDR wires are not used in this stage to avoid additional routing resources consumed by them at this early stage.
- Unnecessary overflows in 3D grid edges (causing a violation of one or two wire congestion constraints) and illegal nets could be produced.
- A net is illegal if its 2D routing tree passes through a 2D grid edge such that at least one of the following two conditions is true
  - The sum of the overflow of each corresponding 3D grid edge is larger than the original overflow of this 2D grid edge
  - The largest overflow among all corresponding 3D grid edges exceeds the allowable maximum wire overflow

# Delay Calculation

- During the layer assignment of a net, the capacitance of each wire segment with a certain wire type on a certain layer is computed based on a table lookup.
- However, the density of a 3D grid edge is one required item to access an entry from the lookup table, but it keeps increasing as more wire segments of different nets are assigned to this 3D grid edge, making the capacitances of all but the last assigned wire segment become under estimated
- In order to look ahead for considering later coupling change, the density of a 3D grid edge is determined by the larger one between the current density of this 3D grid edge and the density of the corresponding 2D grid edge.

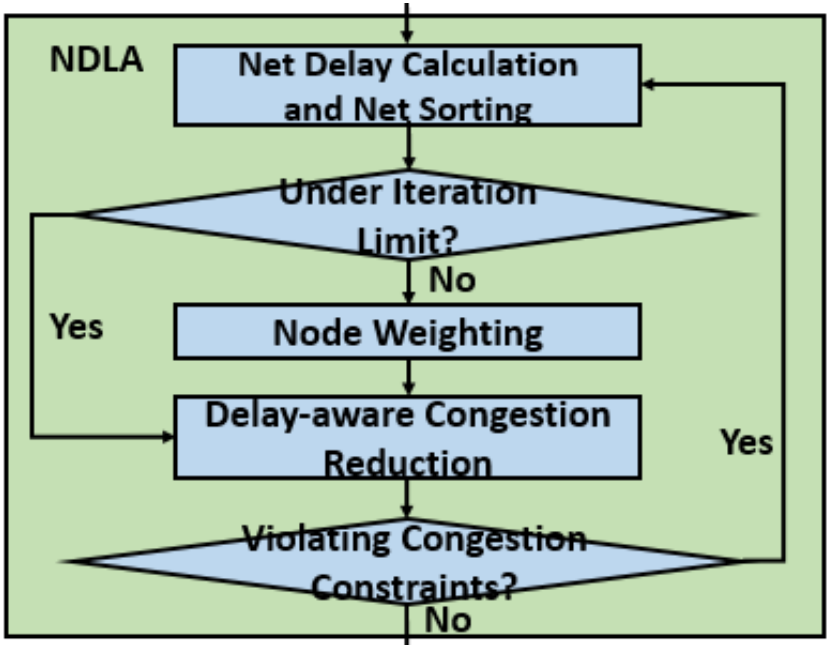
# Second Stage: NDLA

1. Calculate the delay of each illegal net, and sort all illegal nets in a non-increasing order according to their delays.
2. Each wire segment of an illegal net is assigned a weight based on the delays of its downstream sinks from the current layer assignment result.
   - The segment weighting step is skipped after a user-defined iteration count is reached
3. Illegal nets are all ripped up and then re-assigned net-by-net



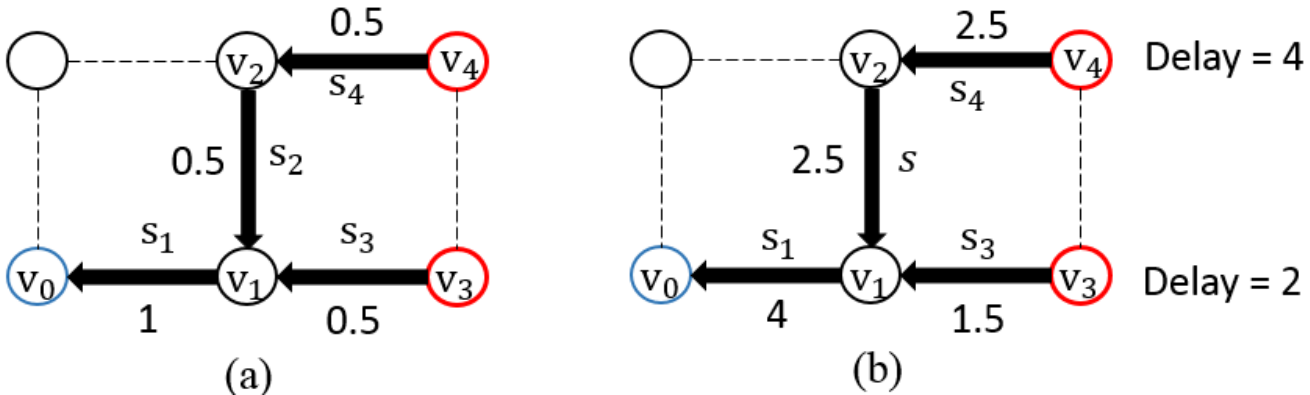
# Second Stage: NDLA

- When the layer assignment of all illegal nets is finished
  - Check wire congestion constraints
  - If wire congestion constraints are not both satisfied
    - Increase the congestion cost of each 3D grid edge with unnecessary overflow
    - Go to the next iteration



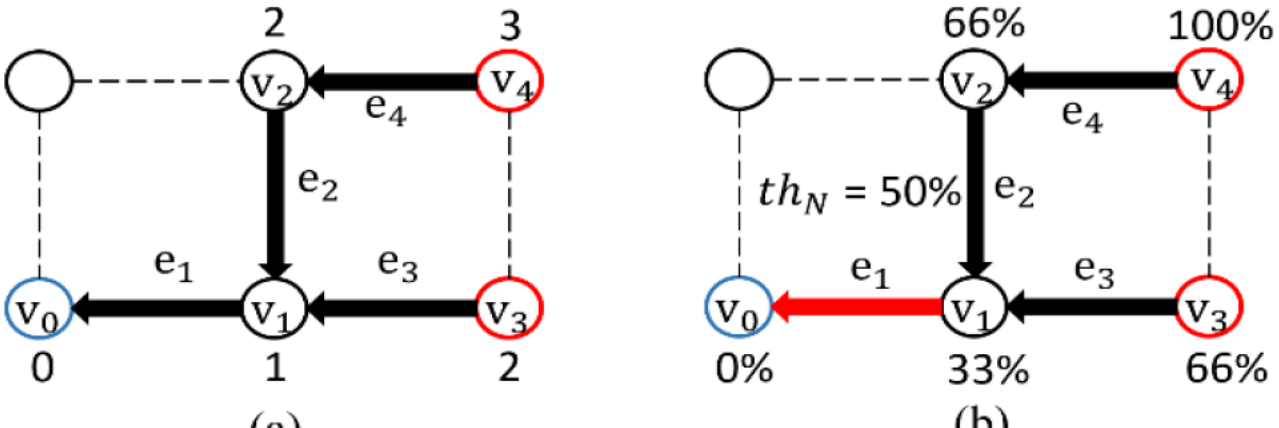
# Segment Weighting

- The delay cost of each segment of a net is computed by the multiplication of its weight and its delay during layer assignment
- The weight of each wire segment *s* of a net is re-calculated at the beginning of each iteration of NDLA
  - $w^0{}_s$ : The sum of the weights of the downstream sinks of *s*.
  - $w_s^{i+1} = w_s^i + \sum_{j=1}^{r} \delta \times [d(si_j)/m]$ , $i$: iteration count
- Timing-critical wire segments are assigned higher weights
- A wire segment with a higher weight has more impact on the delays of its downstream sinks



# Control on Using Parallel Wires and NDR Wires

- Wire segments near the source of a net
  - Have greater influence on the net delay
  - Are better candidates for using parallel wires or NDR wires
- A method to control the usage of parallel wires and NDR wires
  - Each node of a net has a level ratio (the level of the node divided by the maximum level of all nodes of the net)
  - A level-ratio threshold between 0 and 1 is decided for each net
  - For each node, if its level ratio exceeds the threshold, the 2D edge connecting the node and its parent can only be assigned to a default-width wire



# Cost Function

- The objective function takes the wire congestion constraints into account for each illegal net *N*

$$\beta \times d(N) + \#via_N + \sum_{s \in N} cong(e_s)$$

- Congestion cost of a 3D grid edge *e*

$$cong(e) = p_e \times h_e$$

- $p_e$ is the current overflow penalty of *e*

$$p_e = \max\{0, a \times (u(e) - cap(e))\}$$

- History cost of *e* at the (*i*+1)-th iteration

$$h_e^{i+1} = \begin{cases} h_e^i + \rho \times 2^i & \text{, if e has overflow} \\ h_e^i & \text{, otherwise} \end{cases}$$

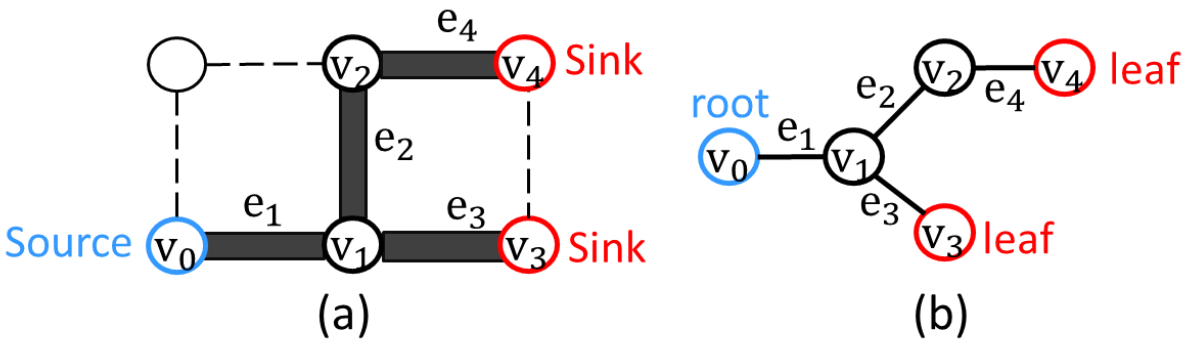
# Third Stage: PO

- Rip up and reassign each net once according to a decreasing order of their delays
  - Always satisfy the wire congestion constraints

- The congestion cost of a 3D grid edge is set to a very large value, if assigning to this edge causes unnecessary overflow

# Single Net Layer Assignment

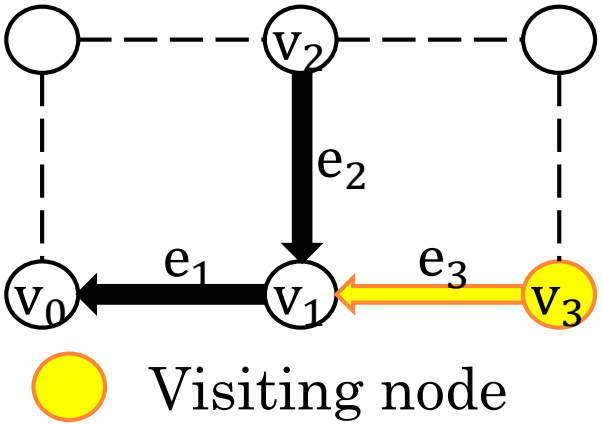
- A dynamic programing based algorithm
  - Regard a routed net in the 2D grid graph as a tree.




  - Visit each node in a bottom-up manner so as to tentatively assign each tree edge to corresponding 3D grid edges.
  - Then find a least-cost layer assignment result in a top-down manner.

# Single Net Layer Assignment

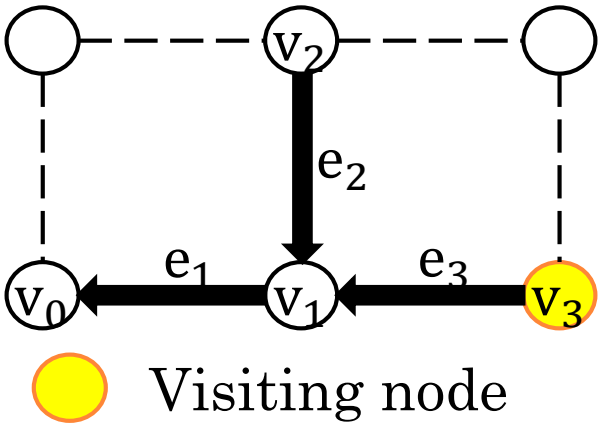
- How to determine the wire types on each layer for a tree edge?

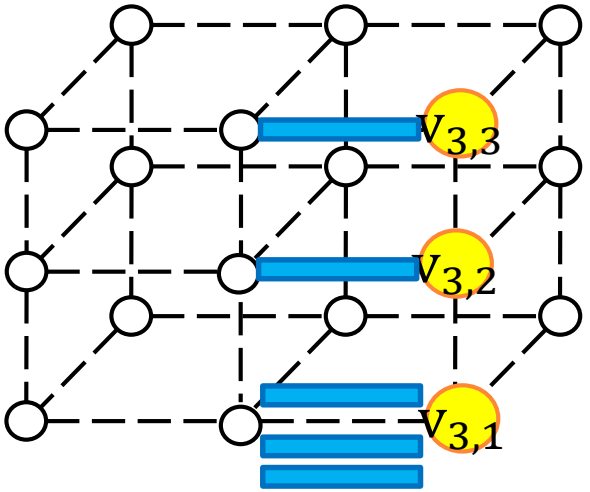



- Extra capacity usage
  - ✓ Default-width wire: 0
  - ✓ Parallel wires: 1
  - ✓ NDR wire: 2

- If the remaining capacity of $e_3$ is more than or equal to the extra capacity usage of wire type *t*, the wire type *t* can be used.
  - ✓ Initial remaining capacity: max(0, capacity – demand)
  - ✓ The remaining capacity will be updated according to the layer assignment result of the net

# Single Net Layer Assignment

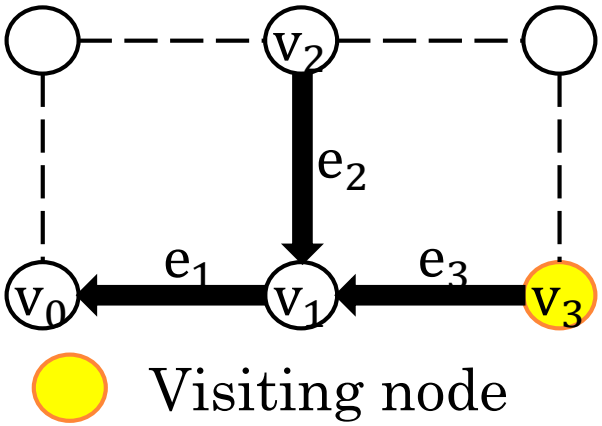


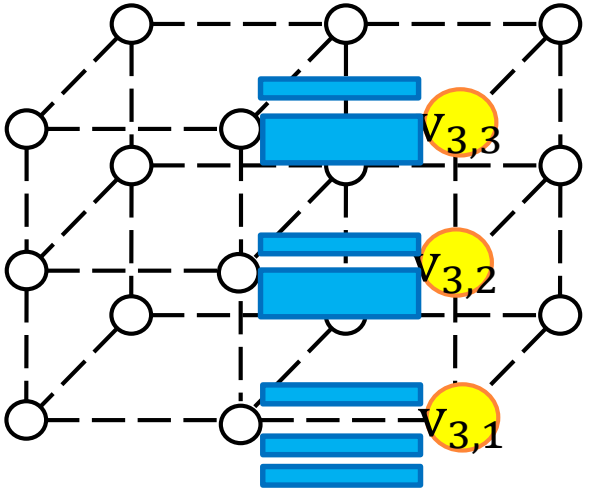




Default width
Layer 1: 1*W*
Layer 2: 2*W*
Layer 3: 4*W*

Assume remaining capacity of $e_3$ is 1

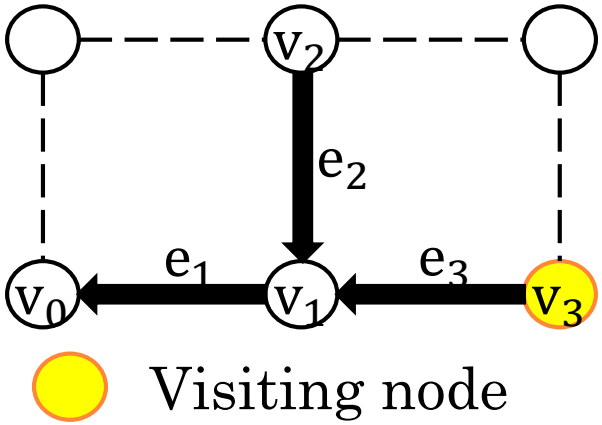
# Single Net Layer Assignment



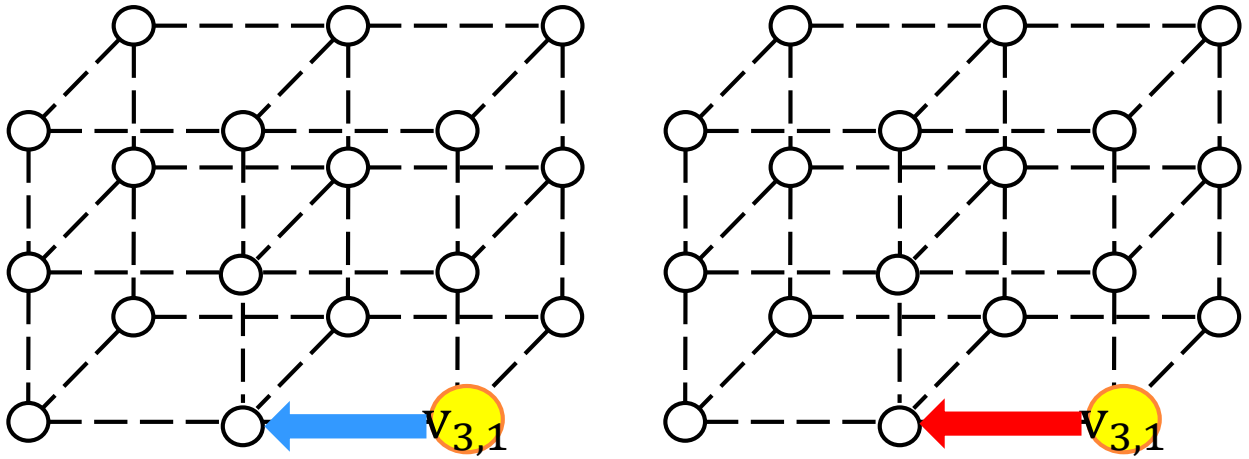




Assume remaining capacity of $e_3$ is 3

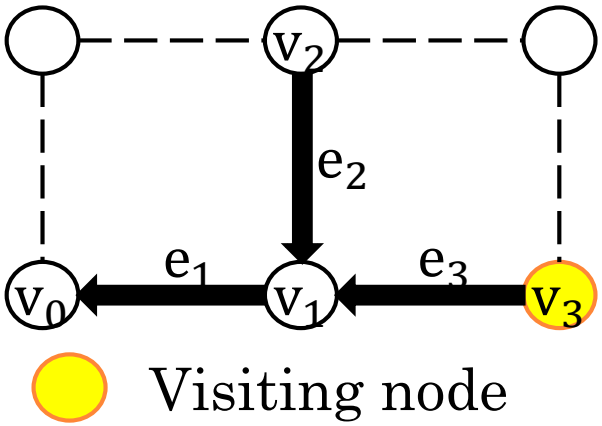
# How to Process a Leaf Node

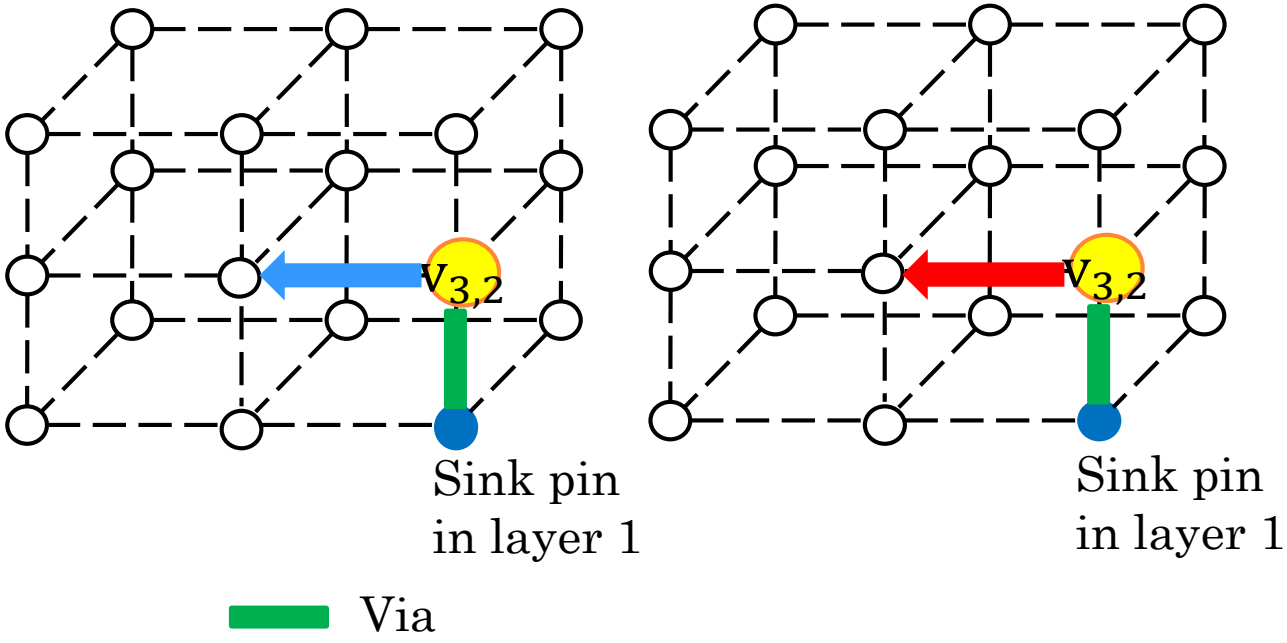



Assume $e_3$ can also use two parallel wires in layer 1, and an NDR wire in layer 2 or 3.

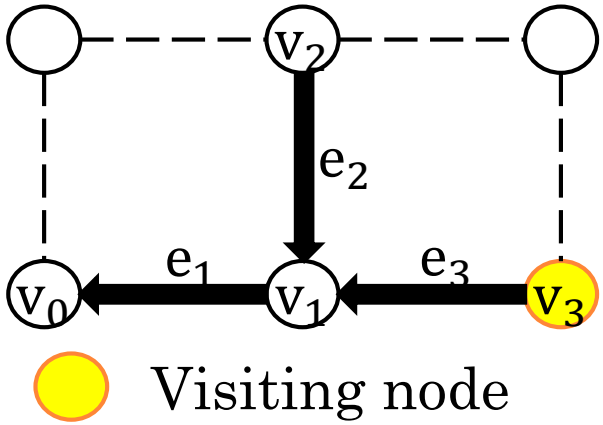


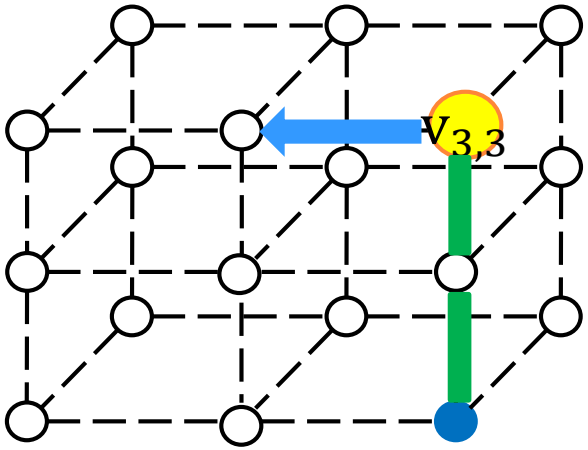
# How to Process a Leaf Node

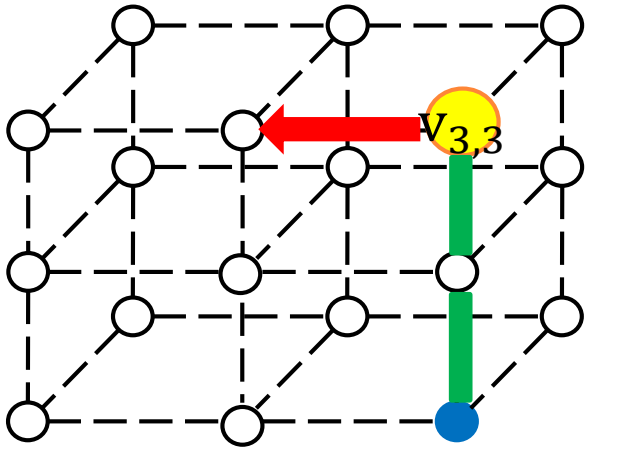


# How to Process a Leaf Node

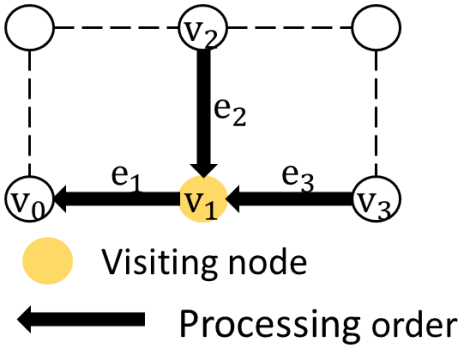


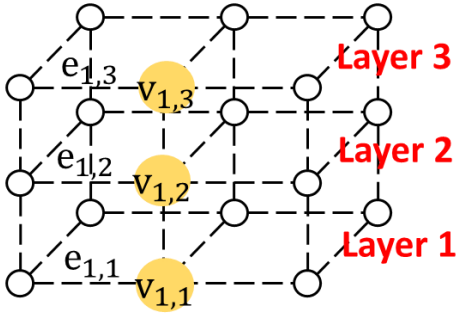



# How to Process an Internal Node

- Consider the scenario where edge $e_1$ is being considered for layer 2 with a default-width wire.
- The layer assignment result of the subtree rooted at $v_2$ ($v_3$) with respect to each eligible layer and wire type for $e_2$ ($e_3$) has been generated.
- Assume
  - $e_2$ can only be assigned to layer 1 , 2, or 3 with a default-width wire but no parallel wires or NDR wire.
  - $e_3$ can use a default-width wire or two parallel wires in layer 1, as well as a default-width wire or an NDR wire in layer 2 or 3.



(a)

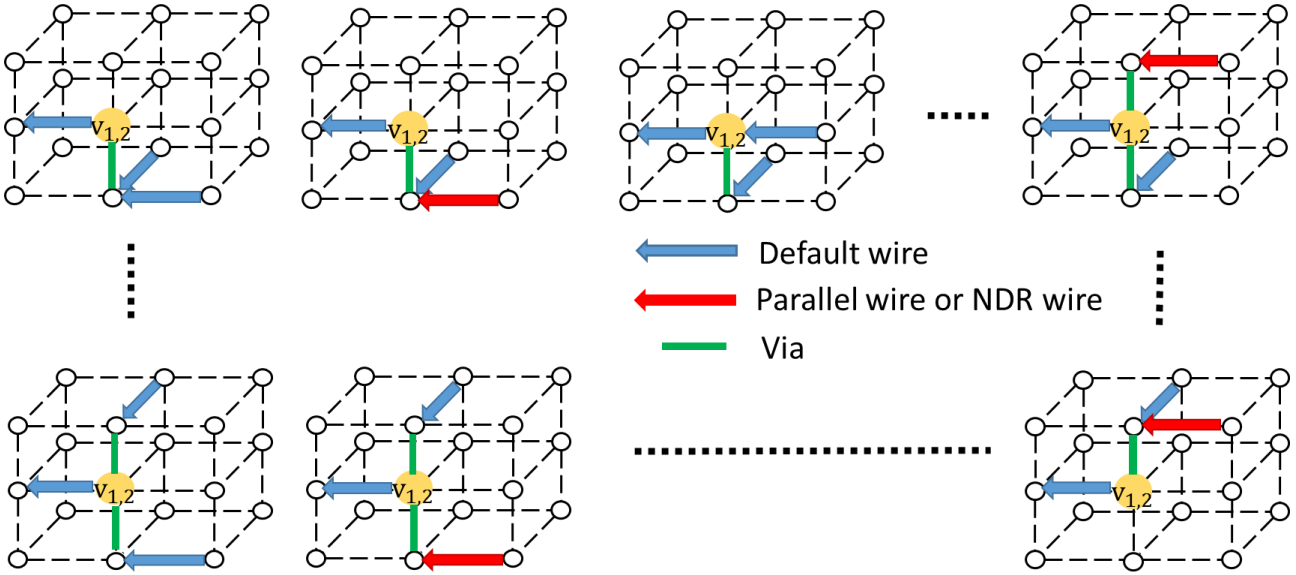


(b)

Default width
Layer 1 : $1W$
Layer 2 : $2W$
Layer 3 : $4W$

# How to Process an Internal Node

- There are 3 layer assignment results for $e_2$ and 6 layer assignment results for $e_3$.
- 18 layer assignment results for the subtree rooted at $v_1$ will be generated.
- The one with the least cost is selected to be the layer assignment result for the subtree rooted at $v_1$ when $e_1$ is assigned to layer 2 using a default-width wire.




(c)

# How to Process the Root

- Enumerate all possible combinations of a layer assignment result from each child node.
- Find the one with the least cost.
- Traverse the least-cost layer assignment result in a top-down manner so as to assign each tree edge to a layer.

# Experimental Setup

- Machine: 2.0 GHz Intel Xeon CPU and 96 GB memory
- Test cases: DAC12 routability-driven placement benchmarks
  - Placement results are obtained by NTUplace4
  - 2D global routing results
    - Compress from the 3D global routing results of NCTU-GR 2.0
    - All have zero total wire overflow

# Delay Impact from Coupling Capacitance

- With and without considering coupling effect

| Benchmark | WCE | | | | WOCE | | | |
|---|---|---|---|---|---|---|---|---|
| | TD | MD | #vc | runtime | TD | MD | #vc | runtime |
| superblue11 | 8.73E+05 | 2186.1 | 5.46E+06 | 421.6 | 1.02E+06 | 2703.2 | 5.13E+06 | 396.6 |
| superblue12 | 6.22E+05 | 1268.7 | 8.18E+06 | 607.0 | 7.24E+05 | 1185.6 | 7.75E+06 | 613.2 |
| superblue14 | 4.82E+05 | 399.7 | 3.92E+06 | 315.1 | 5.53E+05 | 617.3 | 3.70E+06 | 303.5 |
| superblue16 | 6.42E+05 | 357.4 | 3.90E+06 | 374.6 | 7.59E+05 | 566.0 | 3.61E+06 | 347.1 |
| superblue19 | 2.15E+05 | 604.6 | 2.96E+06 | 222.8 | 2.52E+05 | 799.3 | 2.79E+06 | 209.8 |
| superblue2 | 2.58E+06 | 882.8 | 6.80E+06 | 906.3 | 3.08E+06 | 1147.0 | 6.26E+06 | 839.0 |
| superblue3 | 9.69E+05 | 949.5 | 6.09E+06 | 595.9 | 1.09E+06 | 965.7 | 5.61E+06 | 567.3 |
| superblue6 | 7.76E+05 | 428.7 | 5.99E+06 | 503.1 | 8.86E+05 | 619.7 | 5.61E+06 | 489.9 |
| superblue7 | 7.26E+05 | 347.5 | 9.01E+06 | 640.0 | 8.29E+05 | 744.3 | 8.40E+06 | 638.4 |
| superblue9 | 4.82E+05 | 280.8 | 4.93E+06 | 389.7 | 5.64E+05 | 441.7 | 4.60E+06 | 379.3 |
| average | 8.37E+05 | 770.6 | 5.72E+06 | 497.6 | 9.75E+05 | 979.0 | 5.35E+06 | 478.4 |
| ratio | 1.00 | 1.00 | 1.00 | 1.00 | 1.15 | 1.40 | 0.93 | 0.96 |

# Effectiveness of Segment Weighting

- Coupling effect
- With and without segment weighting

| Benchmark | WCE | | | | | | | WCE_NW | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TD | MD | 0.5% | 1% | 5% | #vc | runtime | TD | MD | 0.5% | 1% | 5% | #vc | runtime |
| superblue11 | 8.73E+05 | 2186.1 | 143.2 | 86.4 | 23.7 | 5.46E+06 | 421.6 | 7.90E+05 | 2381.8 | 117.4 | 72.8 | 21.0 | 5.57E+06 | 527.7 |
| superblue12 | 6.22E+05 | 1268.7 | 55.7 | 38.9 | 11.9 | 8.18E+06 | 607.0 | 5.44E+05 | 966.0 | 40.2 | 29.8 | 10.2 | 8.49E+06 | 751.2 |
| superblue14 | 4.82E+05 | 399.7 | 86.5 | 59.4 | 18.9 | 3.92E+06 | 315.1 | 4.23E+05 | 319.9 | 58.9 | 44.1 | 16.0 | 4.03E+06 | 403.5 |
| superblue16 | 6.42E+05 | 357.4 | 84.4 | 63.0 | 23.7 | 3.90E+06 | 374.6 | 5.49E+05 | 179.8 | 54.6 | 43.3 | 19.3 | 4.05E+06 | 500.1 |
| superblue19 | 2.15E+05 | 604.6 | 34.3 | 25.1 | 10.2 | 2.96E+06 | 222.8 | 2.01E+05 | 496.4 | 26.7 | 20.5 | 9.2 | 3.04E+06 | 251.3 |
| superblue2 | 2.58E+06 | 882.8 | 205.8 | 162.4 | 63.8 | 6.80E+06 | 906.3 | 2.32E+06 | 599.8 | 155.6 | 124.0 | 54.6 | 7.06E+06 | 1181.3 |
| superblue3 | 9.69E+05 | 949.5 | 143.9 | 102.2 | 30.8 | 6.09E+06 | 595.9 | 8.01E+05 | 781.6 | 95.3 | 70.0 | 24.3 | 6.36E+06 | 757.2 |
| superblue6 | 7.76E+05 | 428.7 | 89.1 | 62.8 | 20.9 | 5.99E+06 | 503.1 | 6.75E+05 | 344.7 | 59.8 | 45.9 | 17.6 | 6.18E+06 | 652.2 |
| superblue7 | 7.26E+05 | 347.5 | 62.1 | 42.3 | 13.3 | 9.01E+06 | 640.0 | 6.41E+05 | 256.6 | 45.1 | 32.7 | 11.3 | 9.34E+06 | 791.1 |
| superblue9 | 4.82E+05 | 280.8 | 67.2 | 46.7 | 16.0 | 4.93E+06 | 389.7 | 4.29E+05 | 267.1 | 49.1 | 36.3 | 13.8 | 5.12E+06 | 495.5 |
| average | 8.37E+05 | 770.6 | 97.2 | 68.9 | 23.3 | 5.72E+06 | 497.6 | 7.37E+05 | 659.4 | 70.3 | 51.9 | 19.7 | 5.92E+06 | 631.1 |
| ratio | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 0.88 | 0.80 | 0.72 | 0.76 | 0.83 | 1.03 | 1.26 |

# Effectiveness of Parallel and NDR Wires

- Coupling effect + segment weighting
- With and without parallel and NDR wires

| WCE_NW | | | | | | | WCE_NW_P | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TD | MD | 0.5% | 1% | 5% | #vc | runtime | TD | MD | 0.5% | 1% | 5% | #vc | runtime |
| 7.90E+05 | 2381.8 | 117.4 | 72.8 | 21.0 | 5.57E+06 | 527.7 | 7.51E+05 | 1602.3 | 109.7 | 68.2 | 19.8 | 5.59E+06 | 606.3 |
| 5.44E+05 | 966.0 | 40.2 | 29.8 | 10.2 | 8.49E+06 | 751.2 | 5.43E+05 | 581.5 | 40.2 | 29.9 | 10.1 | 8.50E+06 | 933.2 |
| 4.23E+05 | 319.9 | 58.9 | 44.1 | 16.0 | 4.03E+06 | 403.5 | 4.08E+05 | 205.0 | 56.0 | 41.9 | 15.2 | 4.04E+06 | 470.6 |
| 5.49E+05 | 179.8 | 54.6 | 43.3 | 19.3 | 4.05E+06 | 500.1 | 5.44E+05 | 186.3 | 54.4 | 42.9 | 18.9 | 4.12E+06 | 596.2 |
| 2.01E+05 | 496.4 | 26.7 | 20.5 | 9.2 | 3.04E+06 | 251.3 | 1.94E+05 | 304.5 | 24.6 | 19.0 | 8.7 | 3.04E+06 | 317.3 |
| 2.32E+06 | 599.8 | 155.6 | 124.0 | 54.6 | 7.06E+06 | 1181.3 | 2.22E+06 | 503.9 | 145.0 | 116.2 | 51.5 | 7.13E+06 | 1324.5 |
| 8.01E+05 | 781.6 | 95.3 | 70.0 | 24.3 | 6.36E+06 | 757.2 | 7.75E+05 | 535.1 | 89.1 | 66.2 | 23.2 | 6.38E+06 | 886.7 |
| 6.75E+05 | 344.7 | 59.8 | 45.9 | 17.6 | 6.18E+06 | 652.2 | 6.55E+05 | 253.7 | 57.3 | 44.0 | 16.9 | 6.20E+06 | 745.9 |
| 6.41E+05 | 256.6 | 45.1 | 32.7 | 11.3 | 9.34E+06 | 791.1 | 6.25E+05 | 233.2 | 43.1 | 31.4 | 10.9 | 9.34E+06 | 950.6 |
| 4.29E+05 | 267.1 | 49.1 | 36.3 | 13.8 | 5.12E+06 | 495.5 | 4.14E+05 | 211.5 | 46.0 | 34.2 | 13.1 | 5.12E+06 | 581.1 |
| 7.37E+05 | 659.4 | 70.3 | 51.9 | 19.7 | 5.92E+06 | 631.1 | 7.11E+05 | 487.2 | 63.2 | 45.8 | 17.9 | 1.42E+07 | 1589.9 |
| 0.88 | 0.80 | 0.72 | 0.76 | 0.83 | 1.03 | 1.26 | 0.86 | 0.59 | 0.68 | 0.72 | 0.81 | 1.04 | 1.49 |

# Conclusion

- We present a delay-driven layer assignment algorithm that considers the coupling effect and uses parallel wires and NDR wires to pursue better delay reduction.

- The experimental results show that our algorithm can effectively use routing resources to significantly reduce the net delays.