# A ROBUST ECO ENGINE BY RESOURCE-CONSTRAINT-AWARE TECHNOLOGY MAPPING AND INCREMENTAL ROUTING OPTIMIZATION

Shao-Lun Huang

Chi-An Wu

Kai-Fu Tang

Chang-Hong Hsu

Chung-Yang (Ric) Huang

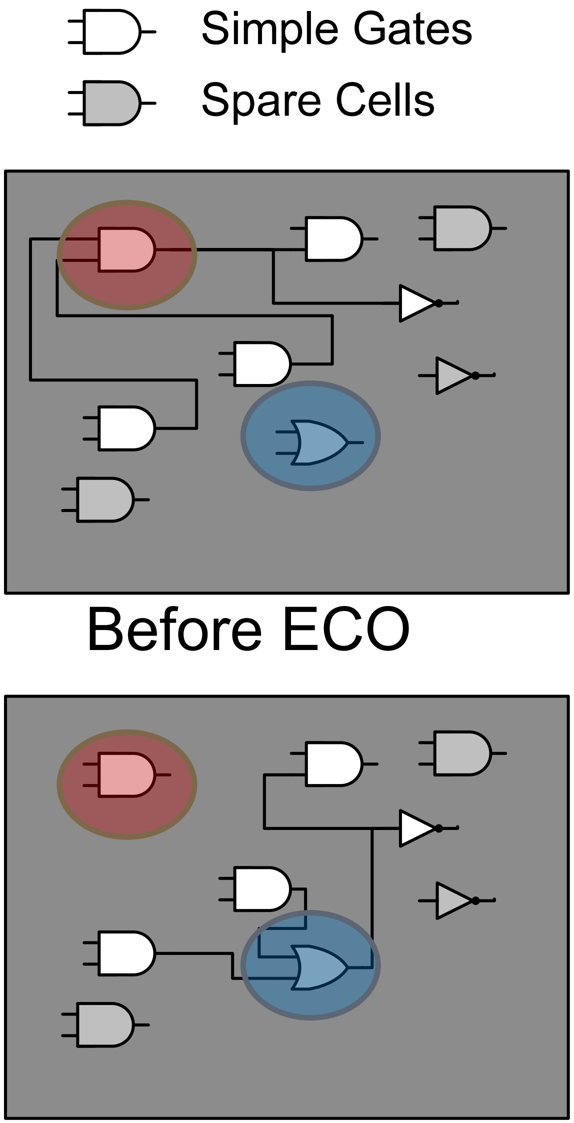
**Design Verification Lab**
**Dept. EE/ Graduate Instiute of Electronics Engineering**
**National Taiwan University, Taiwan**

# Outline

- Introduce to ECO problem
- Motivation
- Contribution
- Flow Chart of Our Methods
- Main Algorithm
    - RSA Technology mapping Algorithm
    - Incremental Routing Optimization
- Experimental Results
- Conclusion

# ECO problems

- Functional/Metal-only ECO Spare Cell Selection
- Given
  - Placed netlist
  - Library
  - Spare Cells
  - ECO List
- Objective
  - Correctly apply functional changes
  - Minimize additional wire length

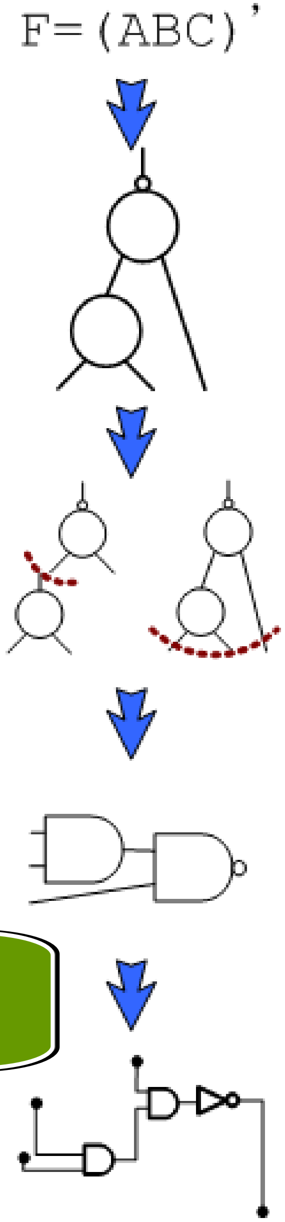


Before ECO

After ECO

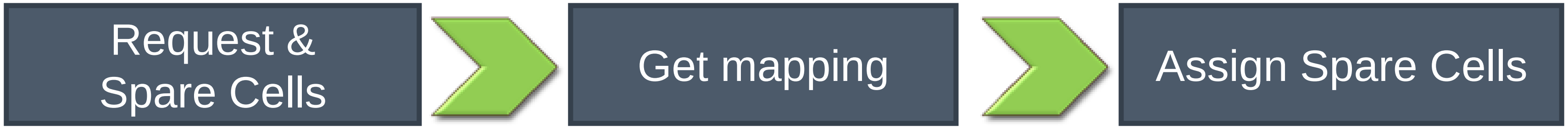
# Motivation

- Many ECO works
  - Tech. Mapping + Max-Flow
    - Y.-M. Kuo et. al. ICCAD 2007
  - Tech. Mapping + Simulated Annealing
    - Modi et. al. ICCD 2008
  - Tech. Mapping + Stable Marriage Heuristic
    - Jiang et. al. DAC 2009
- Is that the best choice?



The best choice for ECO?

# Example



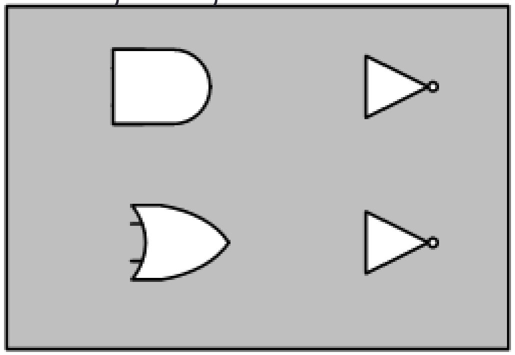
Request & Spare Cells

Get mapping

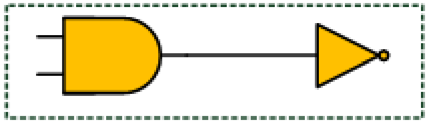
Assign Spare Cells

**ECO requests:**
NAND2,  AND2

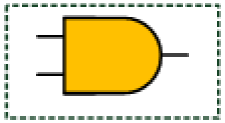
**Spare Cells:**
AND, OR, INVx2

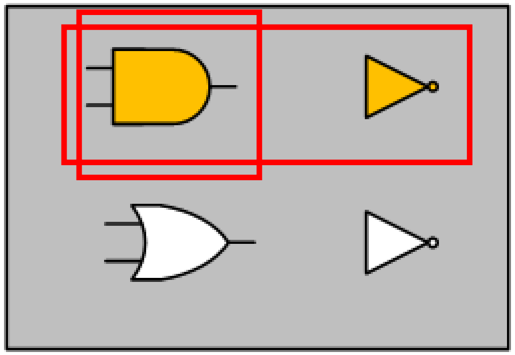


NAND2



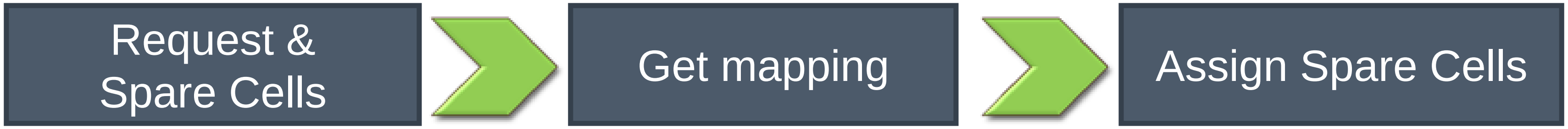
AND2





**Fail!**

# Example



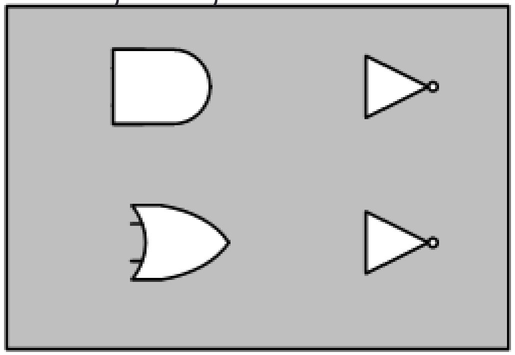
Request & Spare Cells

Get mapping

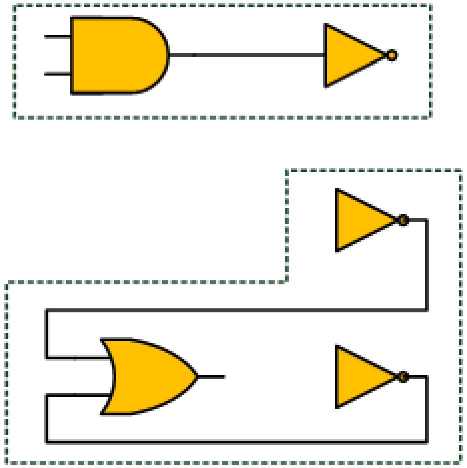
Assign Spare Cells

**ECO requests:**
NAND2, AND2

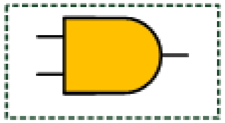
**Spare Cells:**
AND, OR, INVx2

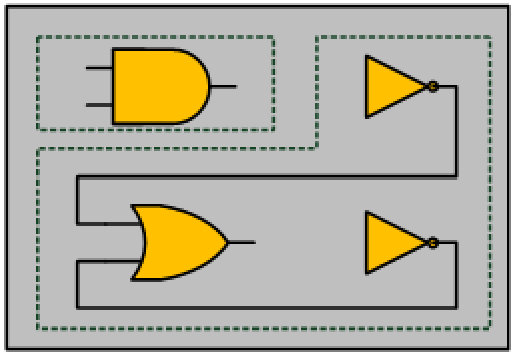


NAND2



AND2



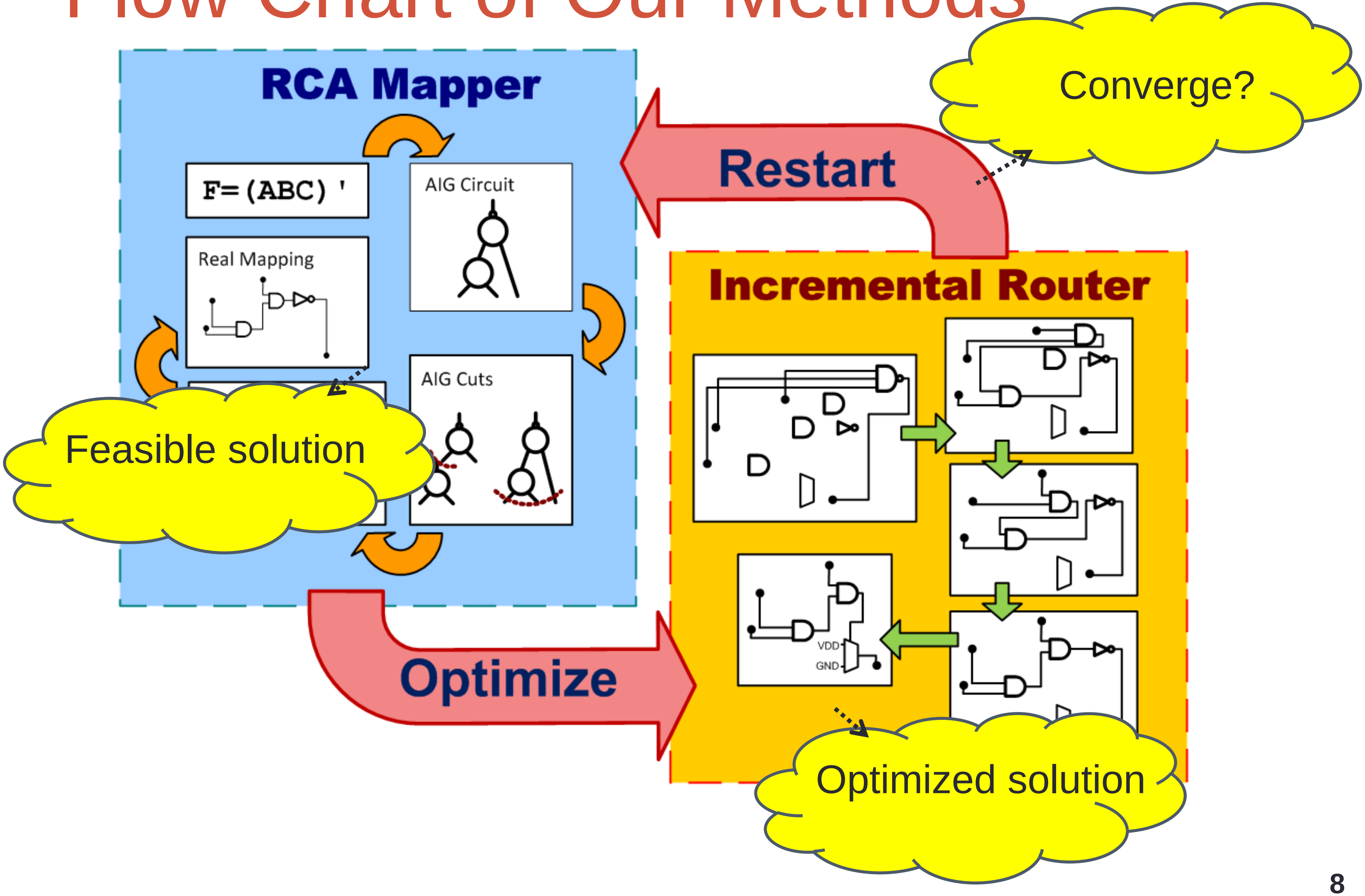


**Pass!**

# Contributions

- Guarantee a feasible solution
  - Even when spare cells limited
- Resource-Constraint-Aware Tech. Mapping
  - Robust
  - Formal
- Incremental Routing Optimization
  - Fast
  - Incremental

# Flow Chart of Our Methods



# Outline

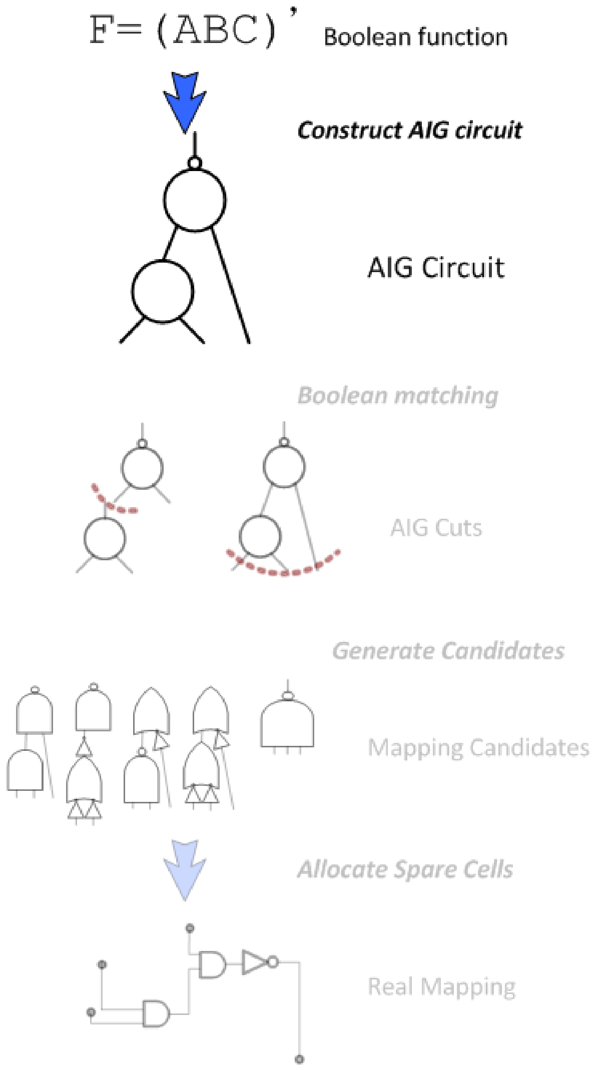
- Introduce to ECO problem
- Motivation
- Contribution
- Flow Chart of Our Methods
- Main Algorithm
  - RSA Technology mapping Algorithm
  - Incremental Routing Optimization
- Experimental Results
- Conclusion

# RSA Technology Mapping Algorithm

1. Construct AIG circuit
2. Boolean Matching
3. Generate Candidates
4. Allocate Spare Cells

# Step 1: Construct AIG circuits

- Objective
  - Boolean Formula → AIG
- Map each ECO gates
- Without optimization



$F = (ABC)'$ Boolean function

*Construct AIG circuit*

AIG Circuit

*Boolean matching*

AIG Cuts

*Generate Candidates*

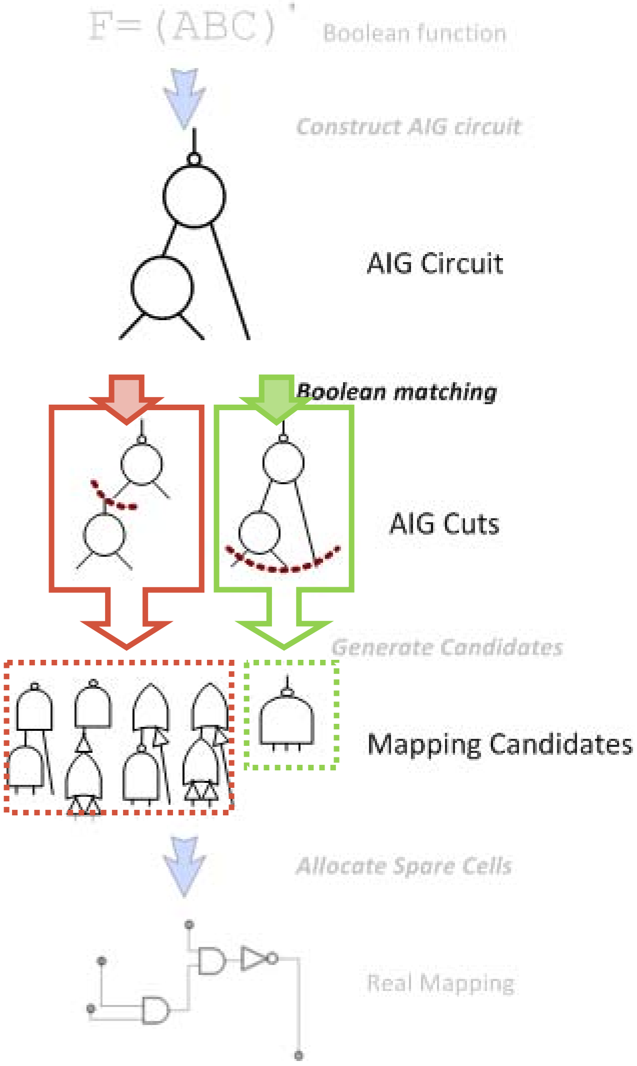
Mapping Candidates

*Allocate Spare Cells*

Real Mapping

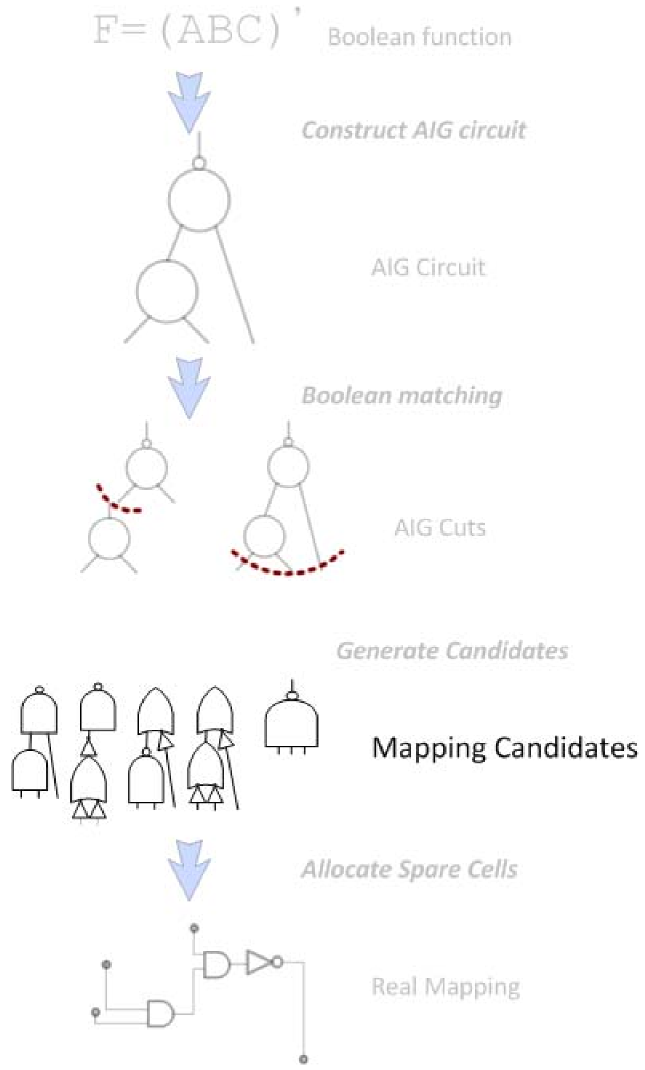
# Step 2: Boolean Matching

- AIG → Mapped Gates
- Cut-based Boolean Matching
- Truth table
    - 5-feasible cuts
    - Most cells ≤ 5 inputs
- Functionally Complete Gate
    - MUX NAND NOR



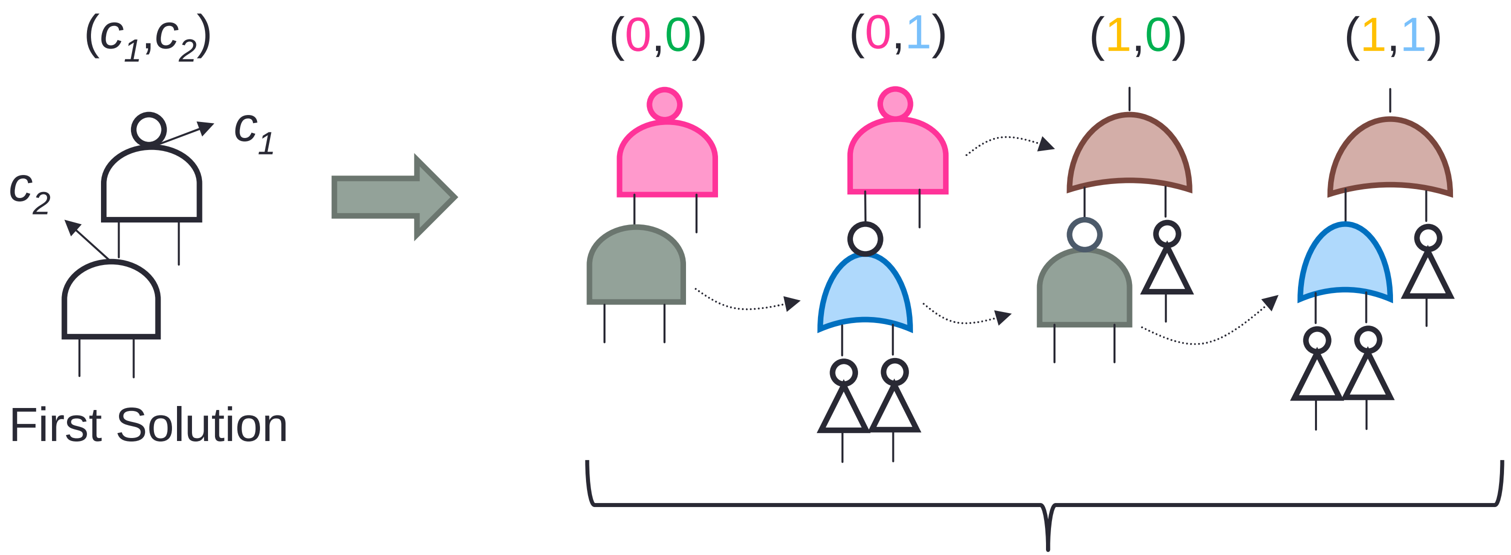
# Step 3: Generate ECO candidates

- Objective
  - AIG nodes → Gates
- ECO gates are small
  - Enumerate more than one mapping candidate



# How to enumerate all candidates?

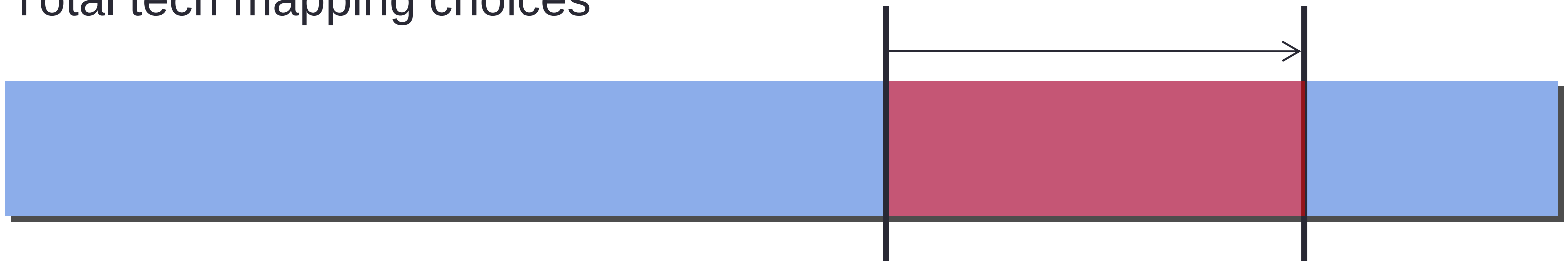
Enumerate mapping solutions from PI toPO



First Solution

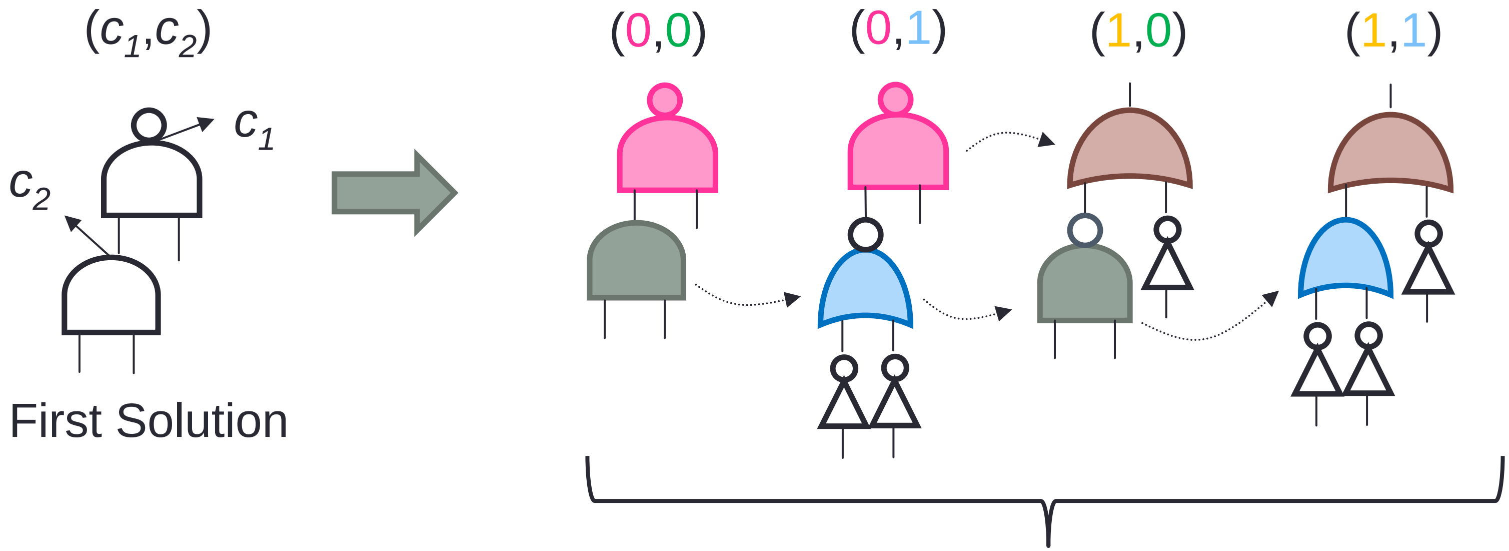
Enumerating *N* circuits

Total tech mapping choices



# How to enumerate all candidates?

Enumerate mapping solutions from PI toPO



$(c_1,c_2)$

$c_1$

$c_2$

First Solution

(0,0)

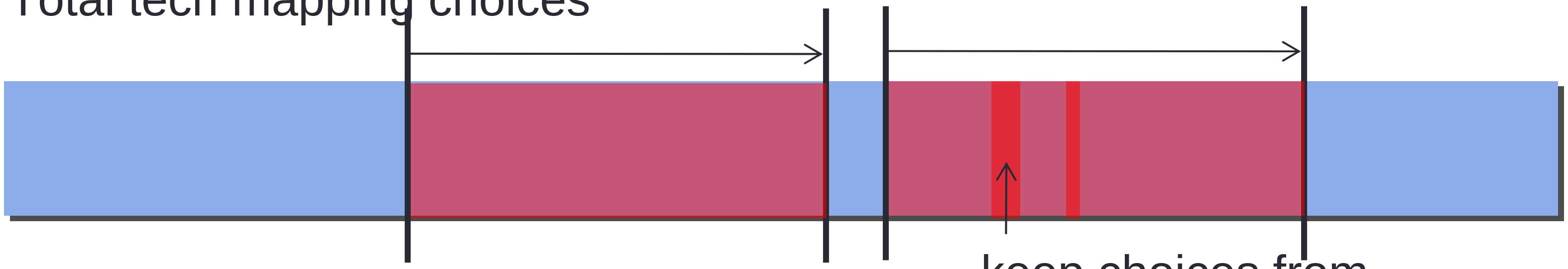
(0,1)

(1,0)

(1,1)

Enumerating *N* circuits

Total tech mapping choices

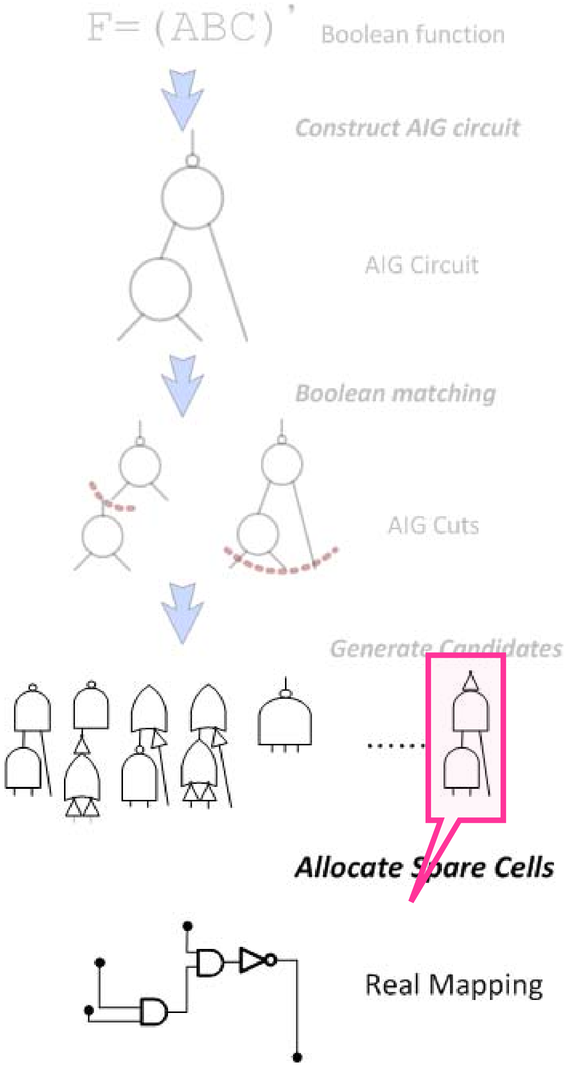


Arbitrarily start

keep choices from
previous optimal solutions

# Step 4: Allocate spare cells to ECO gates

- Objective
  - Distribute spare cells
- Methods
  - Greedy
  - Pseudo-Boolean

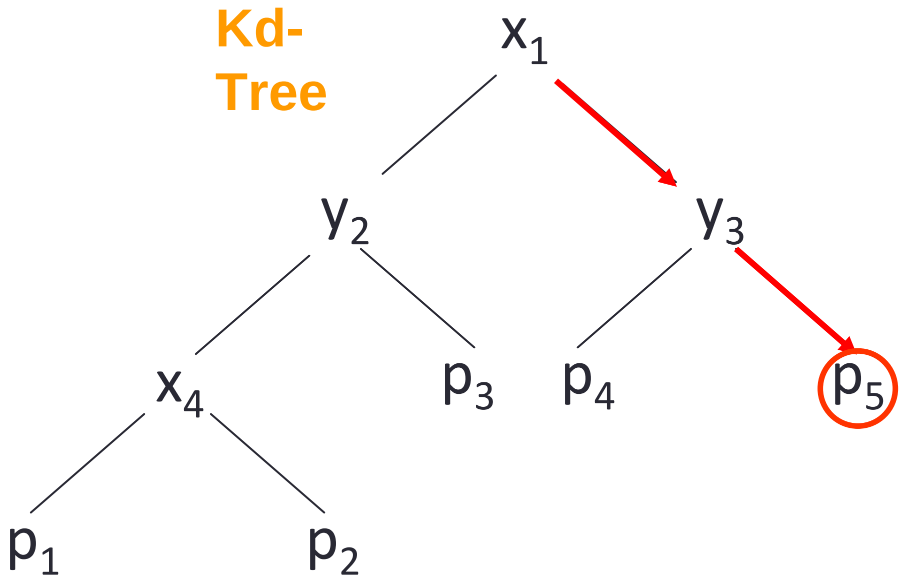


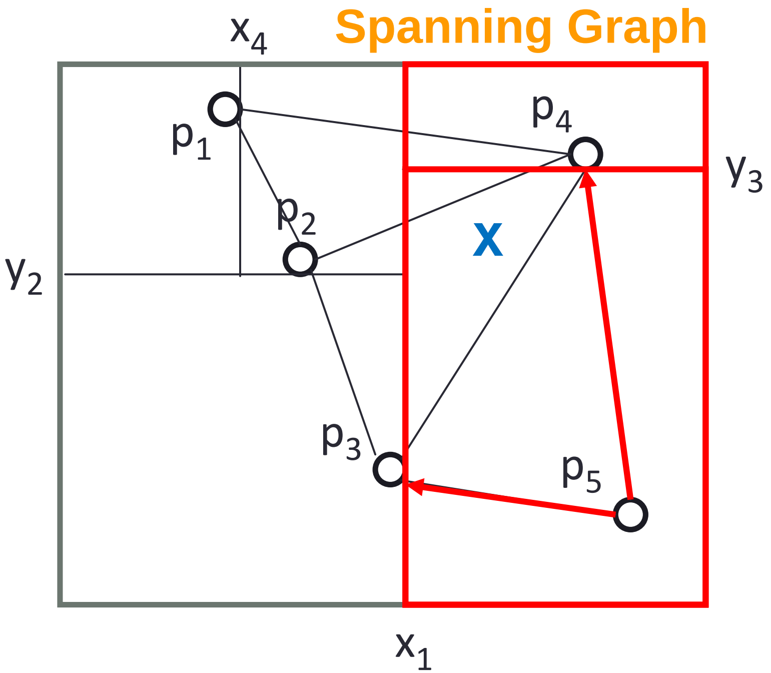
# Incremental Optimizer

- Neighborhood Searching Algorithm
- Optimization Techniques
  - Alternating Inner Topology
  - Swapping Cells
  - Integrating Constant Insertion
  - Searching Better ECO Candidates

# Neighborhood Searching

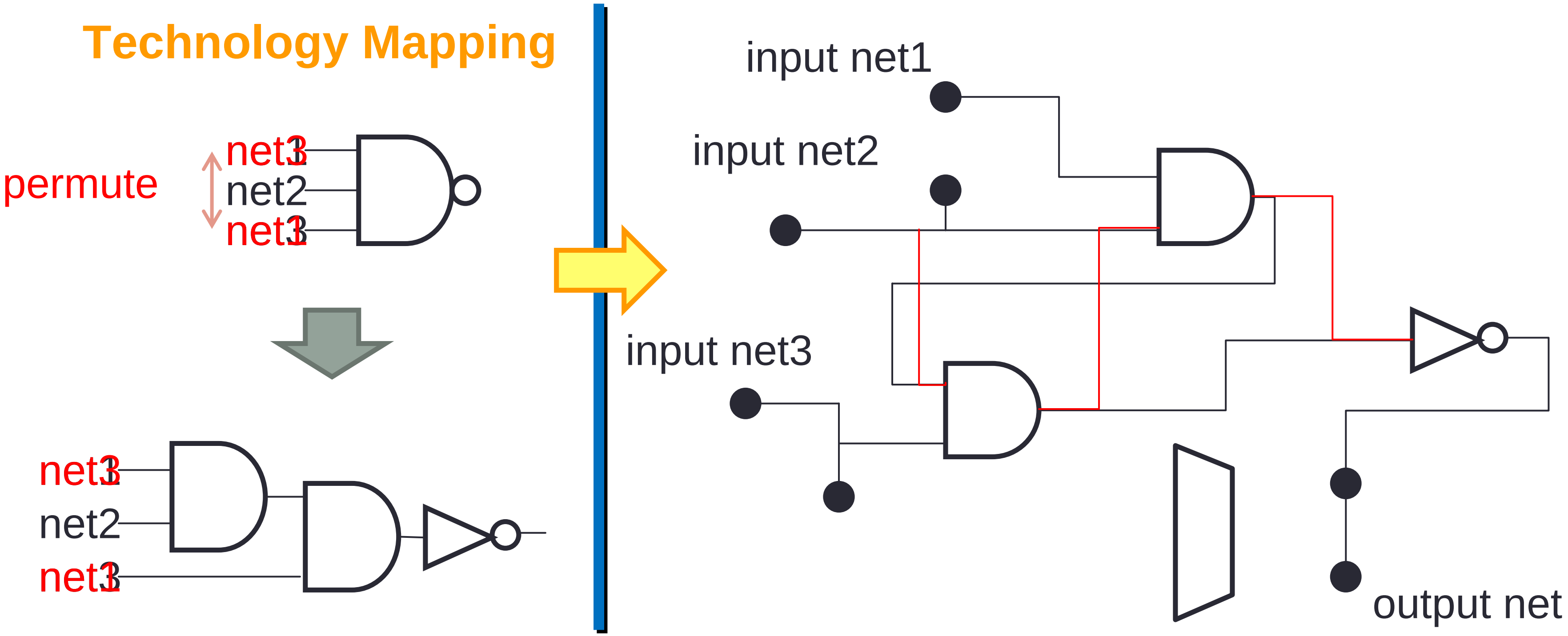
- Given a position
  1. Search for the corresponded cell  on Kd-Tree
  2. Search for its nearby cells from spanning graph





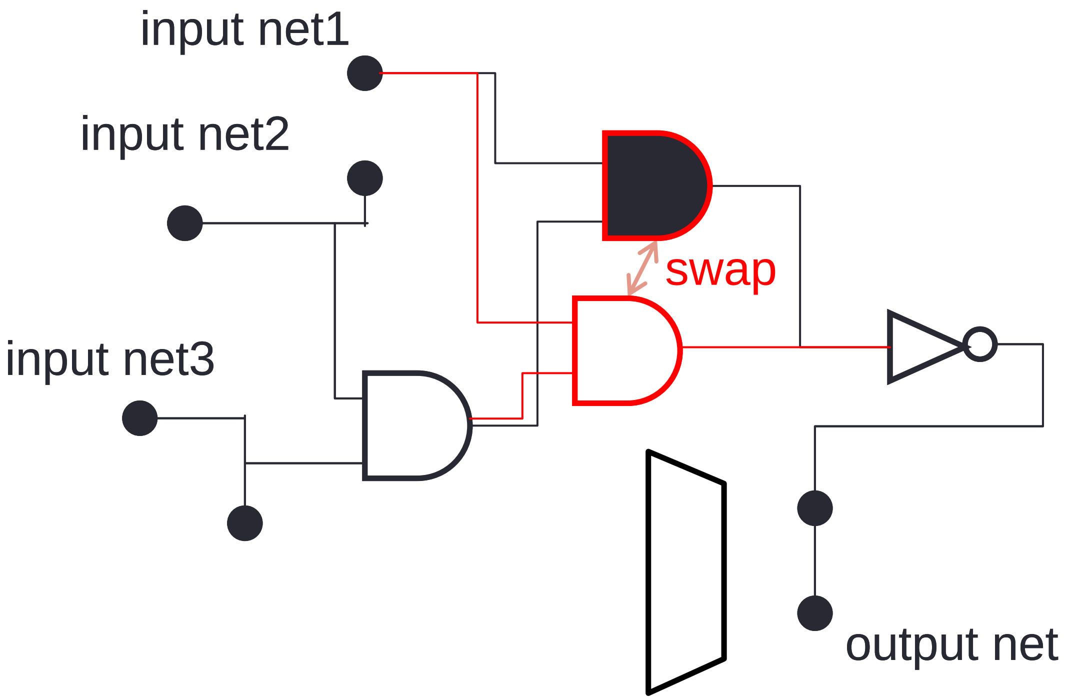
# Alternating Inner Topology

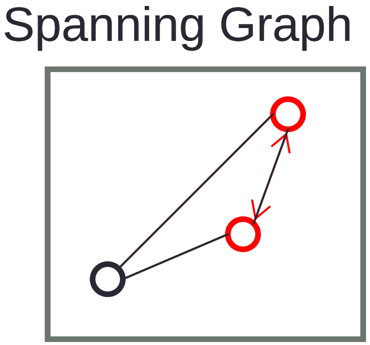
- 3~6 inputs: Enumerate all possible combinations



# Swap Cells

- Swap cells on spanning graph

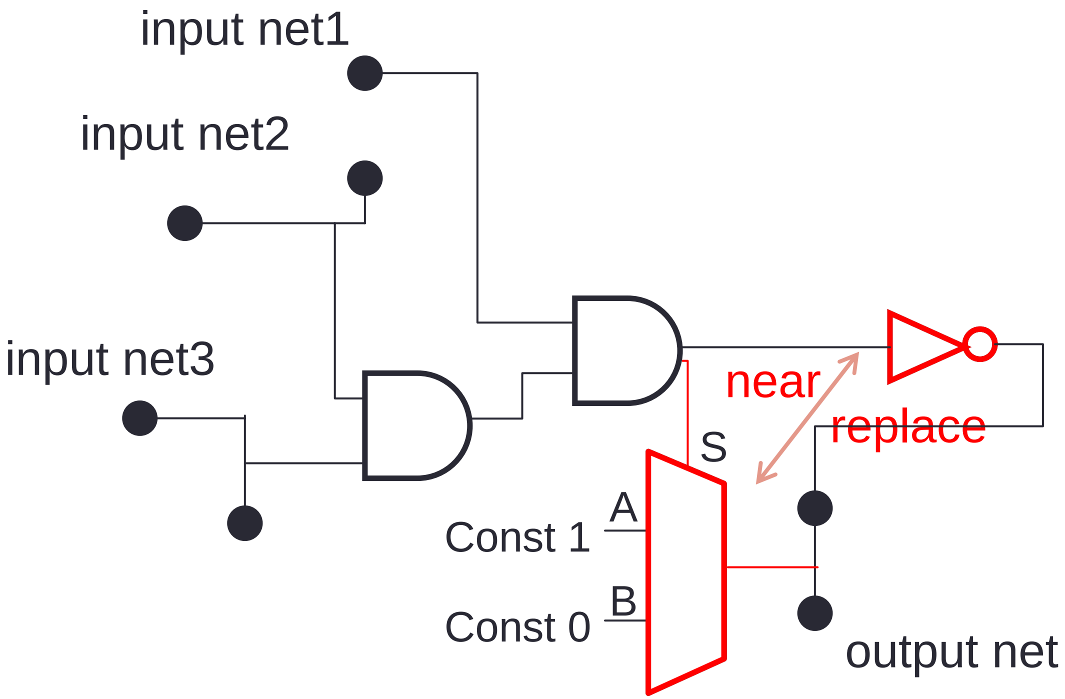




# Constant Insertion

- Change gate types
- NOR, NAND, MUX → INV

  MUX → AND, OR



# Outline

- Introduce to ECO problem
- Motivation
- Contribution
- Flow Chart of Our Methods
- Main Algorithm
    - RSA Technology mapping Algorithm
    - Incremental Routing Optimization
- Experimental Results
- Conclusion

# Testcases

- 6 Industrial cases
- Wire length: Half-Perimeter Bounding Box

| Circuit | #pin | #gate | #net | #spare | #ECO | HPBB |
|---|---|---|---|---|---|---|
| Test1 | 483 | 28,591 | 28,705 | 350 | 7 | 4.05x$10^9$ |
| Test2 | 483 | 30,891 | 28,705 | 2,300 | 49 | 4.14x$10^9$ |
| Test3 | 483 | 30,891 | 28,705 | 2,300 | 94 | 4.14x$10^9$ |
| Test4 | 33 | 198 | 181 | 40 | 3 | 2.31x$10^9$ |
| Test5 | 30 | 938 | 850 | 100 | 4 | 1.29x$10^9$ |
| Test6 | 3,354 | 73,586 | 76,413 | 2,419 | 63 | 4.46x$10^9$ |

MUX+INV

NAND2 only

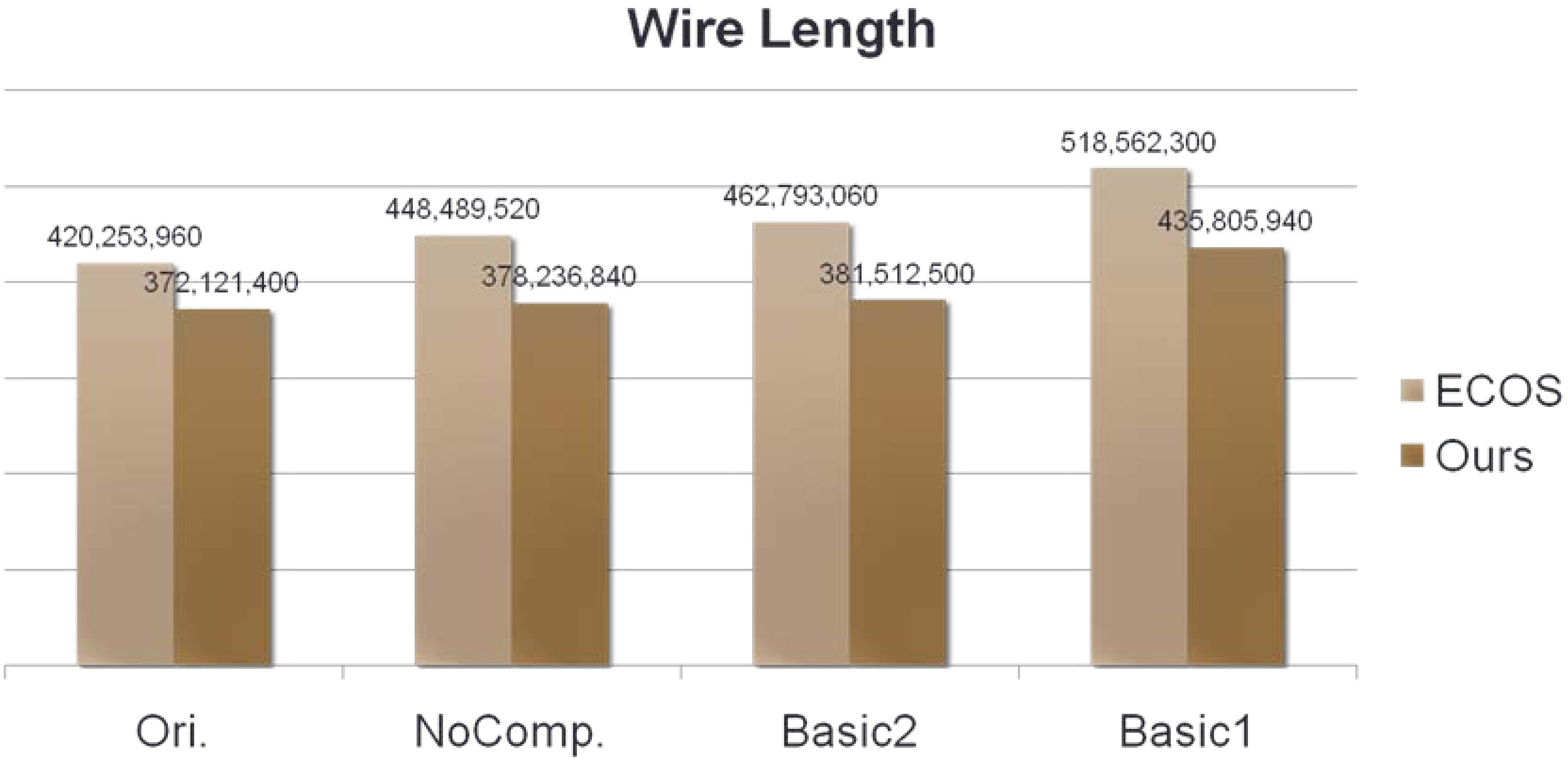
- 10% to 70% improvement
- 43% CPU time

| Circuit | ECOS [9] | | | Ours | | |
|---|---|---|---|---|---|---|
| | HPBB | ECO-WL | Tine | HPBB | ECO-WL | Tine |
| Test1 | 4.05e9 | 0.04% | 1.71 | 4.05e9 | 0.04% | 1.00 |
| Test2 | 4.35e9 | 5.00% | 87.39 | 4.31e9 | 4.03% | 38.32 |
| Test3 | 4.56e9 | 10.14% | 61.04 | 4.51e9 | 8.98% | 45.46 |
| Test4 | 2.32e6 | 0.56% | 0.20 | 2.32e6 | 0.56% | 0.04 |
| Test5 | 1.30e7 | 0.34% | 0.24 | 1.30e7 | 0.34% | 0.08 |
| Test6 | 4.50e9 | 0.87% | 36.16 | 4.47e9 | 0.23% | 10.69 |
| Ratio | | 1 | 1 | | 0.81 | 0.43 |

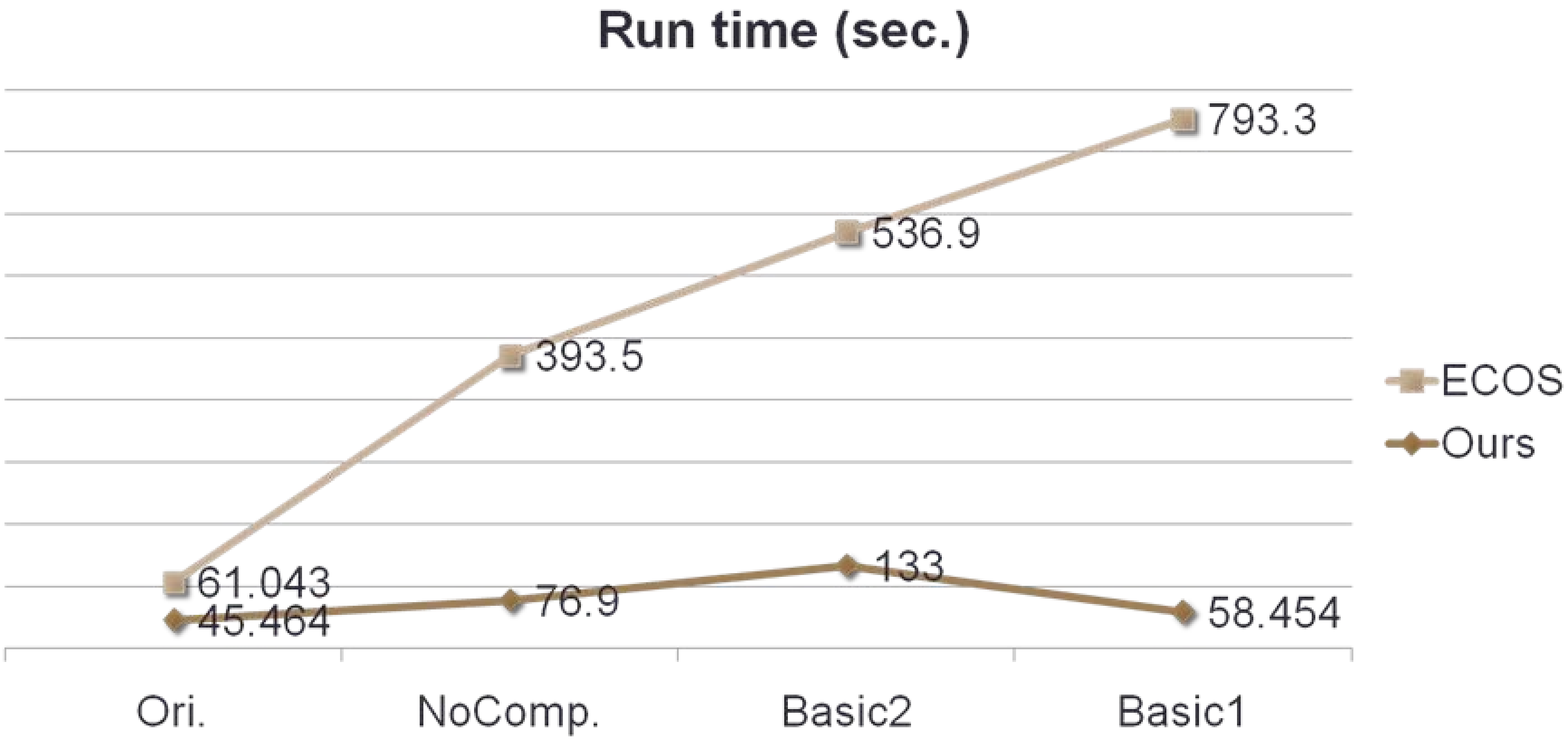
# When Spare Cells Are Sparse

- Robustness study on Testcase 3.
  - Sequential Circuits
  - Require remapping
- Spare Cells
  - Basic Cell 1: INV + AND2 + OR2
  - Basic Cell 2: Basic1 + NAND2 + NOR2 + MUX
  - No Complex Cell: Basic2 + XOR
  - Original Cell: NoComp. + AOI22 + MAO222 + FA1

# Wire Length


Wire Length
420,253,960
372,121,400
448,489,520
378,236,840
462,793,060
381,512,500
518,562,300
435,805,940
ECOS
Ours
Ori.
NoComp.
Basic2
Basic1

# Runtime


Run time (sec.)
793.3
536.9
393.5
61.043
45.464
76.9
133
58.454
ECOS
Ours
Ori.
NoComp.
Basic2
Basic1

# Conclusions

- Robust ECO engine
  - Map with spare cells.
  - Especially when spare cells sparse.
- Resource-constraint-Aware Tech. mapping
  - Fast & Robust
  - Incremental
  - Larger solution space
- Incremental Routing Optimizer
  - Fast
  - Good routing cost

# Future works

- Permute original netlist
  - Enlarge mapping solutions
- Use less Spare cells
  - Reduce design cost

*Thanks For Your Attention*