SAT-Based Optimal Hypergraph Partitioning with Replication

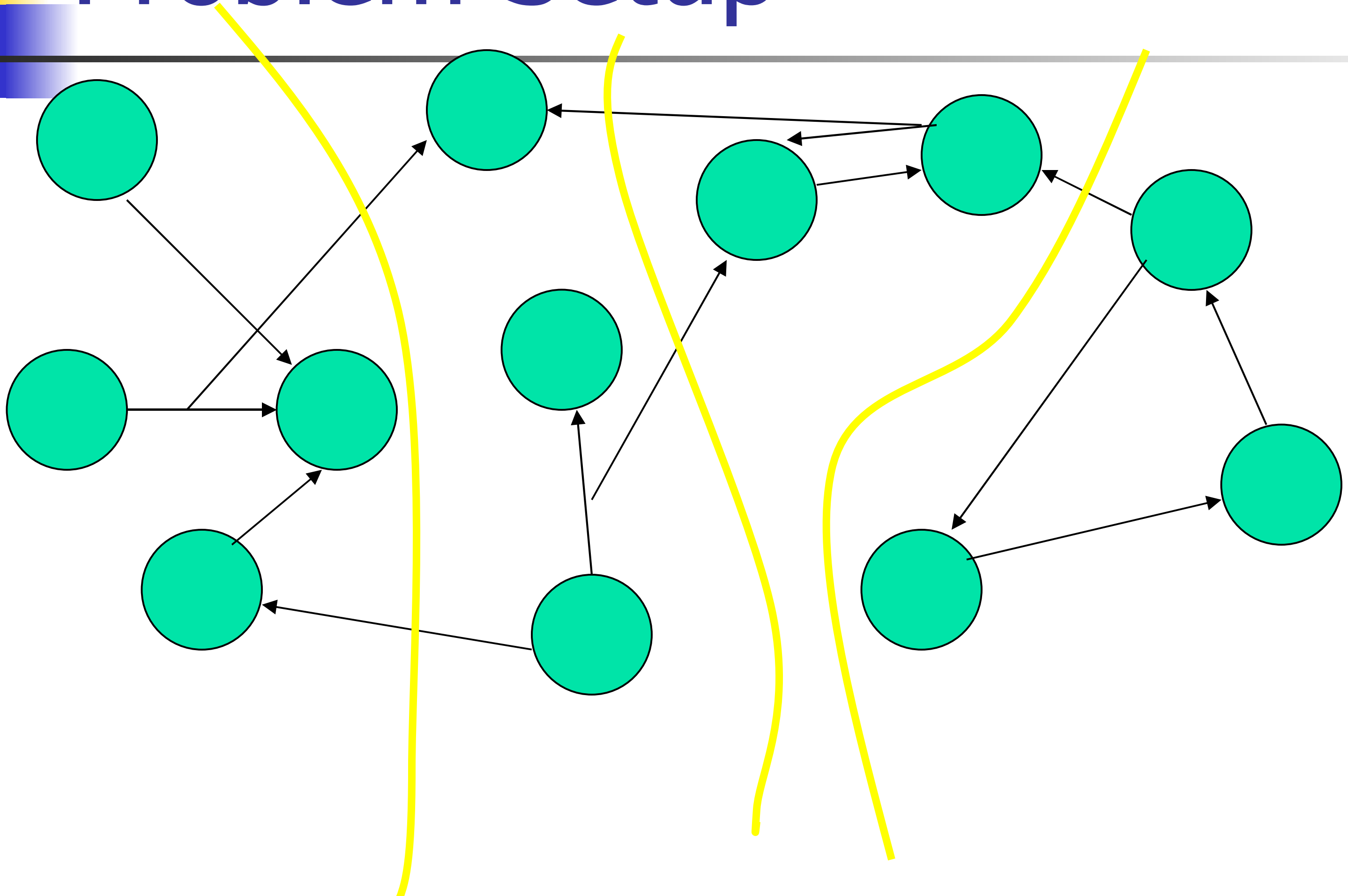
Michael Wrighton/Andre DeHon

mwrighton@tabula.com

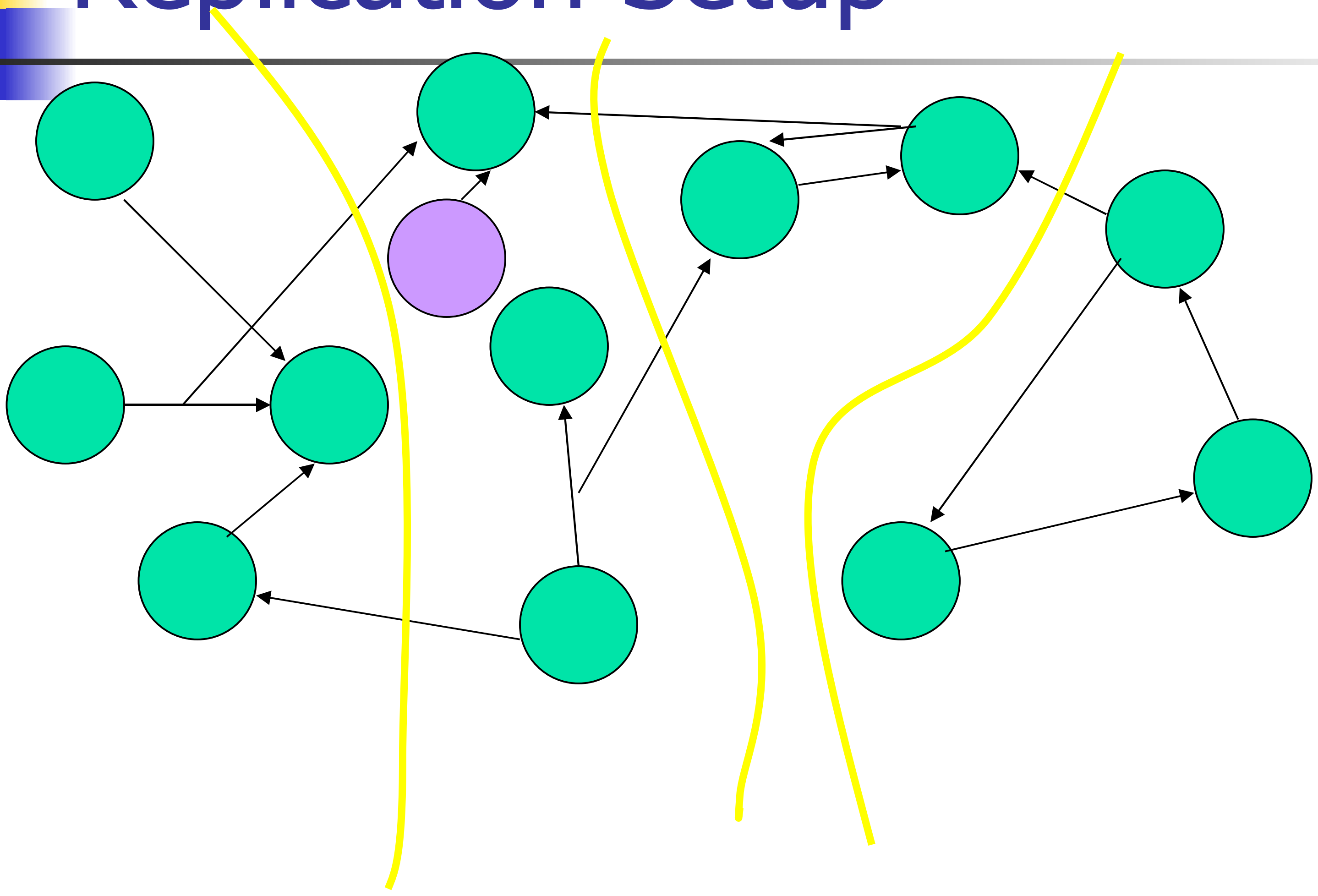
Outline

- Problem Setup/Motivation
- Related Work
- Simple SAT Partitioning Formulation
- Techniques for Improving Efficiency
- Results
- More Sophisticated Metrics
- Future Directions

Problem Setup



Replication Setup



Current State of the Art

- All Useful, Balanced Optimization Functions NP-Hard
- Simple 2-Way Partitioning
  - Optimal Solutions Available via Branch and Bound
  - Suggest Fidducia-Matthyses Techniques Near Optimal
  - But BB Results Only Scale to ~50 nodes

Current State of the Art

- Multiway Partitioning
  - No published optimal results on VLSI netlists
    - Synthetic netlist experiments
  - Much larger solution space
  - Typically solved via recursive bipartitioning
- Partitioning with Replication
  - Optimal Techniques for Unbalanced Case
    - Based on Network Flows
  - Unknown Quality of Heuristics

What’s Missing?

- Optimal Solutions
  - Multiway Partitioning
  - Replication
  - More “Realistic” Cost Functions
- Solutions Could Guide Heuristic Development
- Solve Practical Problems

Optimal Partitioning Algorithm Development

- Capo “Small” Partitioner
  - Best available optimal partitioning tool
  - 2-way cut-hyperedges formulation
  - Relies on pruning techniques applicable only to bipartitioning
- Not Clear How to Generalize Techniques to Multiway, Multiobjective, and Replication Formulations

Optimization vs. Decision Problems

- Any Optimization Problem Can Be Transformed into a Series of Decision Problems

```
while (upperBound > lowerBound) {
  thisTry = (upperBound + lowerBound) / 2
  if (existsSolution(this)) {
     upperBound = thisTry
  }
  else {
     lowerBound = thisTry
  }
}
```

Partitioning at Some Metric:
NP-Complete
Decision Problem

NP-Complete Problems

- Fundamental Property of NP-Complete Decision Problem:
- Can Transform An Instance from One Member of the Class to Another Within Polynomial Time and Space
- Offers Mechanism to Leverage Advances Between Problem Domains

Early SAT Partitioning

- S. Devadas, ICCAD1989
  - Bipartitioning Formulation
  - 32 Node Netlists
    - SAT Solvers & Hardware of 1989
  - “An attractive feature of this approach is that the entire space of feasible solutions can be represented in a compact way, facilitating the search for optimal solutions under complex cost functions and associated constraints.”

SAT Solver Development

- Very Competitive Marketplace for SAT Solvers
  - ICSAT Conference Annual Competition
  - Standard Input Format
    - Conjunctive Normal Form (Product of Sums)
  - Trivial Output
- Fast 2005 Solver ~10x Faster than Fastest 2004 Solver
  - Not Atypical
- Practical to Solve Millions of Clauses

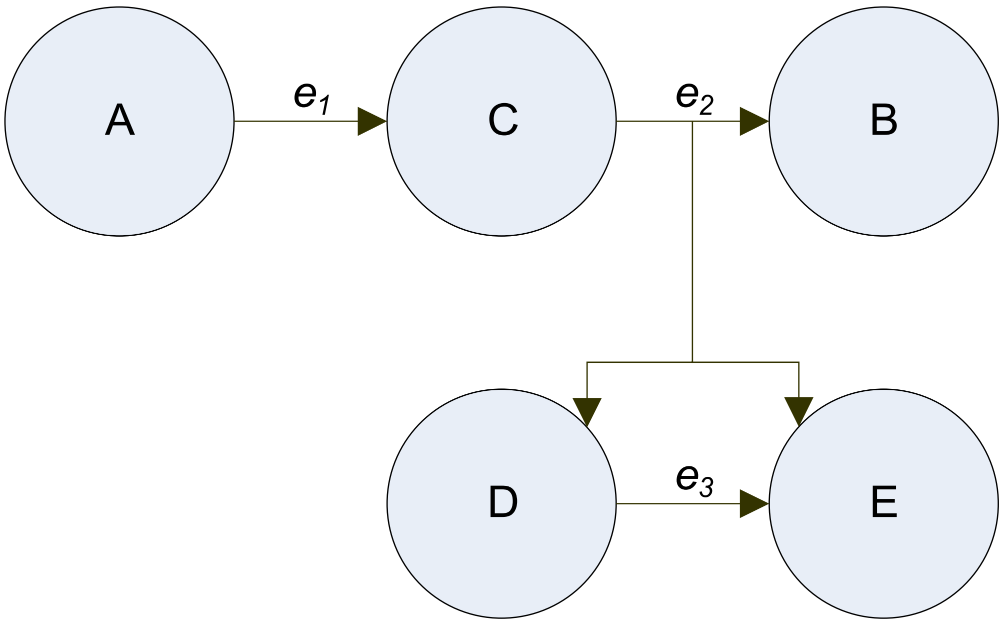
Our Formulation

- Build Problem as Traditional Logic Circuit
- For Every Node, Assign K Inputs (one for each partition)
  - i.e. $A_1 \ldots A_k, B_1 \ldots B_k$, …
  - Assert Exactly One of K Inputs Set (for now)

$$
\begin{aligned}
SAT = \ & AllNodesRepresented \\
& \wedge \ PartitionsBalanced \\
& \wedge \ MetricMet
\end{aligned}
$$

Example Partitioning Problem

- 3-Way Partition

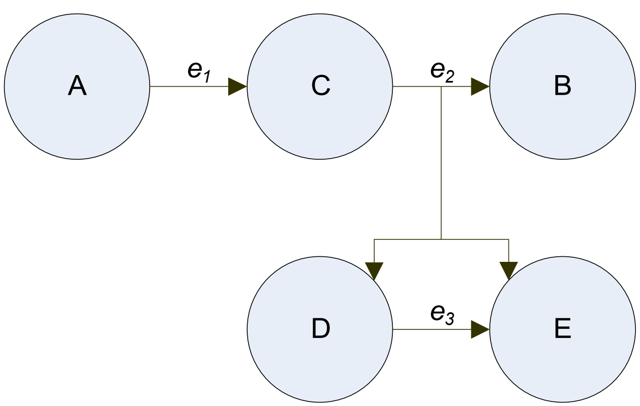


$$SAT = AllNodesRepresented \wedge PartitionsBalanced \wedge MetricMet$$

Components

- All Nodes Represented in A Partition

$$AllNodesRepresented = \begin{array}{ll} (A_0 \vee A_1 \vee A_2) & \wedge \\ (B_0 \vee B_1 \vee B_2) & \wedge \\ (C_0 \vee C_1 \vee C_2) & \wedge \\ (D_0 \vee D_1 \vee D_2) & \wedge \\ (E_0 \vee E_1 \vee E_2) & \end{array}$$



$$AllNodesRepresented = \bigwedge_{A \in Nodes} \left(\bigvee_{0 <= i < nparts} \left(A_k \right) \right)$$

$$SAT = AllNodesRepresented \wedge PartitionsBalanced \wedge MetricMet$$

Components

- Balance Condition

$$PartitionsBalanced = \bigwedge_{0 < i <= nparts} \left(\left(\sum_{A \in Nodes} A_i \right) \leq MaxSize \right)$$

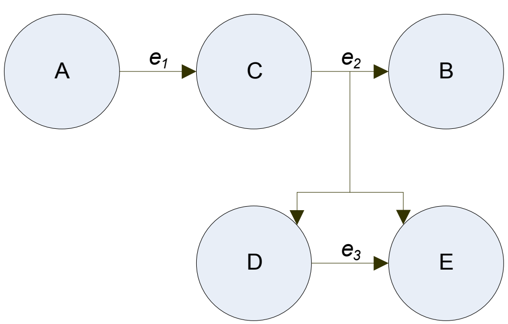
- Binary Counter and Comparators
- More Efficient Representation
  - Bailleux and Boufkhad

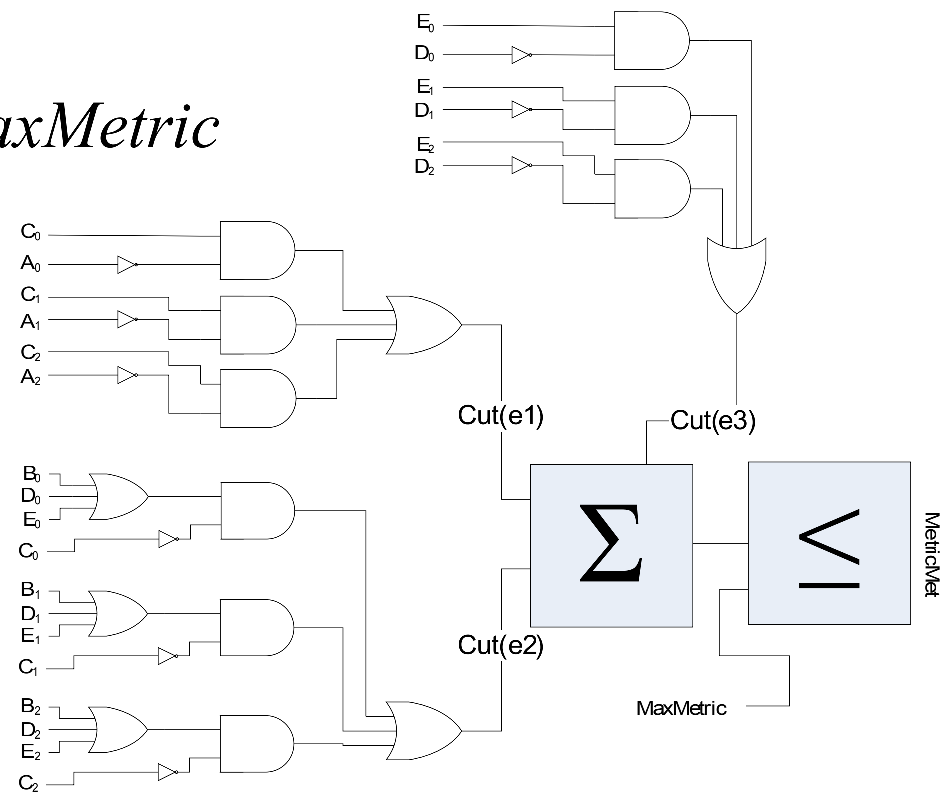
$$SAT = AllNodesRepresented \wedge PartitionsBalanced \wedge MetricMet$$

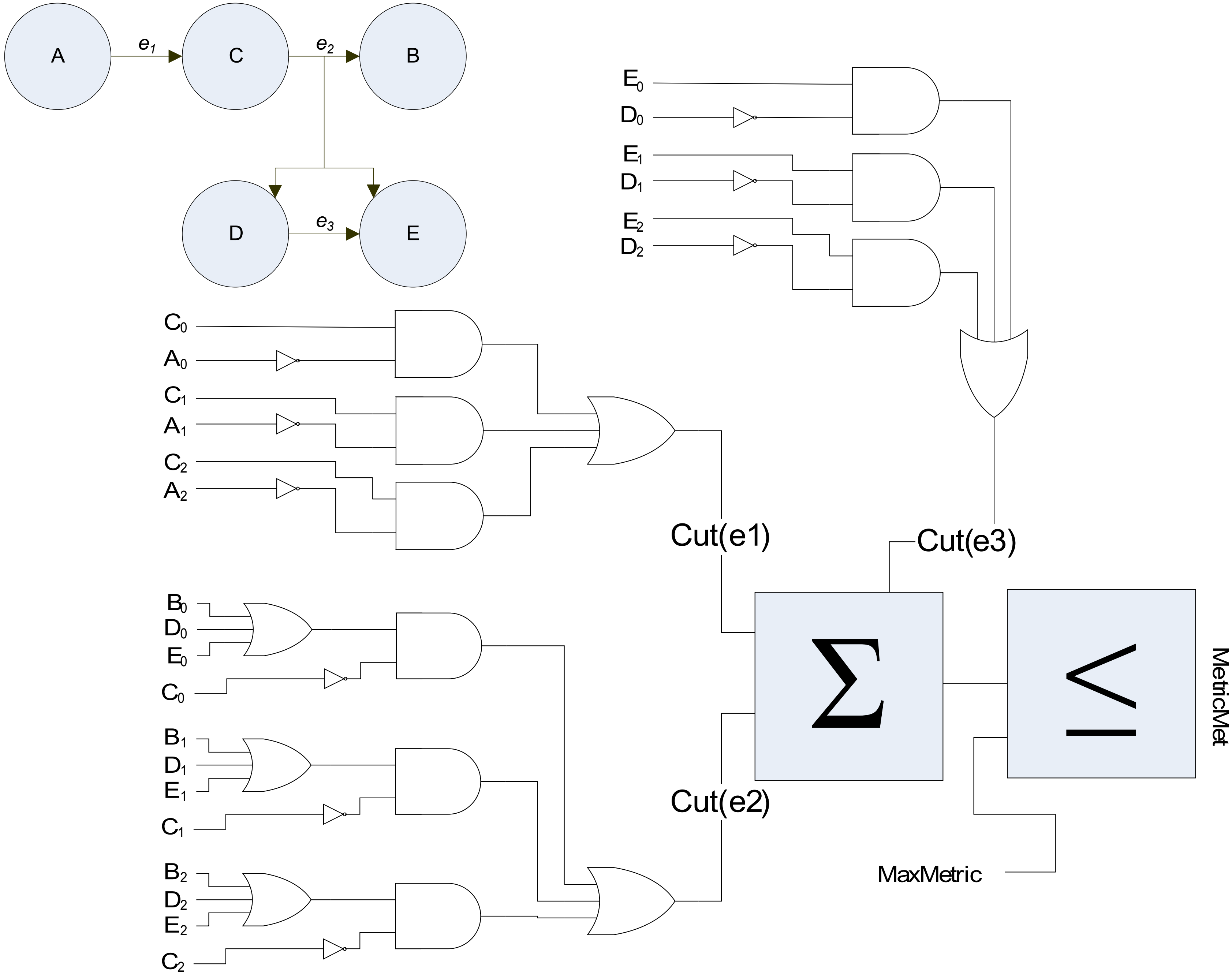
Cut Hyperedges

$$Cut(e) = \bigvee_{0<=i<nparts} \bigvee_{s\in e.Sinks} \left(\overline{e.Source_i} \wedge s_i\right)$$

$$MetricMet = \sum_{e\in Edges} Cut(e) \le MaxMetric$$



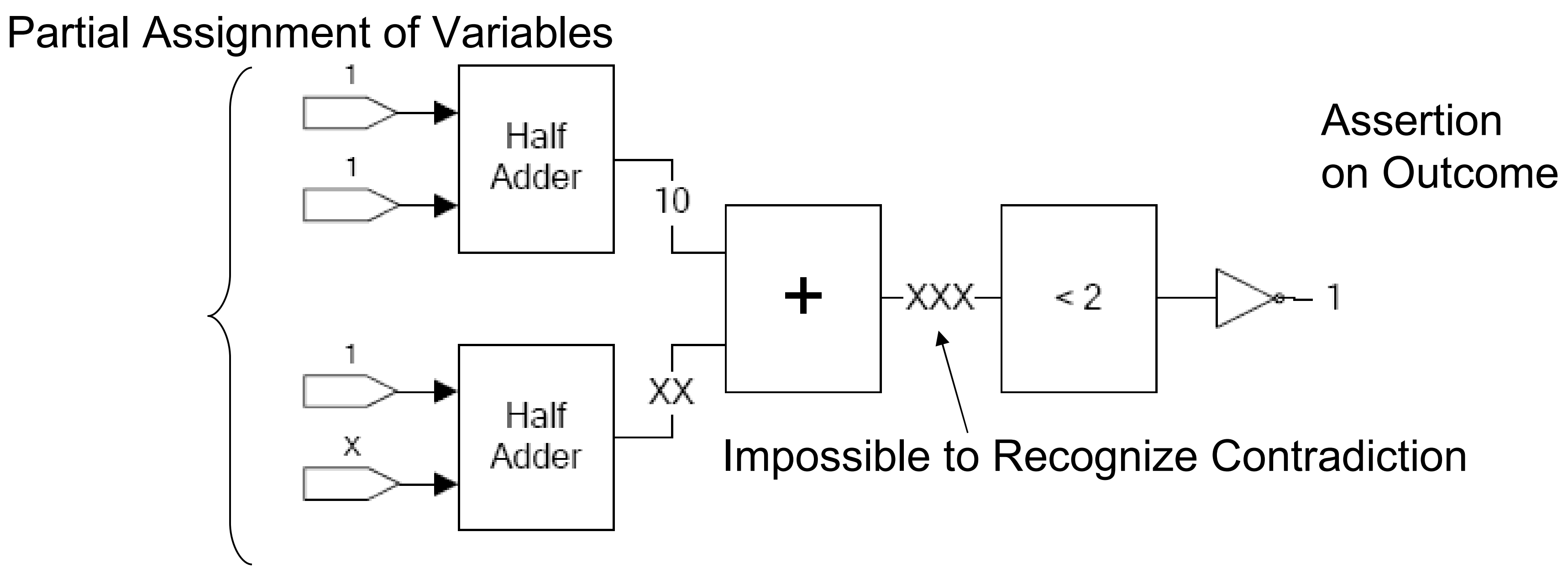




A
C
B
D
E
e_1
e_2
e_3
E_0
D_0
E_1
D_1
E_2
D_2
C_0
A_0
C_1
A_1
C_2
A_2
Cut(e1)
Cut(e3)
B_0
D_0
E_0
C_0
B_1
D_1
E_1
C_1
B_2
D_2
E_2
C_2
Cut(e2)
Σ
$\leq$
MetricMet
MaxMetric

Cardinality Constraints

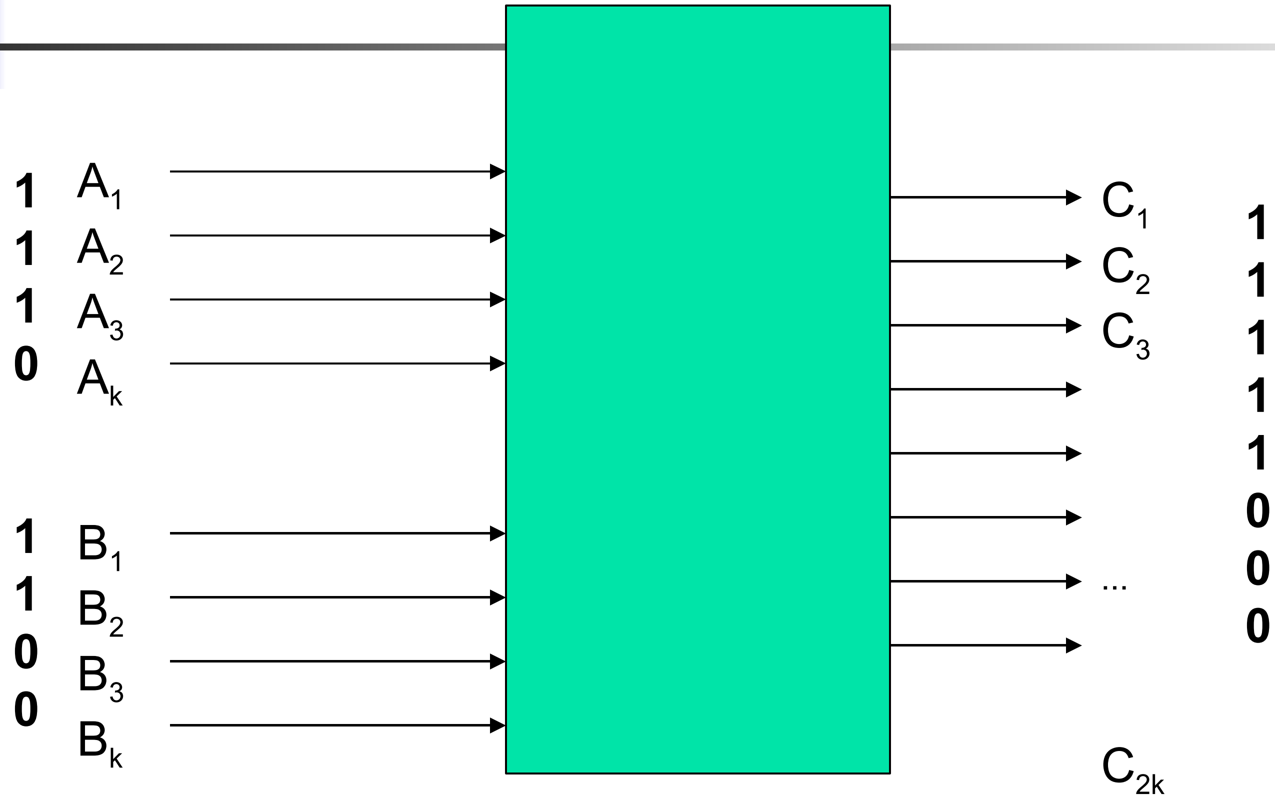
- Key Component of Metrics, Balance Constraints
- Simplistic Counter Representation



Cardinality Constraints

- Pre-Unate Representation
  - Bailleux and Boufkhad, SAT2004
- Represent Cardinalities Of Max Size N with Bit Vectors of N bits
  - Represent ‘k’ by setting first k bits
  - Example:
    - Max Value = 5
    - Express 3 as: 11100

Pre-Unate Totalizer



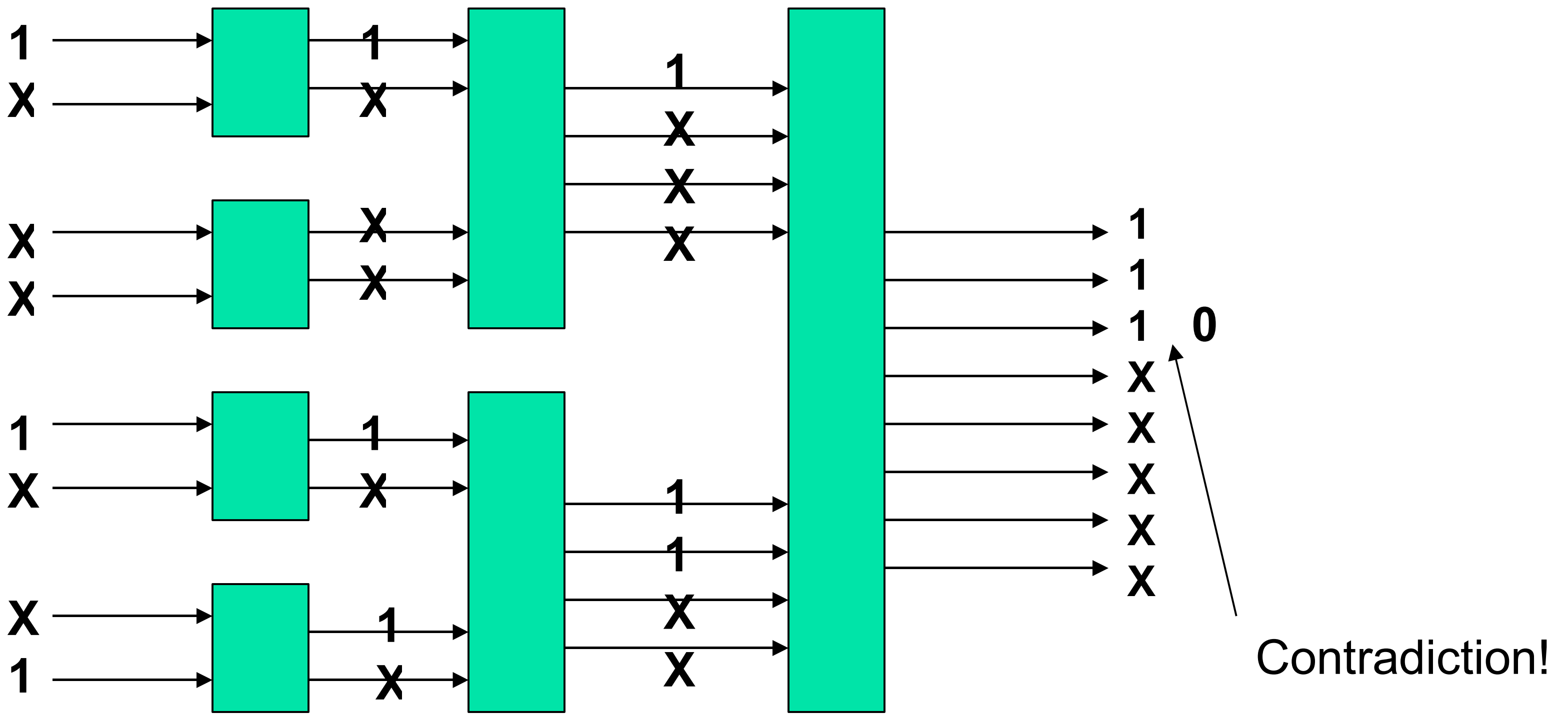
$C_1 = A_1 \vee B_1$

$C_k = A_k \vee B_k \vee (A_1 \wedge B_k\text{-}1) \ldots \vee (A_k \wedge B_{k-1})$

$C_{2k} = A_k \wedge B_k$

Pre-Unate Cardinality Constraints

- Recursively Decompose Cardinalities As Before

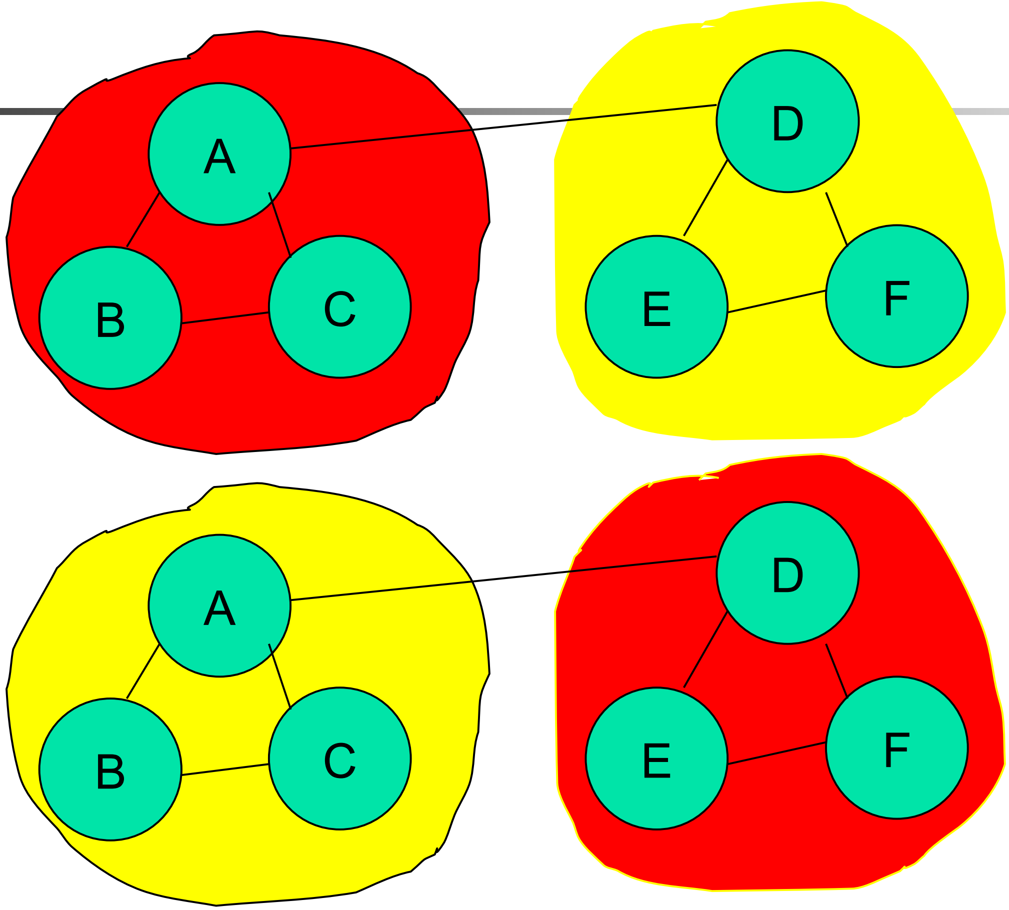


Pre-Unate Speedup

Netlist	Size	k	SAT Runtime (ms) Binary	SAT Runtime (ms) Bailleux & Boufkhad	Speedup
ex4	55	2	16360	2762	5.9
		3	20130	4168	4.8
		4	42877	10960	3.9
		5	93999	15332	6.1
		6	174480	22921	7.6
misex2	97	2	16098	1246	12.9
		3	59514	14344	4.1
		4	105858	14678	7.2
		5	160268	16358	9.8
		6	524047	62035	8.4
5xp1	100	2	73936	8631	8.6
		3	207867	40410	5.1
		4	716010	102163	7.0
		5	Timeout	243489	-
		6	Timeout	652259	-
f51m	114	2	7416	2010.25	3.7
		3	25970	3049	8.5
		4	27988	2694	10.4
		5	135951	9881	13.8
		6	533034	34044	15.7
kirkman	151	2	400696	22714	17.6
		3	742069	75493	9.8
		4	1442291	160343	9.0
		5	Timeout	Timeout	-
		6	Timeout	Timeout	-

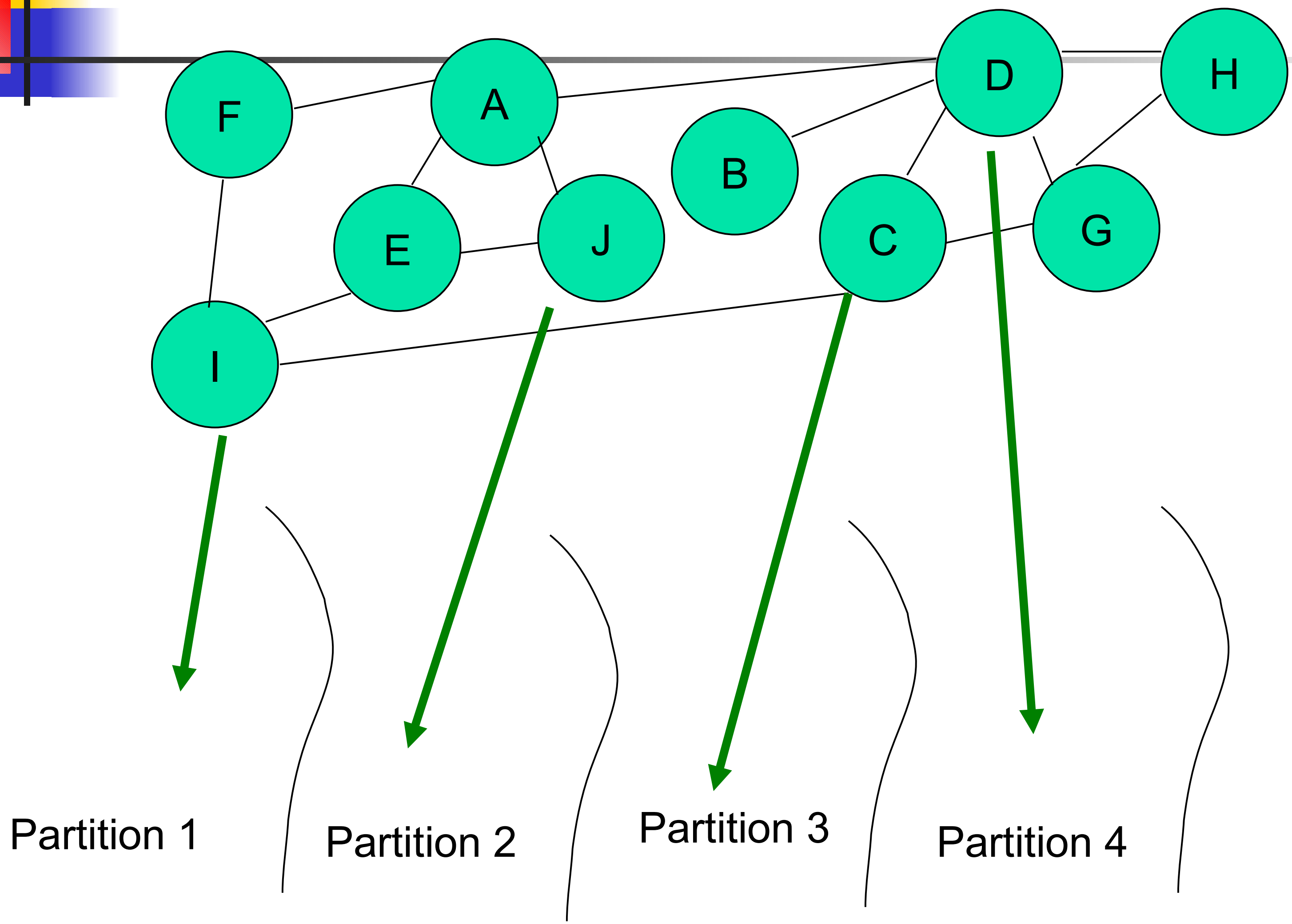
Typical Speedup of About an Order of Magnitude

Symmetry Breaking

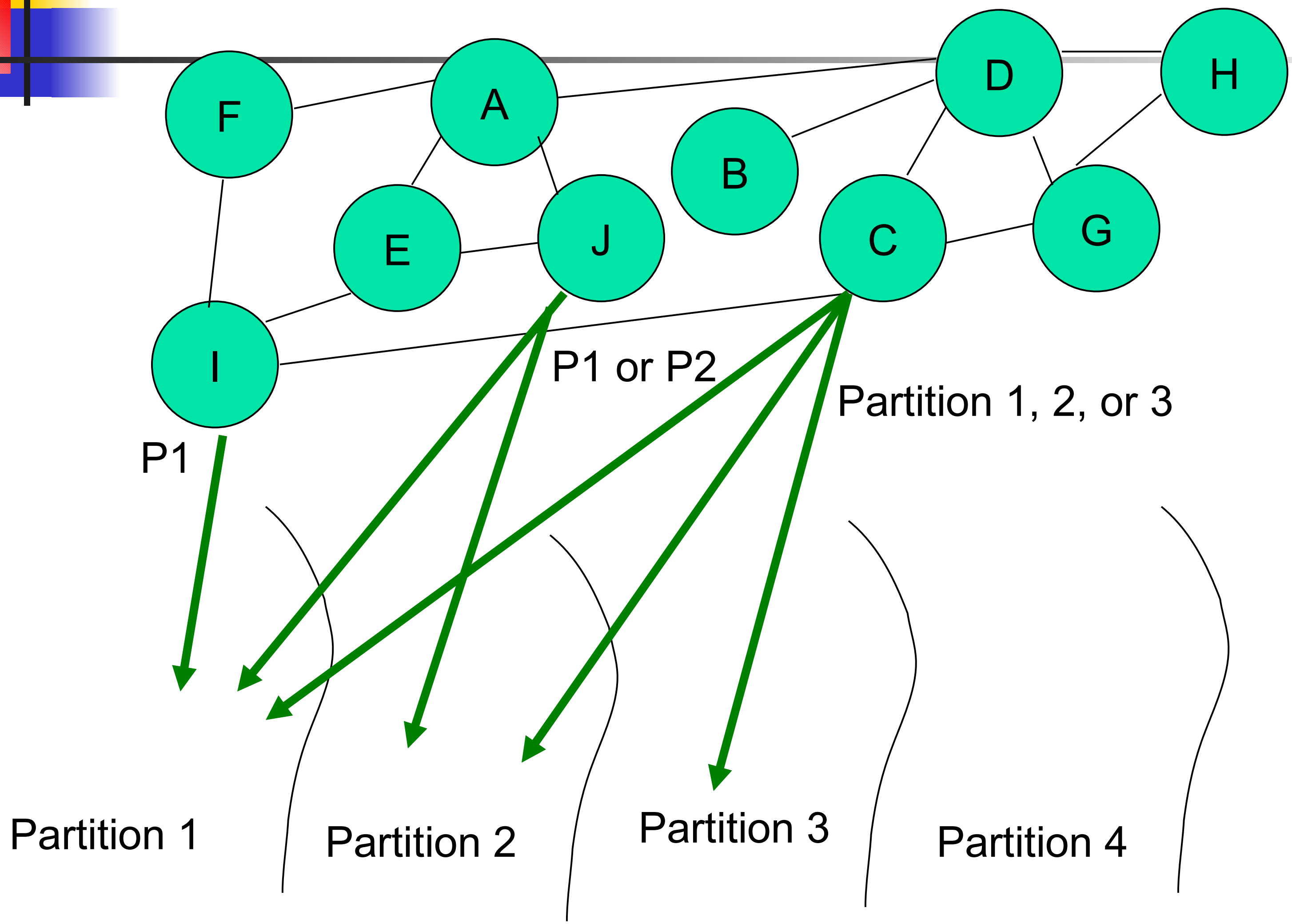


- In General: K! Potential Symmetries

Preassigning an Ordering



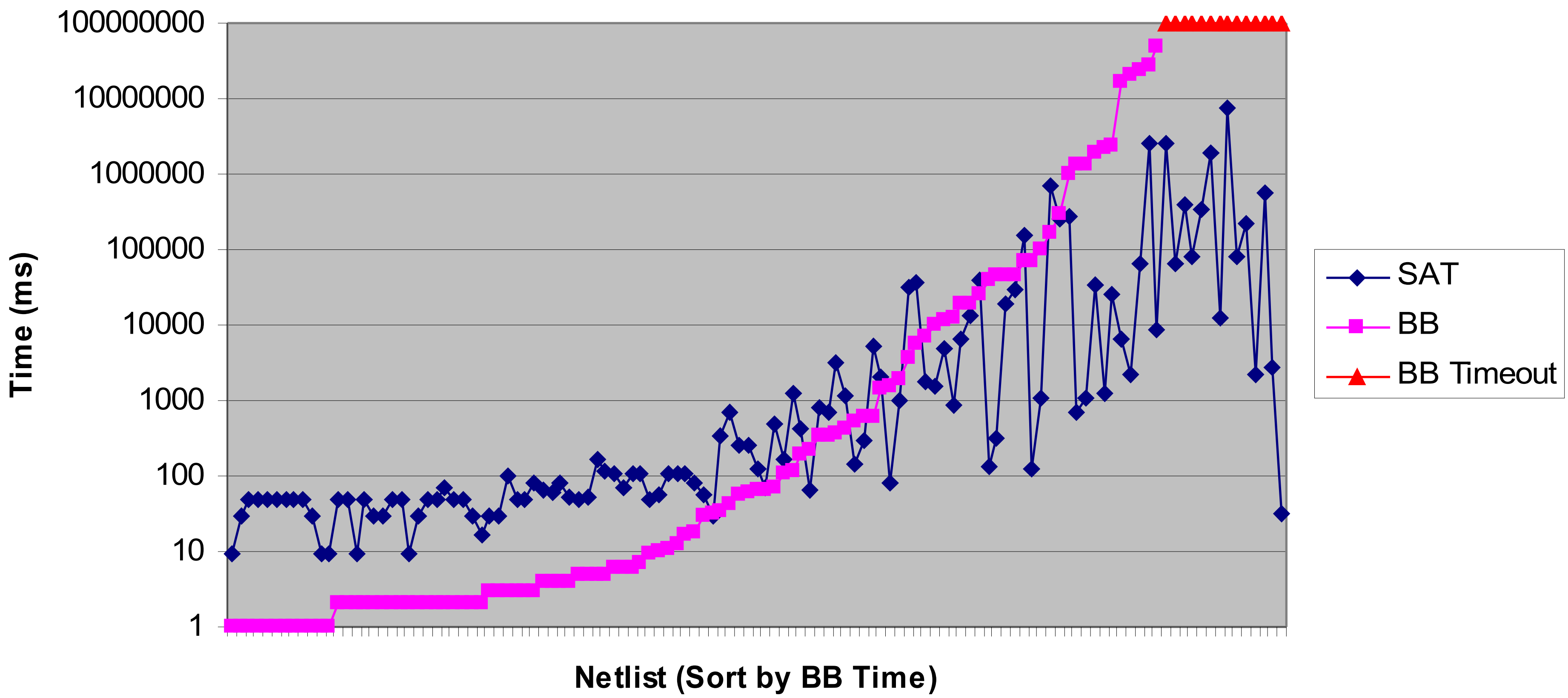
Preassigning a ‘Weak’ Ordering



Runtime: Misex2, 97 Nodes

k	No Symmetry Breaking	Weak Ordering
2	1237	1092
3	15082	4443
4	14740	4814
5	16671	5463
6	64989	23974

Comparison Against BB



Adding Replication

$$AllNodesRepresented = \bigwedge_{A \in Nodes} \left(\bigvee_{0 <= i < nparts} \left(A_k \right) \right)$$

$$PartitionsBalanced = \bigwedge_{0 < i <= nparts} \left(\left(\sum_{A \in Nodes} A_i \right) \leq MaxSize \right)$$

- If MaxSize > Minimum Capacity to Fit Nodes
  - Replication Allowed

Replication Performance

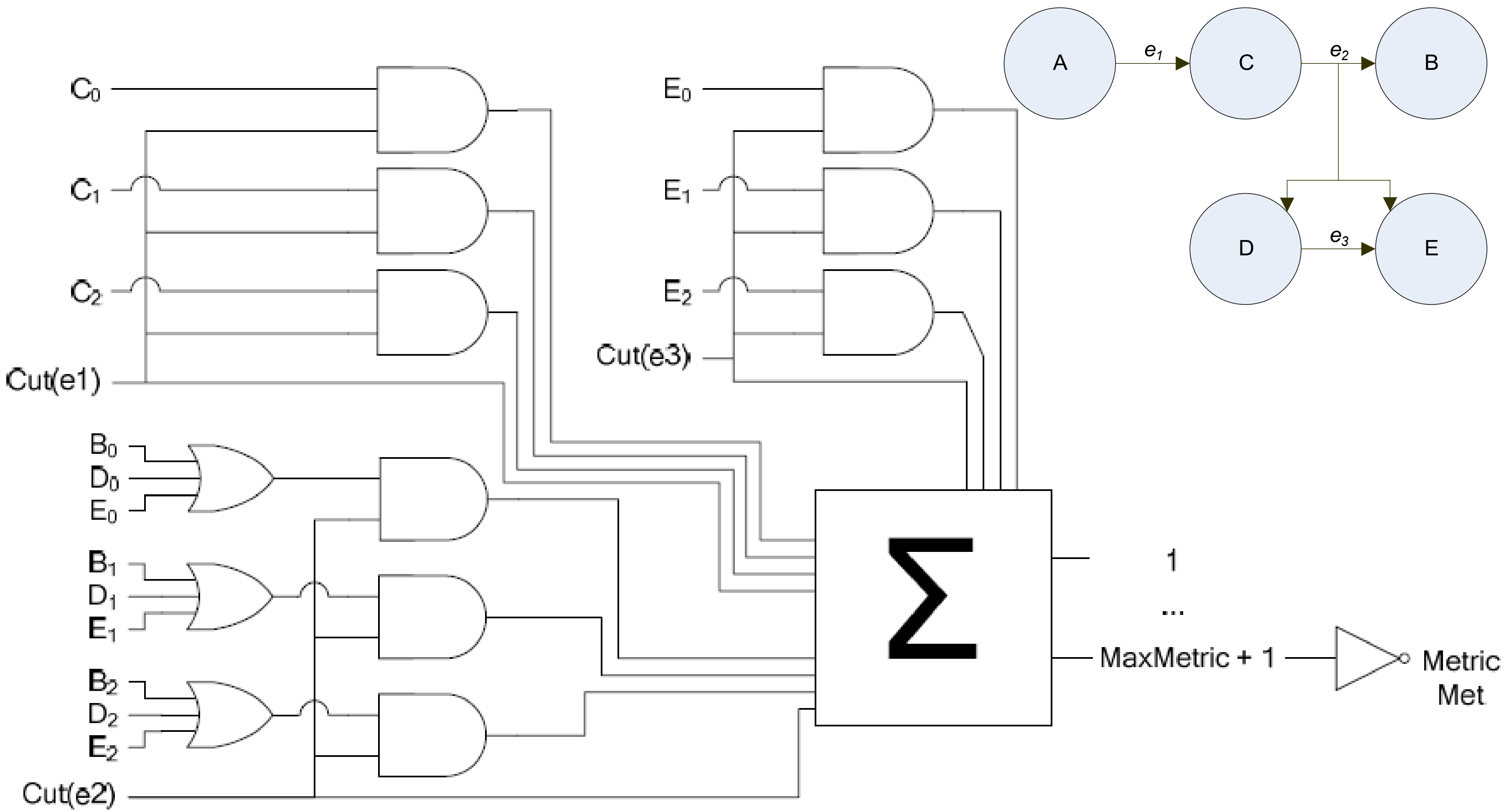
- 2 Partitions, Each 60% of Total Area

Netlist	Size	No Replication		Replication		Slowdown	% Cutsize Impr.
		Cut	ms	Cut	ms		
c8	131	8	1413	8	2228	1.58	0
sao2	133	15	188887	10	7401	0.04	33
s641	135	13	55061	10	16559	0.30	23
s713	137	13	56494	10	12840	0.23	23
mm9b	141	17	344367	15	3348853	9.72	12
C1355	147	16	32097	16	117767	3.67	0
C499	147	16	28155	16	292111	10.38	0
cse	148	18	1522416	11	221276	0.15	39
cht	151	5	170	5	145	0.85	0
kirkman	151	12	11317	9	15006	1.33	25
Avg.						**2.82**	**15.5**

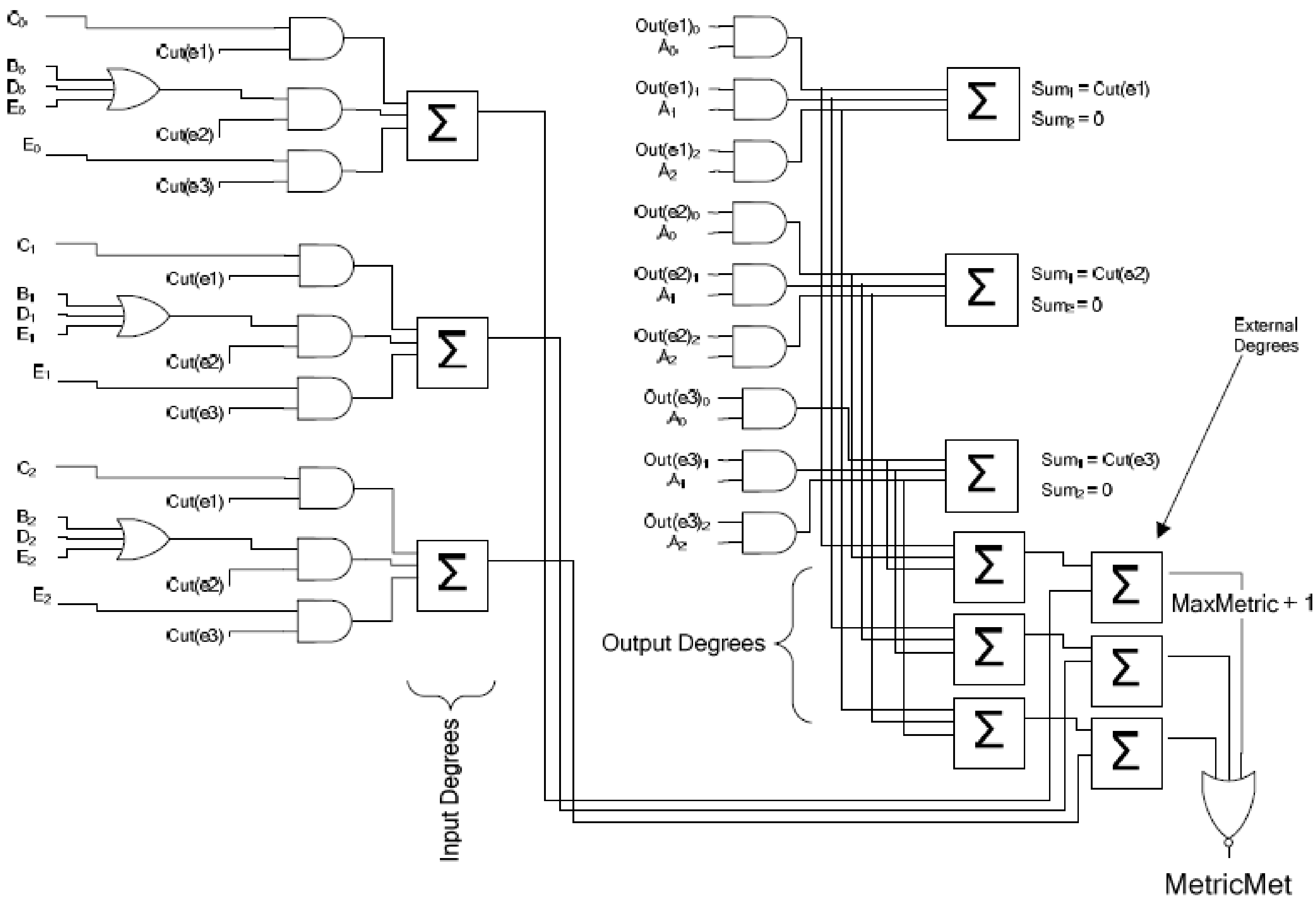
More Realistic Optimization Targets

- Sum of External Degrees
  - Considers Total Number of Pins on Partitions
  - Prefers Solutions Where Cut Edges Interact with Few Partitions
- Maximum Subdomain Degree
  - Consider Maximum IO Into Any Partition
- Practical to Solve ~40 Node Netlists

Sum of External Degrees



Maximum Subdomain Degree



Future Directions

- Higher-K Partitioning
  - More Aggressive Symmetry-Breaking
  - Cost Objectives with Intrinsic Ordering of Partitions
- Hybrid Cost Functions
  - Minimize(Cut Hyperedge Metric) &&
    Maximal Subdomain Metric < x

Conclusions

- Practical SAT Based Formulation of Hypergraph Partitioning
  - Multiway
  - Replication
  - Sophisticated Objective Functions
- Substantial Speedup Over an Existing Optimal Tool for Bipartitioning
- First Published Results for Replication, Multiway, and Realistic Objective Functions

Questions