# A Faster Approximation Scheme for Timing Driven Minimum Cost Layer Assignment

Shiyan Hu*, Zhuo Li**, and Charles J. Alpert**
*Dept of ECE, Michigan Technological University
**IBM Austin Research Lab

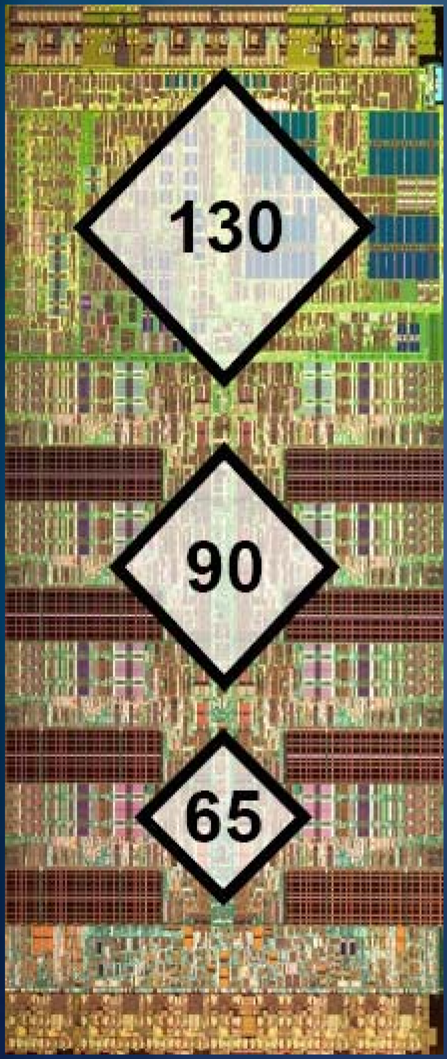
Michigan Tech
Create the Future

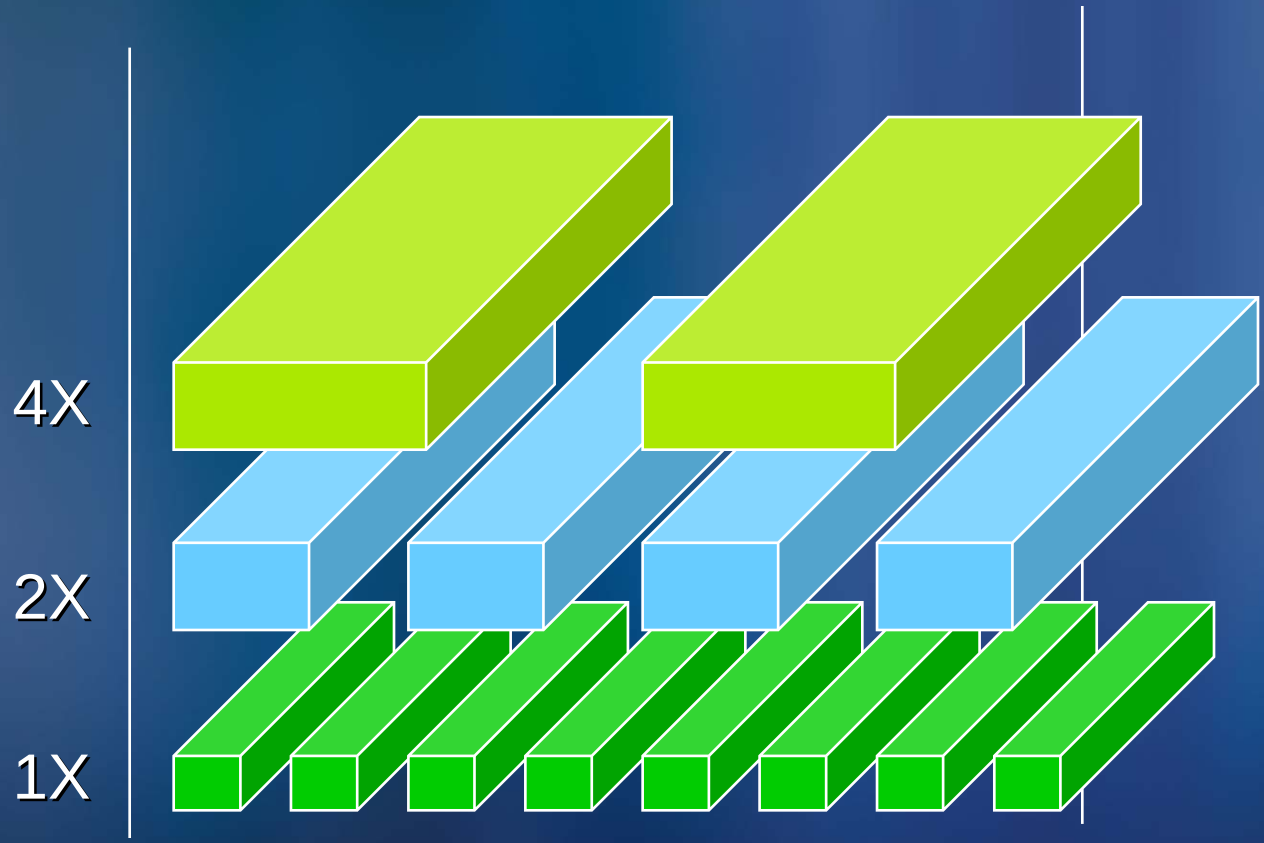
IBM

# Outline

- Introduction
- Problem Formulation
- The Algorithm
  - Linear time dynamic programming
  - Bound independent oracle search
- Experimental Results
- Conclusion

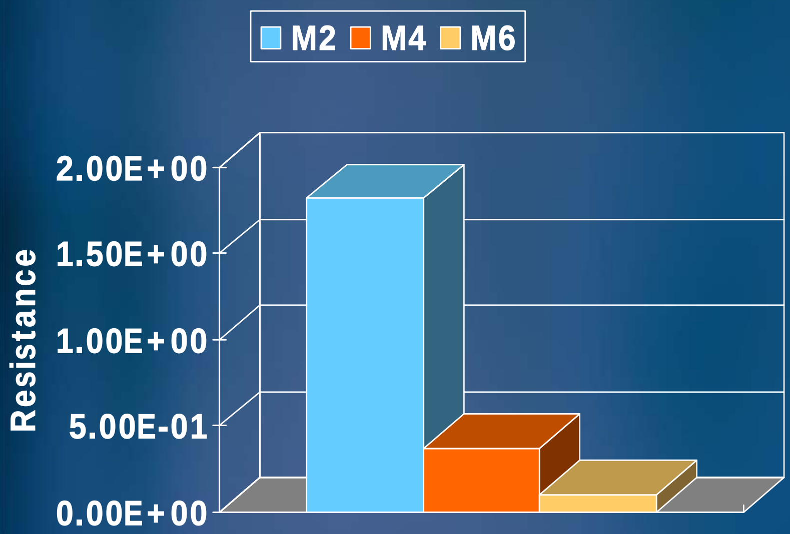
# Layer Assignment

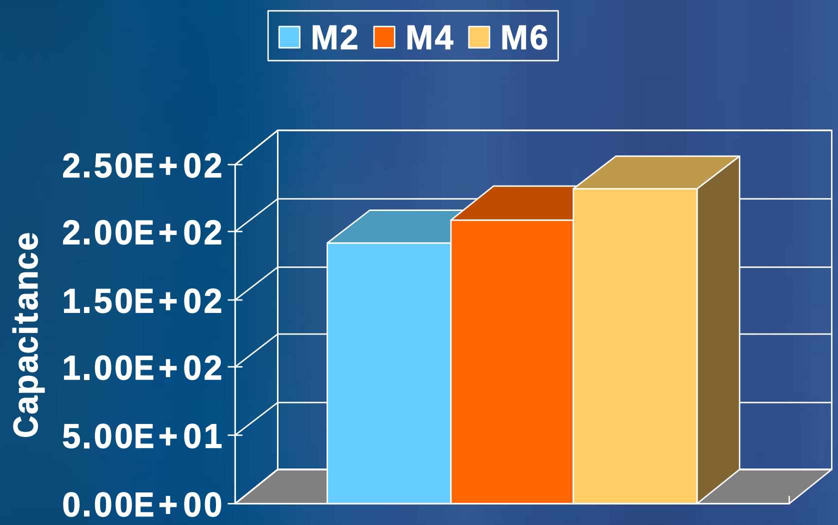


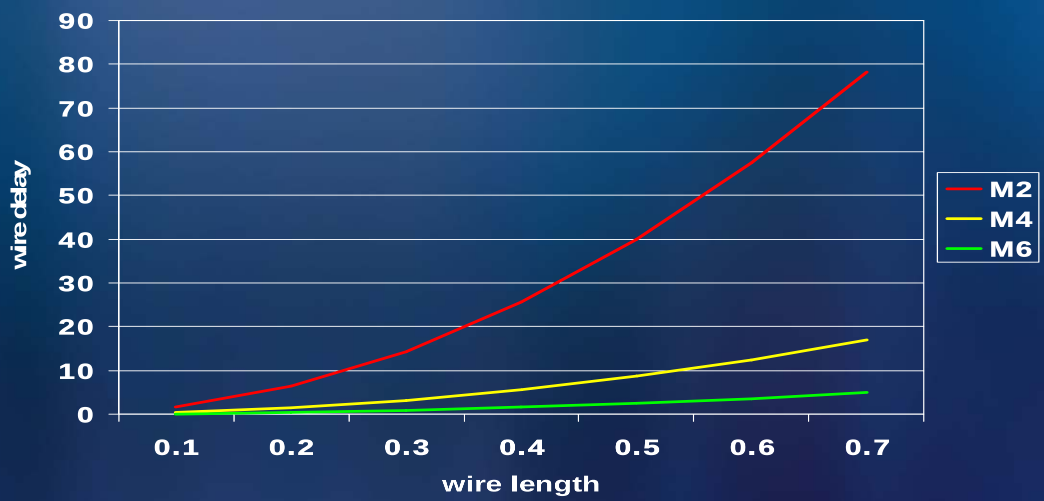




- In 45nm technology, layer assignment is critical for timing and buffer area optimization

# Wire RC and Delay



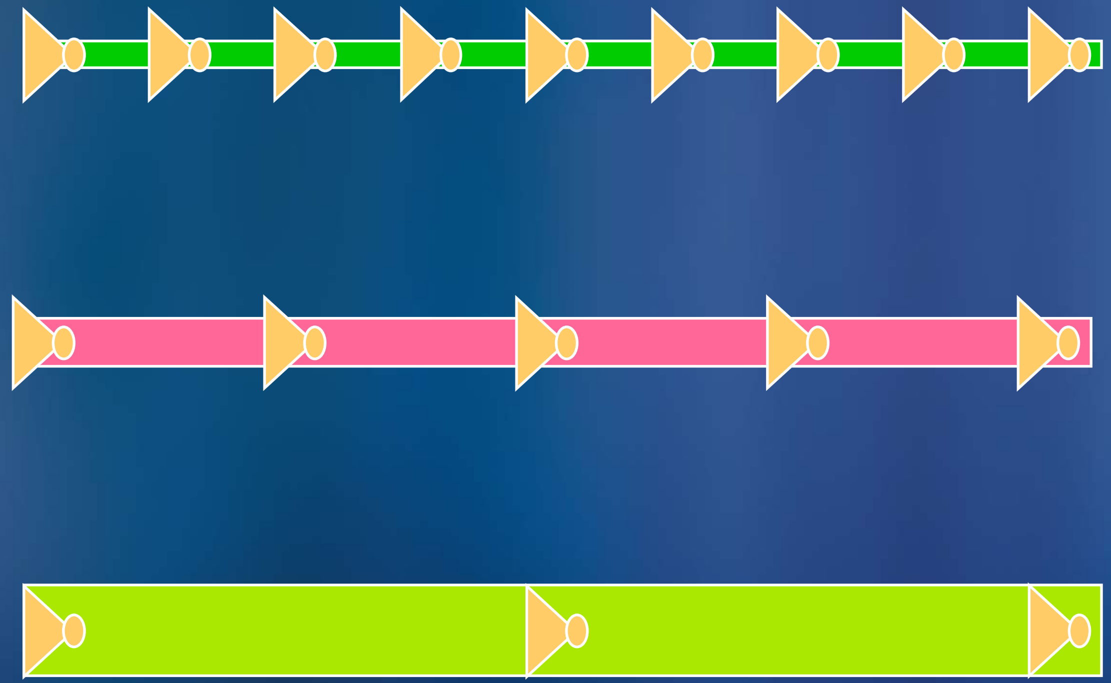



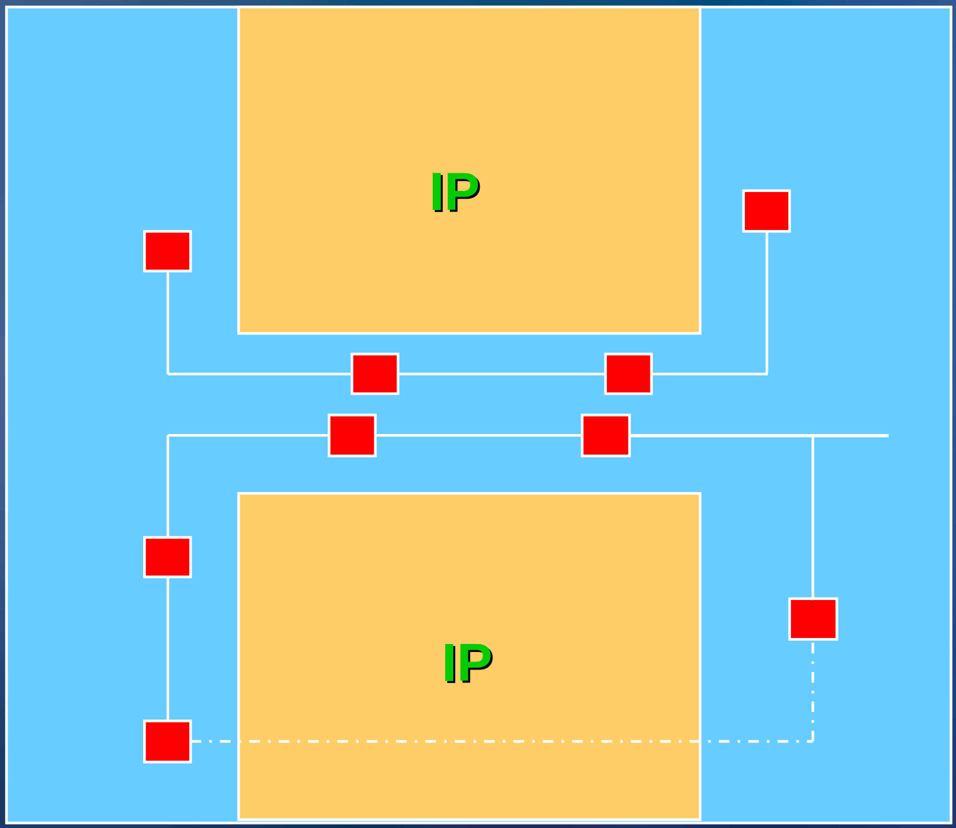

Wire in higher layer has much smaller delay

# Impact to Buffering

- A buffer can drive longer distance in higher layer
  - Timing is improved
  - Fewer buffers are needed



# Impact to Routing/Buffering

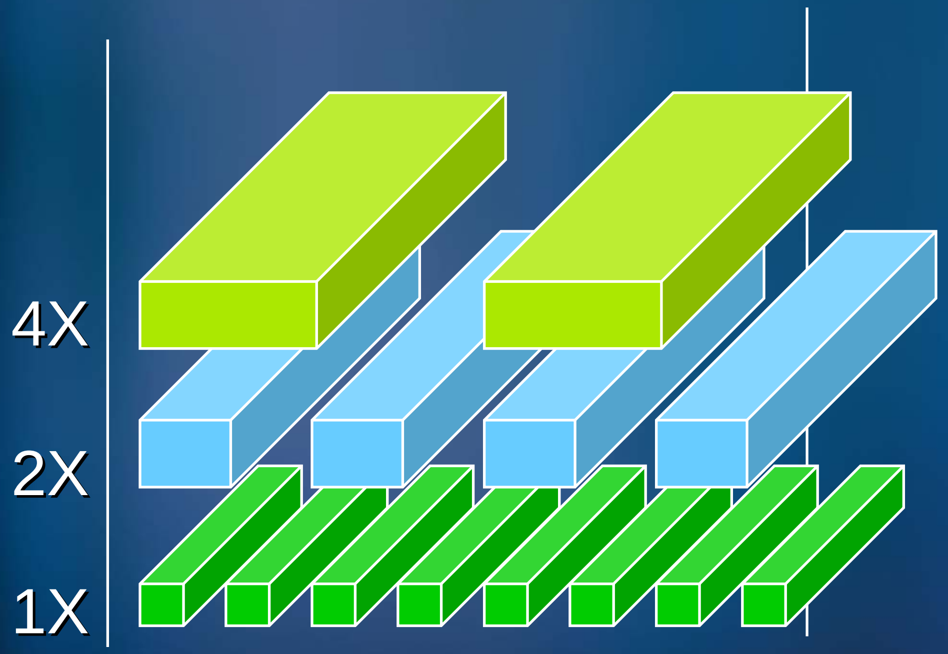


# Problem Formulation

Can be different layers

- A layer refers to a pair of horizontal and vertical layers with similar RC characteristics
- Between any buffers, one layer is used
- In early design stage, when buffering effect is considered, wire shaping is not important [Alpert TCAD'01]
- In post-routing stage, wire shaping could improve timing, reduce vias and reduce coupling and so forth

# Fully Polynomial Time Approximation Scheme (FPTAS)




- A Fully Polynomial Time Approximation Scheme
  - Provably good
  - Within $(1+\varepsilon)$ optimal cost for any $\varepsilon>0$
  - Runs in time polynomial in n (segments), m (layers) and $1/\varepsilon$
  - Ultimate solution for an NP-hard problem in theory
  - Highly practical

# Previous Work in ICCAD'08

- It depends on M and uses a DP of $O(mn^3/\varepsilon^2)$ time

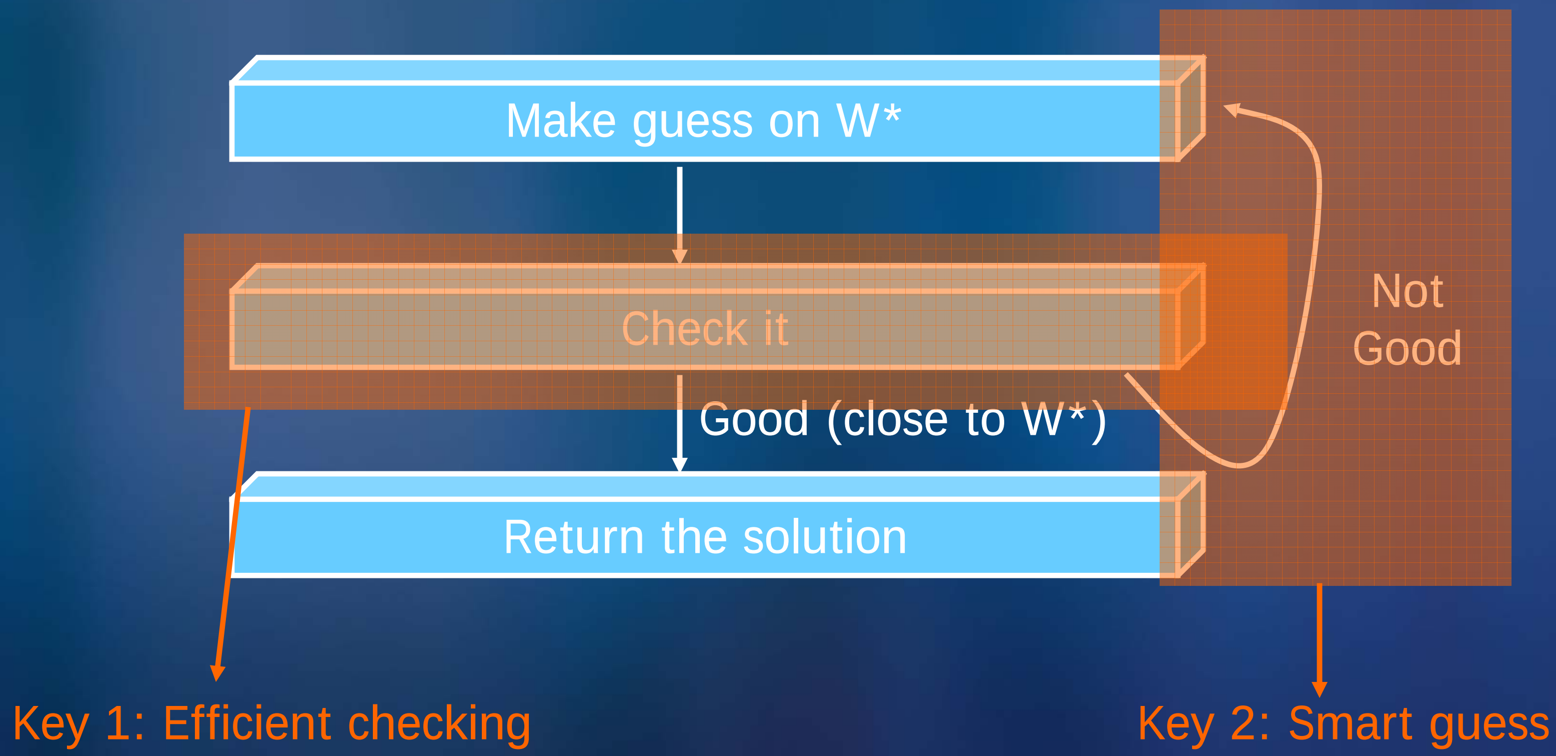
Ratio between upper and lower bounds of the cost of optimal layer assignment → Bound independent oracle query

An iterative DP with incremental W → Our DP needs one run for all W

- New FPTAS runs in $O(mn^2/\varepsilon)$ time

# The Rough Picture

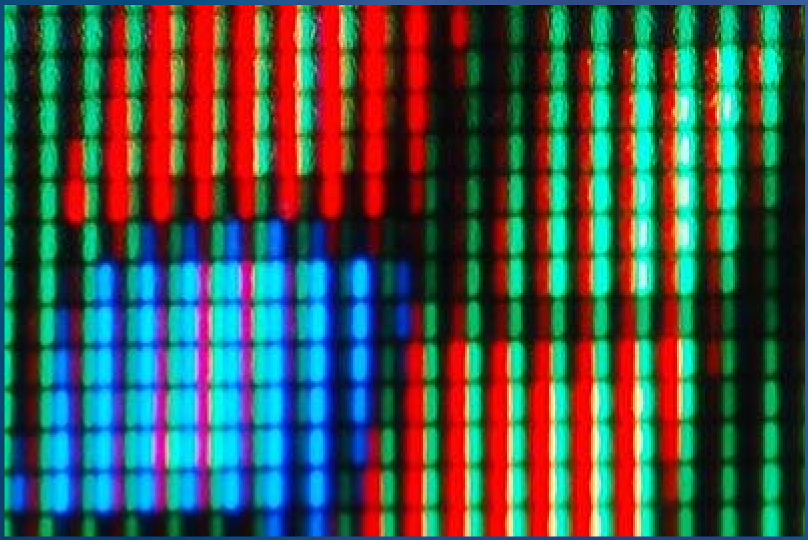
W*: the cost of optimal solution



# Key 1: Efficient Checking

**Benefit of guess**

- Only maintain the solutions with cost no greater than the guessed cost
- Accelerate DP



# The Oracle

- Oracle (x … r x>W* or not
  - Without knowing W*
  - Answer efficiently

Setup upper and lower bounds of cost W*

Guess x within the bounds

Oracle (x)

Update the bounds

# Construction of Oracle(x)

Dynamic Programming

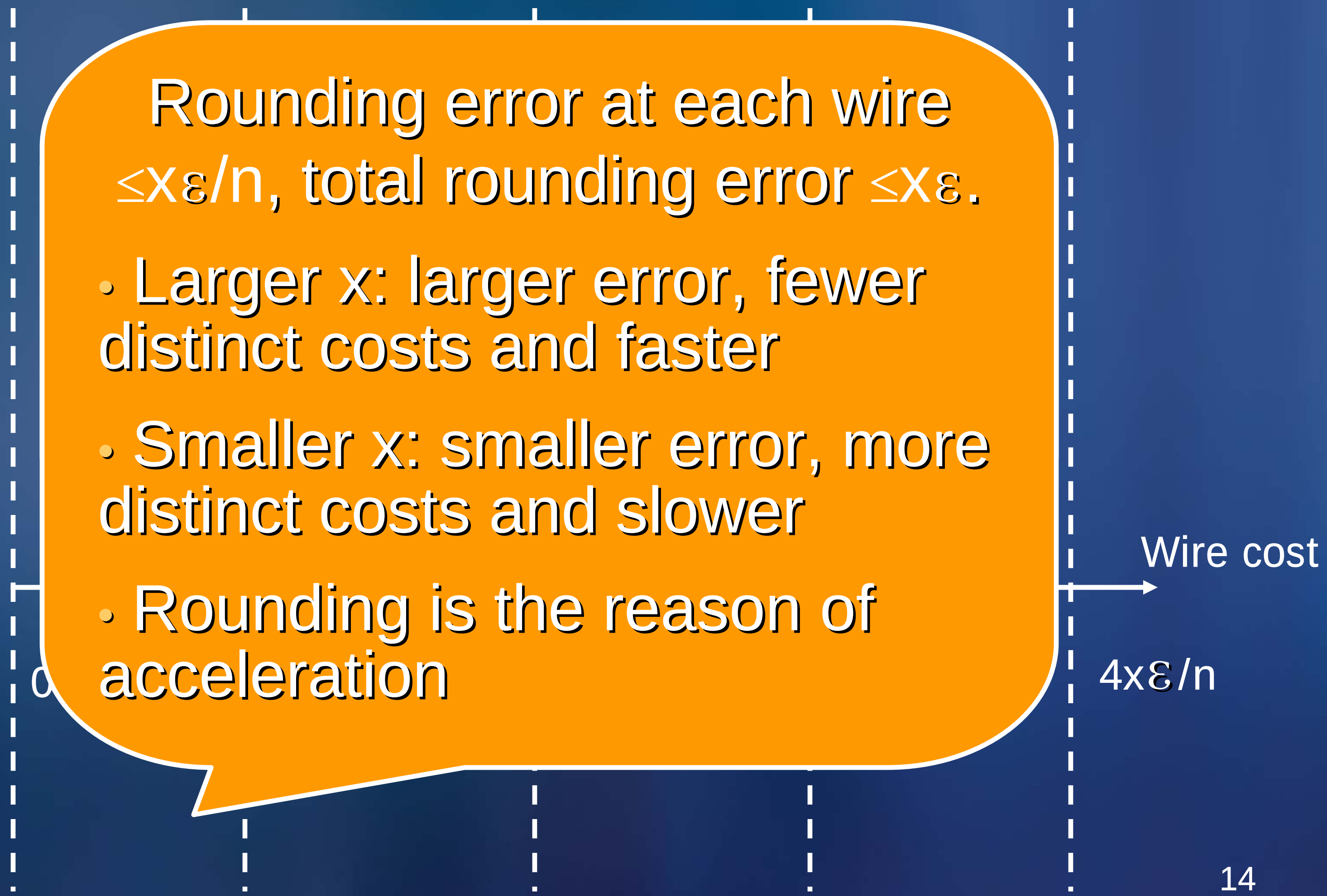
Only interested in whether there is a solution with cost up to x satisfying timing constraint

Scale and round each wire cost

$$w = \left\lfloor \frac{w}{x\varepsilon / n} \right\rfloor$$

Perform DP to scaled problem with cost bound n/ε. Time polynomial in n/ε

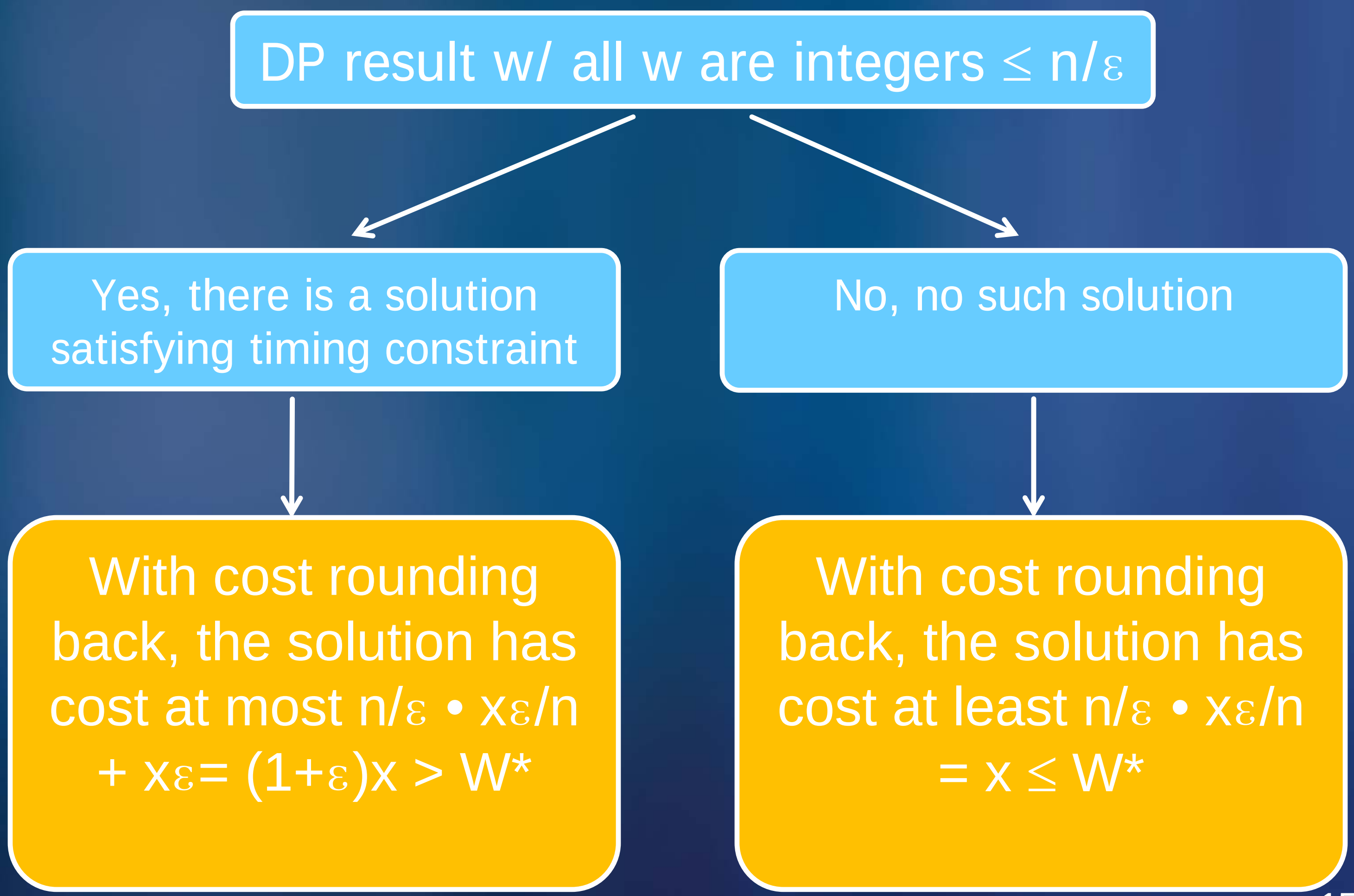
# Scaling and Rounding




Rounding error at each wire $\leq x\varepsilon/n$, total rounding error $\leq x\varepsilon$.

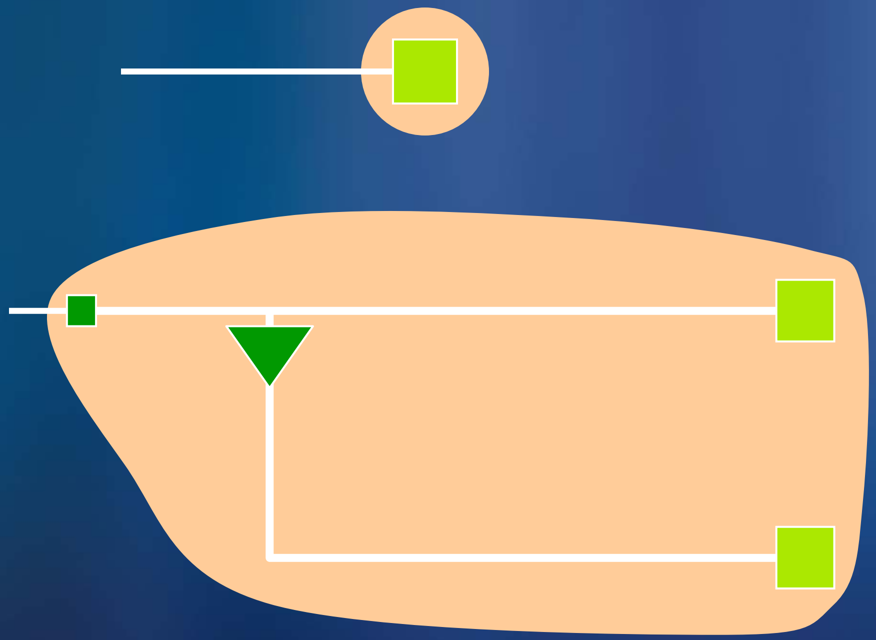
- Larger x: larger error, fewer distinct costs and faster
- Smaller x: smaller error, more distinct costs and slower
- Rounding is the reason of acceleration

# Dynamic Programming Results

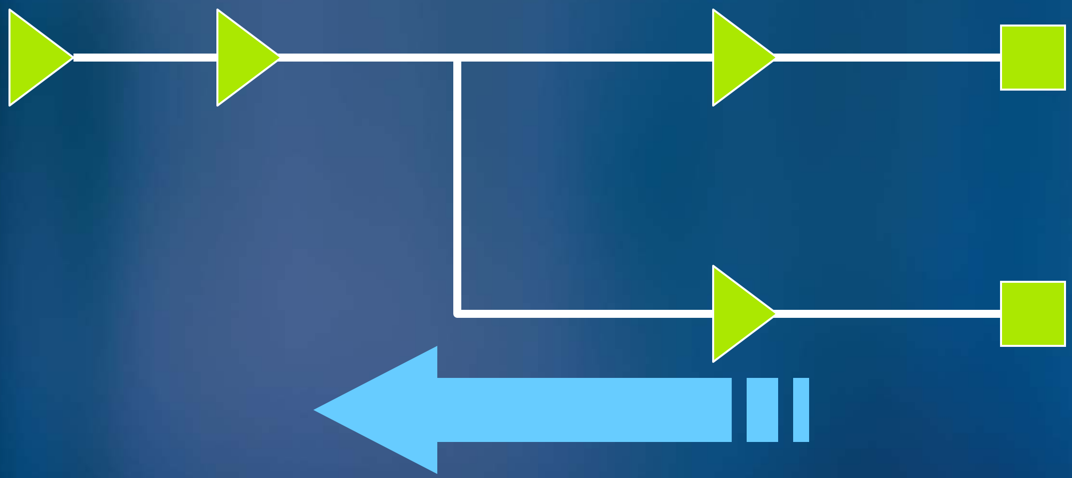


# Solution Characterization

- To model effect to upstream, a candidate solution is associated with
  - v: a node
  - Q: required arrival time
  - W: cumulative wire cost

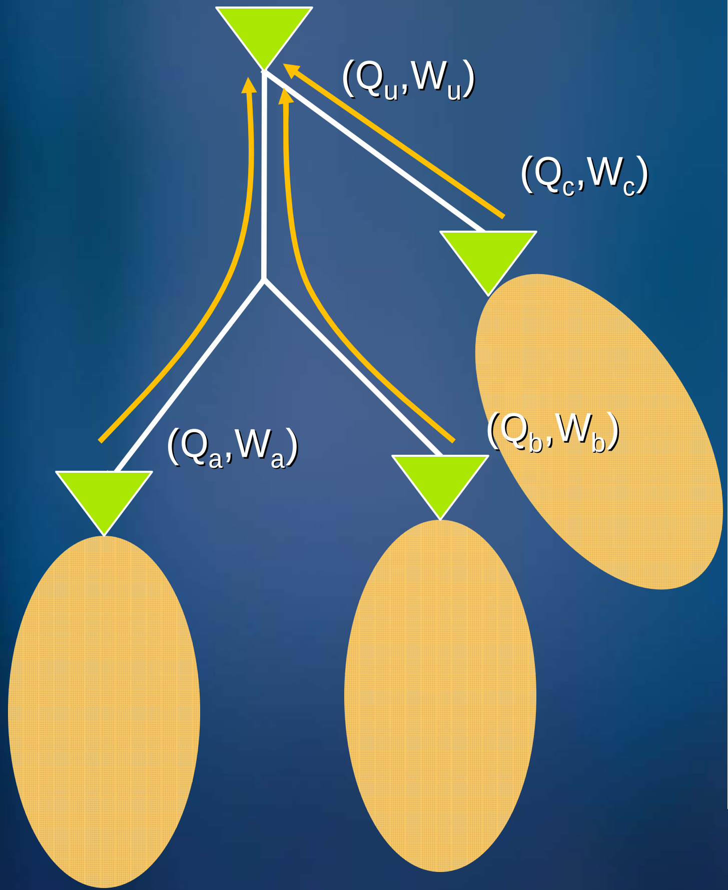


# Cost (W)-Bounded Dynamic Programming (DP)




- Start from sinks
- Candidate solutions are generated
- Two operations
  - Subtree processing
  - Solution update at buffer
- Solution Pruning

# Subtree Processing




- Three paths
  - $p_a$: a -> u
  - $P_b$: b -> u
  - $P_c$: c -> u
- $Q_u(l)=\min\{Q_a-d(p_a,l),Q_b-d(p_b,l),Q_c-d(p_c,l)\}$
- $W_u(l)=W_a+W_b+W_c+w(T,l)$
- Wires are in the same layer l

# Exponential # of Solutions

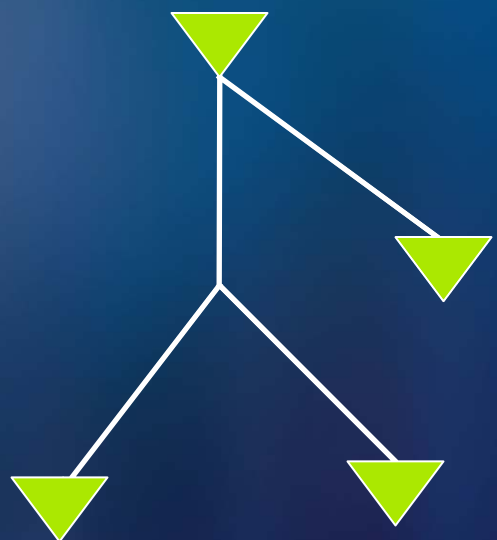
- For two solutions at a node with the same W, the one with smaller Q is dominated
- Try to only generate non-dominated solutions since most of $O(W^k)$ solutions are dominated solutions

| | |
|---|---|
| $(Q_{a,1},W_{a,1})$ | $(Q_{b,1},W_{b,1})$ |
| $(Q_{a,2},W_{a,2})$ | $(Q_{b,2},W_{b,2})$ |
| $(Q_{a,3},W_{a,3})$ | $(Q_{b,3},W_{b,3})$ |
| $(Q_{a,4},W_{a,4})$ | $(Q_{b,4},W_{b,4})$ |

# Multi-Way Merging

- If best Q for cost w is obtained by merging $Q(a^1_{i1})$, $Q(a^2_{i2})$,..., $Q(a^k_{ik})$, where $i_1+i_2+...i_k=w$, best Q for cost w+1 is obtained by

$$\max_{1 \le r \le k} \min \{Q(a^1_{i1}),Q(a^2_{i2}),..., Q(a^r_{ir+1}), ...,Q(a^k_{ik})\}$$



# Four-Branch Example

Solution(w=8, Q=9) is shown.
To compute Solution (w=9, Q)

| (W,Q) | $a^1$ | $a^2$ | $a^3$ | $a^4$ |
|---|---|---|---|---|
| 1 | (1,10) ← | (1,12) | (1,15) | (1,12) |
| 2 | (2,8) | (2,10) ← | (2,12) | (2,10) ← |
| 3 | (3,7) | (3,4) | (3,9) ← | (3,7) |
| 4 | (4,5) | (4,2) | (4,5) | (4,6) |
| 5 | (5,2) | (5,1) | (5,7) | (5,2) |

# Four-Branch Example – Case 1

Candidate Solution (w=9, Q=8)

| (W,Q) | $a^1$ | $a^2$ | $a^3$ | $a^4$ |
|---|---|---|---|---|
| 1 | (1,10) | (1,12) | (1,15) | (1,12) |
| 2 | (2,8) ← | (2,10)← | (2,12) | (2,10)← |
| 3 | (3,7) | (3,4) | (3,9)← | (3,7) |
| 4 | (4,5) | (4,2) | (4,5) | (4,6) |
| 5 | (5,2) | (5,1) | (5,7) | (5,2) |

# Four-Branch Example – Case 2

Candidate Solution (w=9, Q=4)

| (W,Q) | $a^1$ | $a^2$ | $a^3$ | $a^4$ |
|---|---|---|---|---|
| 1 | (1,10)← | (1,12) | (1,15) | (1,12) |
| 2 | (2,8) | (2,10) | (2,12) | (2,10)← |
| 3 | (3,7) | (3,4)← | (3,9)← | (3,7) |
| 4 | (4,5) | (4,2) | (4,5) | (4,6) |
| 5 | (5,2) | (5,1) | (5,7) | (5,2) |

# Four-Branch Example – Case 3

Candidate Solution (w=9, Q=5)

| (W,Q) | $a^1$ | $a^2$ | $a^3$ | $a^4$ |
|---|---|---|---|---|
| 1 | (1,10) ← | (1,12) | (1,15) | (1,12) |
| 2 | (2,8) | (2,10) ← | (2,12) | (2,10) ← |
| 3 | (3,7) | (3,4) | (3,9) | (3,7) |
| 4 | (4,5) | (4,2) | (4,5) ← | (4,6) |
| 5 | (5,2) | (5,1) | (5,7) | (5,2) |

# Four-Branch Example – Case 4

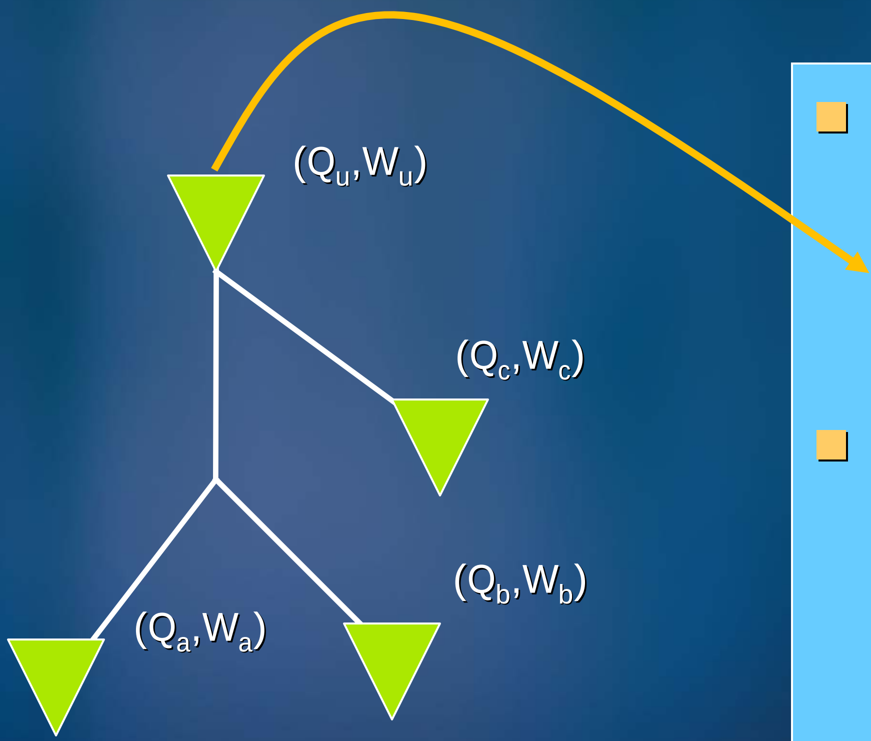
Candidate Solution (w=9, Q=7)

| (W,Q) | $a^1$ | $a^2$ | $a^3$ | $a^4$ |
|---|---|---|---|---|
| 1 | (1,10) ← | (1,12) | (1,15) | (1,12) |
| 2 | (2,8) | (2,10) ← | (2,12) | (2,10) |
| 3 | (3,7) | (3,4) | (3,9) ← | (3,7) ← |
| 4 | (4,5) | (4,2) | (4,5) | (4,6) |
| 5 | (5,2) | (5,1) | (5,7) | (5,2) |

# Linear Time Multi-Way Merging

- Lemma: given a subtree with m layers, k branches and W non-dominated solutions at each downstream buffer, one can merge them in O(mkW) time.

# Solution Update at Buffer



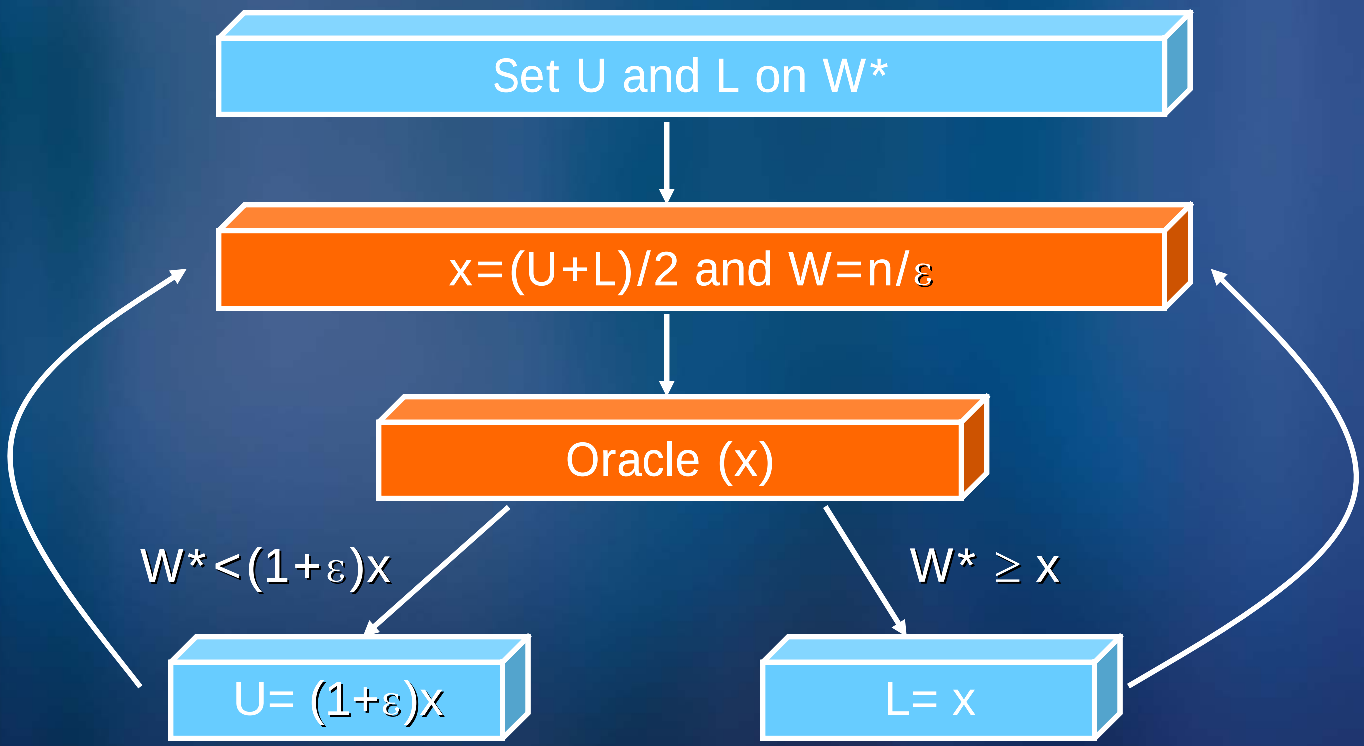

- After merging, one non-dominated solution per layer per cost, totally O(mW) solutions
- For each cost, find largest Q for all layers after buffer and propagate it

# Cost-Bounded DP

- Lemma: given a tree with n wire segments and m layers, the optimal layer assignment subject to cost budget $W=n/\varepsilon$ can be computed in $O(mnW)=O(mn^2/\varepsilon)$ time.

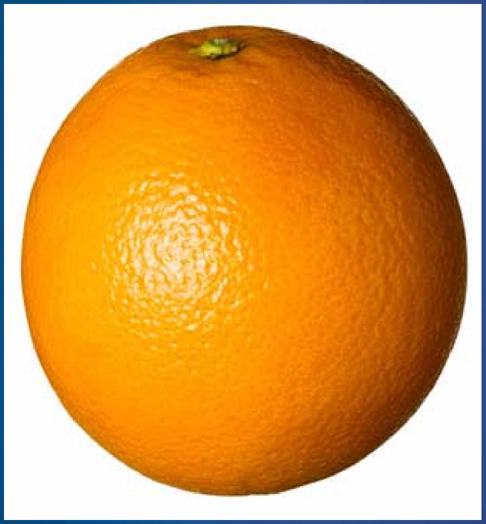
# Key 2: Bound Independent Guess

- U (L): upper (lower) bound on W*
- Naive binary search style approach



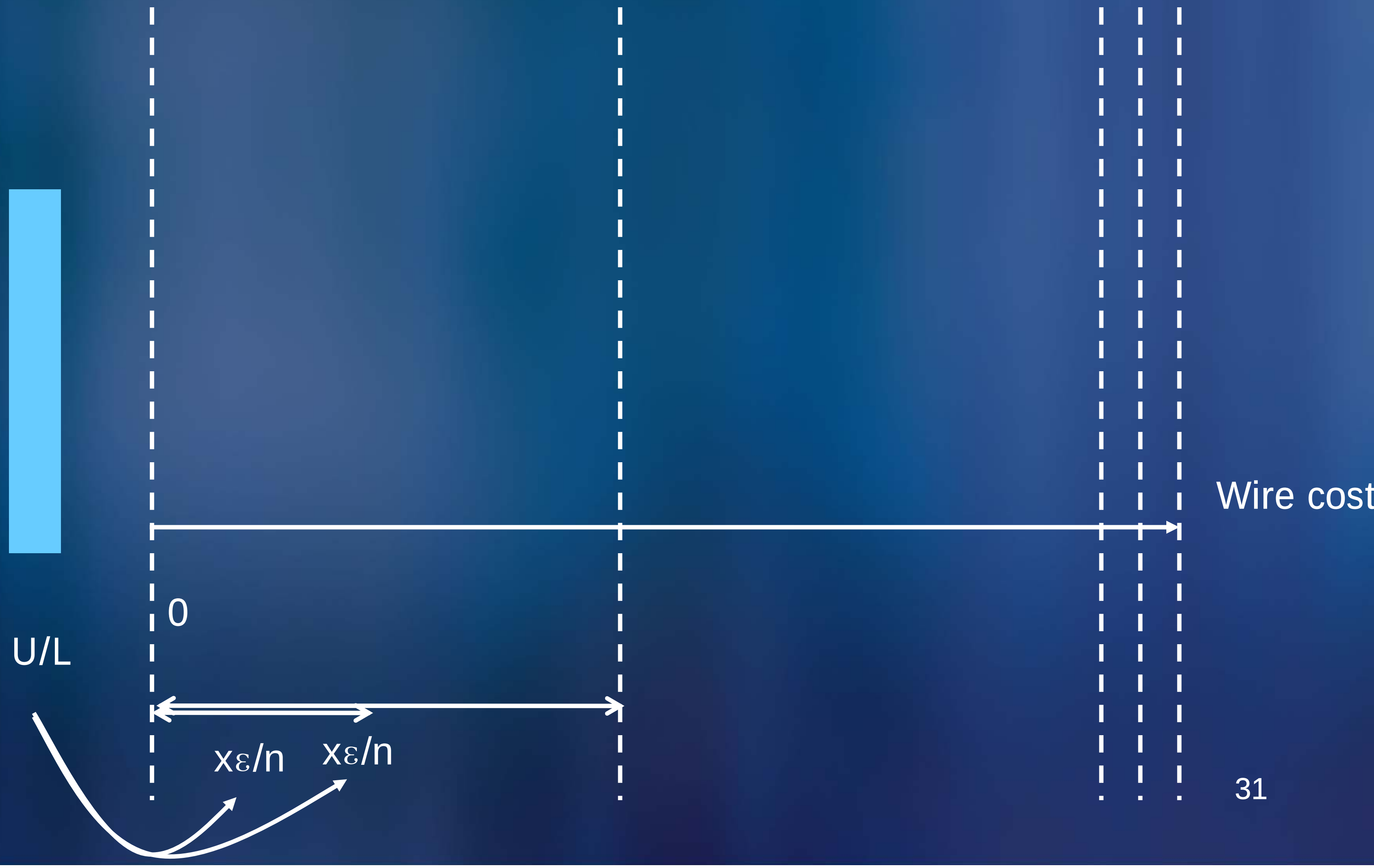

- Runtime depends on the initial bounds U and L

# Adapt ε



- Rounding factor $x\varepsilon/n$ for cost
- Larger ε: faster with rough estimation
- Smaller ε: slower with accurate estimation
- Adapt ε and relate it with U and L

# U/L Related Scale & Round



# Conceptually

- Begin with large ε' and progressively reduce it according to U/L as x approaches W*

- Set ε' as a geometric sequence of ..., 8, 4, 2, 1, 1/2, ..., ε
- One run of DP takes about O(n/ε) time. Total runtime is O(... + n/8 + n/4 + n/2 + ... + n/ε) = O(n/ε). Independent of # of iterations

# Oracle Query Till U/L<2

$$\varepsilon_i^{'} = \sqrt{\frac{W_{u,i}^*}{W_{l,i}^*}} - 1,\ x = \sqrt{\frac{W_{u,i}^* \cdot W_{l,i}^*}{1+\varepsilon_i^{'}}}$$

$$\frac{W_{u,i+1}^*}{W_{l,i+1}^*} = \left(\frac{W_{u,i}^*}{W_{l,i}^*}\right)^{3/4} \Rightarrow \frac{W_{l,i}^*}{W_{u,i}^*} = \left(\frac{W_{l,i}^*}{W_{u,i}^*}\right)^{4/3} \Rightarrow \frac{W_{l,i}^*}{W_{u,i}^*} = \left(\frac{W_{l,t}^*}{W_{u,t}^*}\right)^{(4/3)^{t-i}}$$

$$O(mn^2 \sum_{1\le i\le t} \frac{1}{\varepsilon_i^{'}}) = O(mn^2 \sum_{1\le i\le t} \sqrt{\frac{W_{l,i}^*}{W_{u,i}^*}}) = O(mn^2 \sum_{1\le i\le t} \left(\frac{W_{l,i}^*}{W_{u,i}^*}\right)^{1/2\cdot(4/3)^{t-i}})$$

$$O(mn^2 \sum_{0\le j<t} \left(\frac{W_{l,i}^*}{W_{u,i}^*}\right)^{1/2\cdot(4/3)^{j}}) = O(mn^2 \sum_{0\le j<t} \left(0.59^{1/2\cdot(4/3)^{j}}\right)) = O(mn^2)$$

# When U/L<2

Scale and round each cost by $L\varepsilon/n$

↓

$W=2n/\varepsilon$

↓

Run DP

↓

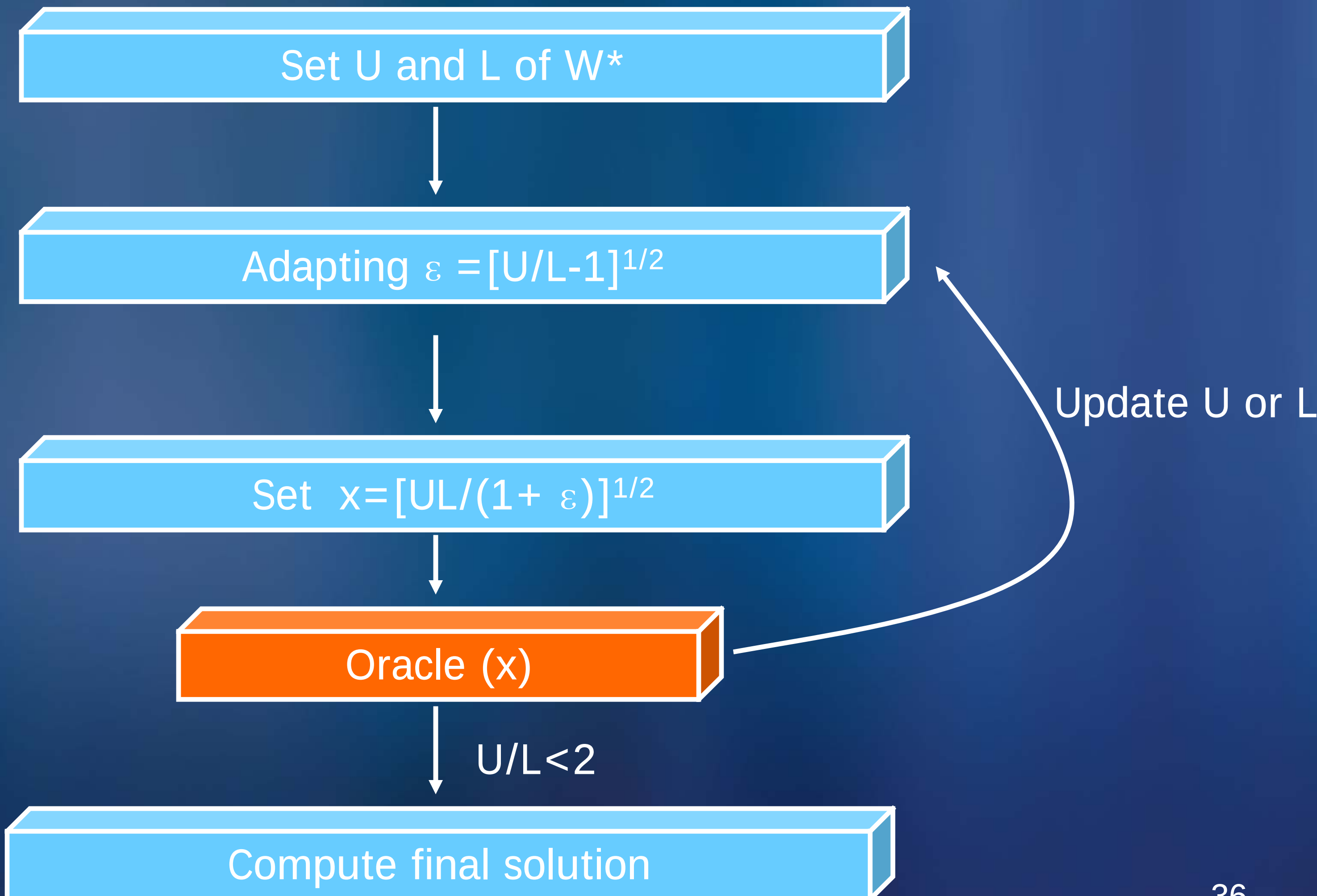
Pick min cost solution satisfying timing at driver

- At least one feasible solution, otherwise no solution w/ cost $2n/\varepsilon \bullet L\varepsilon/n = 2L \geq U$
- Runs in $O(mn^2/\varepsilon)$ time

# FPTAS for Layer Assignment

- Theorem: a $(1+\varepsilon)$ approximation to the timing constrained minimum cost layer assignment problem can be computed in $O(mn^2/\varepsilon)$ time for any $\varepsilon>0$.

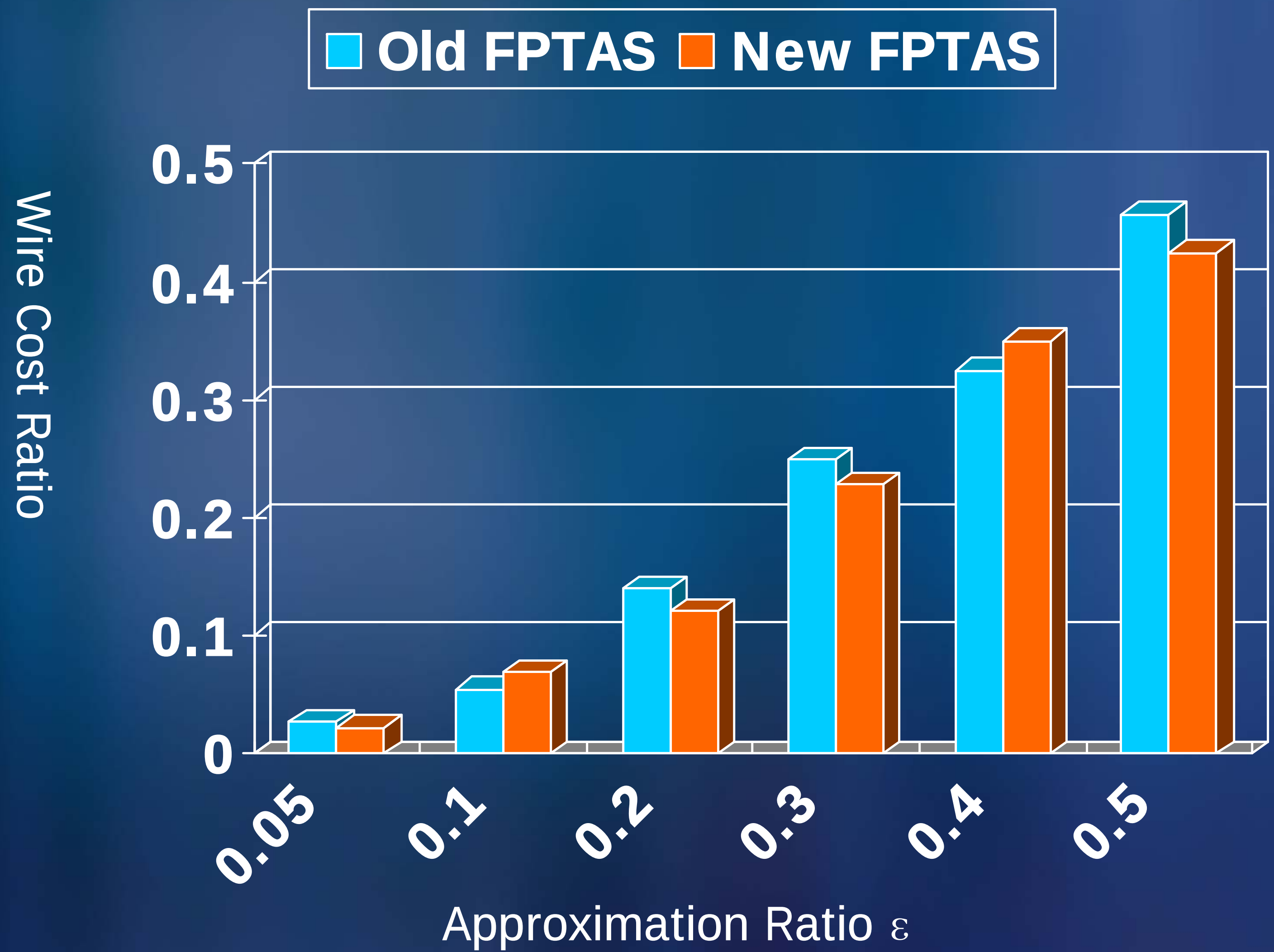
# The Algorithmic Flow



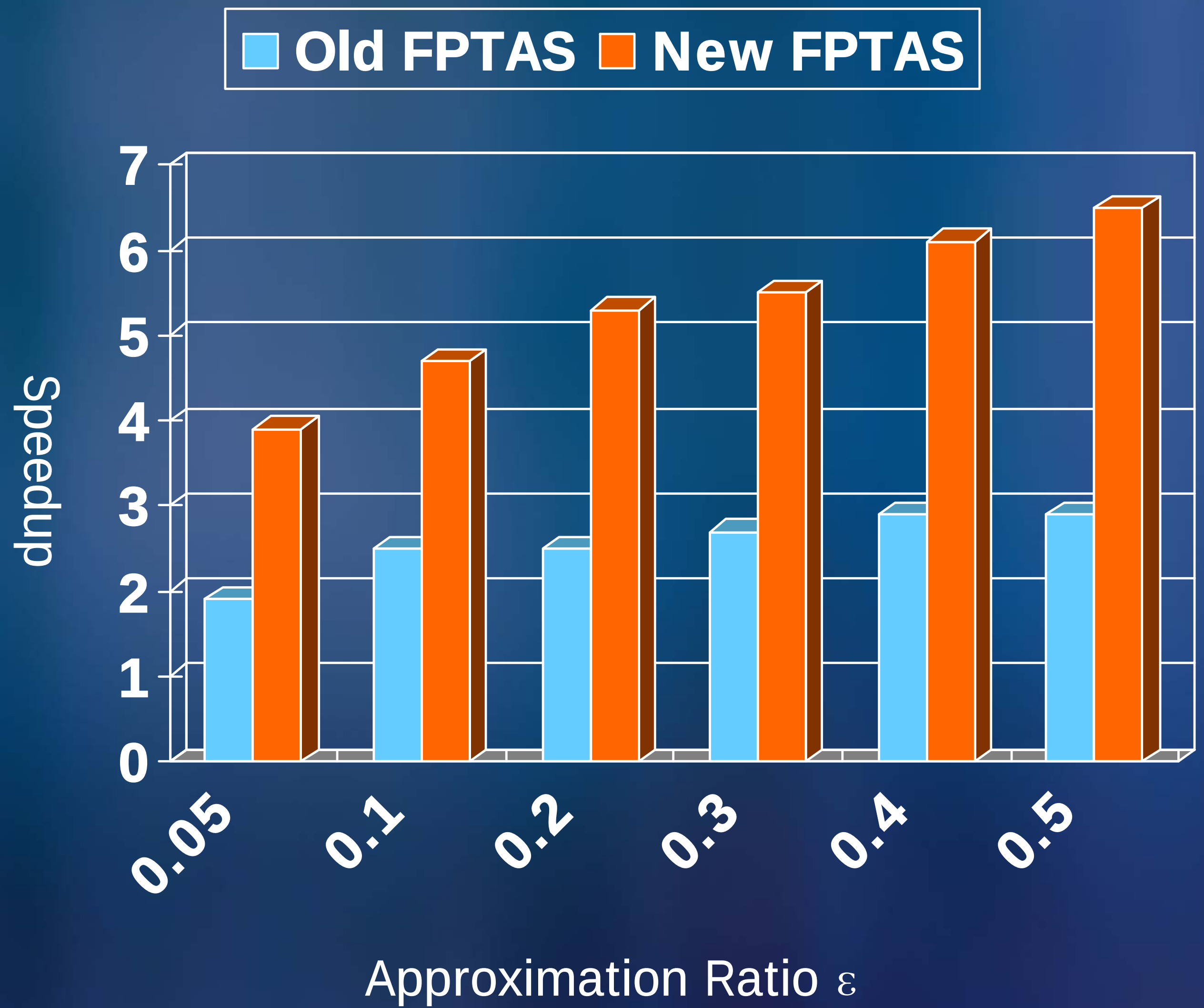
# Experiments

- Experimental Setup
  - 1000 industrial nets
- Compared to Dynamic Programming and the previous FPTAS [ICCAD'08]

# Cost Ratio Compared to DP



# Speedup Compared to DP



# Observations

- FPTAS always achieves the theoretical guarantee
- Larger $\varepsilon$ leads to more speedup
- 3.9x faster with 2.2% additional wire area compared to DP
- Up to 6.5x faster than DP
- On average about 2x faster than previous FPTAS

# Conclusion

- Propose a (1+ ε) approximation for timing constrained layer assignment for any ε > 0 running in $O(mn^2/\varepsilon)$ time
  - Linear time DP running in O(mnW) time
  - Bound independent oracle query
  - Up to 6.5x faster than DP and 2x faster than previous FPTAS
  - Few percent additional wire area compared to DP as guaranteed theoretically

# *Thanks*