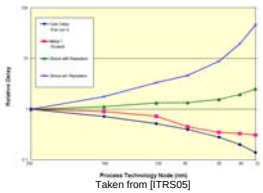
Sidewinder: A Scalable ILP-Based Wire Router

Jin Hu, Jarrod A. Roy, and Prof. Igor L. Markov

Motivation

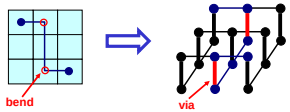
Minimize Wirelength



Taken from [ITRS05]

- Less power consumption
- Smaller wire delay

Minimize Bends

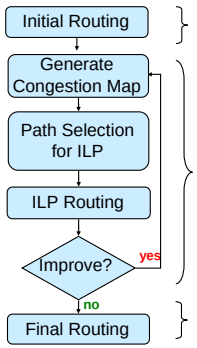


- Directional preference on layer
- Bends require change layers → vias
- Tungsten vs. copper or aluminum
- More resistant, larger area, less reliable

Why Integer Linear Programming?

- Finding an optimal routing solution for all nets is **NP-Complete**
  - ILP considers all variables **simultaneously** instead of one at a time
- Historically **unscalable** but have improved
- ILP solver technology: CPLEX v9, v10, v11
  - More powerful CPUs
- **Significant progress in ILP formulations**

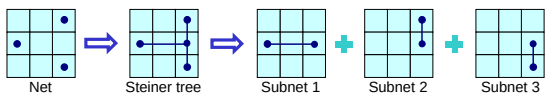
Sidewinder Flow



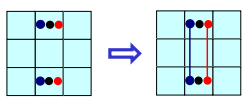
- Breaks up multi-pin nets into subnets
- Routes with *L*-shape initially
- Calculates total congestion
- Chooses best **2** options for each net (not limited to *L*)
- Generates new solution
  → **New solution is never worse than previous**
- Repeats until no improvement
- Routes remaining nets with maze router
- Allows overflow

Initial Routing

- **Decompose all multi-pin nets into 2-pin subnets**



- **Route all subnets using only *L* shapes**
- **Some nets can be unrouted**

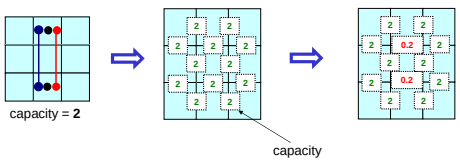


Routed Nets: Blue, Red

Unrouted Nets: Black

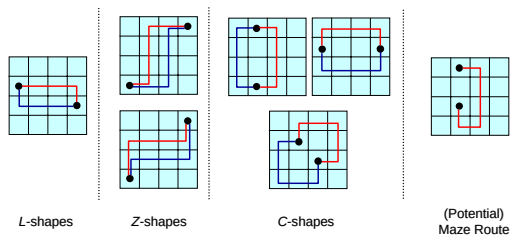
Generate Congestion Map

- For each routed net, subtract **90%** of usage from total capacities
  - Every net has **2** choices
  - 1 from current solution, 1 from congestion map
  - Higher priority given to current solution



Path Selection

- Each path is selected based on congestion map
  - For each unrouted net, select **2** possible paths
  - For each routed net, select **1** possible path



ILP Formulation

$$Maximize: \sum_{n}(w_{1n}x_{1n} + w_{2n}x_{2n})$$ — Maximizes total number of nets routed

$$Subject\ to:$$

$$x_{1n} + x_{2n} \le 1 \quad \forall n$$
$$x_{1n}, x_{2n} \in \{0,1\} \quad \forall n$$ — Ensures at most **1** path is selected

$$\sum_{x_{1n},x_{2n} \in g(i,j)\to g(i,j+1)} (x_{1n} + x_{2n}) \le c_{g(i,j)\to g(i,j+1)} \quad g(i,j) \in G, 0 \le i < X, \quad 0 \le j < Y-1$$ — Capacity constraint in **N** direction

$$\sum_{x_{1n},x_{2n} \in g(i,j)\to g(i,j-1)} (x_{1n} + x_{2n}) \le c_{g(i,j)\to g(i,j-1)} \quad g(i,j) \in G, 0 \le i < X, \quad 0 < j < Y$$ — Capacity constraint in **S** direction

$$\sum_{x_{1n},x_{2n} \in g(i,j)\to g(i+1,j)} (x_{1n} + x_{2n}) \le c_{g(i,j)\to g(i+1,j)} \quad g(i,j) \in G, 0 \le i < X-1, 0 \le j < Y$$ — Capacity constraint in **E** direction

$$\sum_{x_{1n},x_{2n} \in g(i,j)\to g(i-1,j)} (x_{1n} + x_{2n}) \le c_{g(i,j)\to g(i-1,j)} \quad g(i,j) \in G, 0 < i < X, \quad 0 \le j < Y$$ — Capacity constraint in **W** direction

Where:

x_{1n}, x_{2n} : the 2 paths for each net $n \in$ netlist N

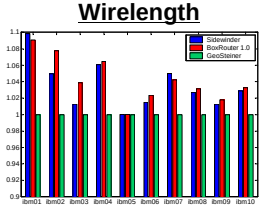
w_{1n}, w_{2n} : the net weights for x_{1n}, x_{2n}

$g(i,j)$: the grid cell in G of size X x Y

$c_{g(i,j)\to g(i\pm1,j\pm1)}$: the 4 capacity constraints for each $g(i,j)$

Results

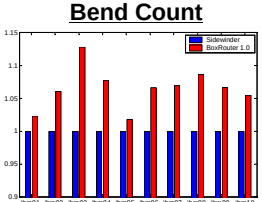
Wirelength



- Normalized to GeoSteiner: Steiner tree generator
- **Best lower bound for WL**
- No capacity restrictions (solutions illegal)
- Impractically slow

On average, 0.5% less WL

Bend Count

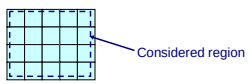


- Normalized to minimum of both routers
- Significantly fewer bends
- Vast majority of routes are patterns – *L*, *Z*, *C*
- Better bend count on every benchmark

On average, 6.4% fewer bends

Conclusions

- Global ILP – considers entire region



- Uses dynamically updated congestion map
- Uses flexible and economical route shapes: *L*, *Z*, *C*
- Produces better solution quality
  - Average of 0.5% less wirelength
  - Average of 6.4% fewer vias