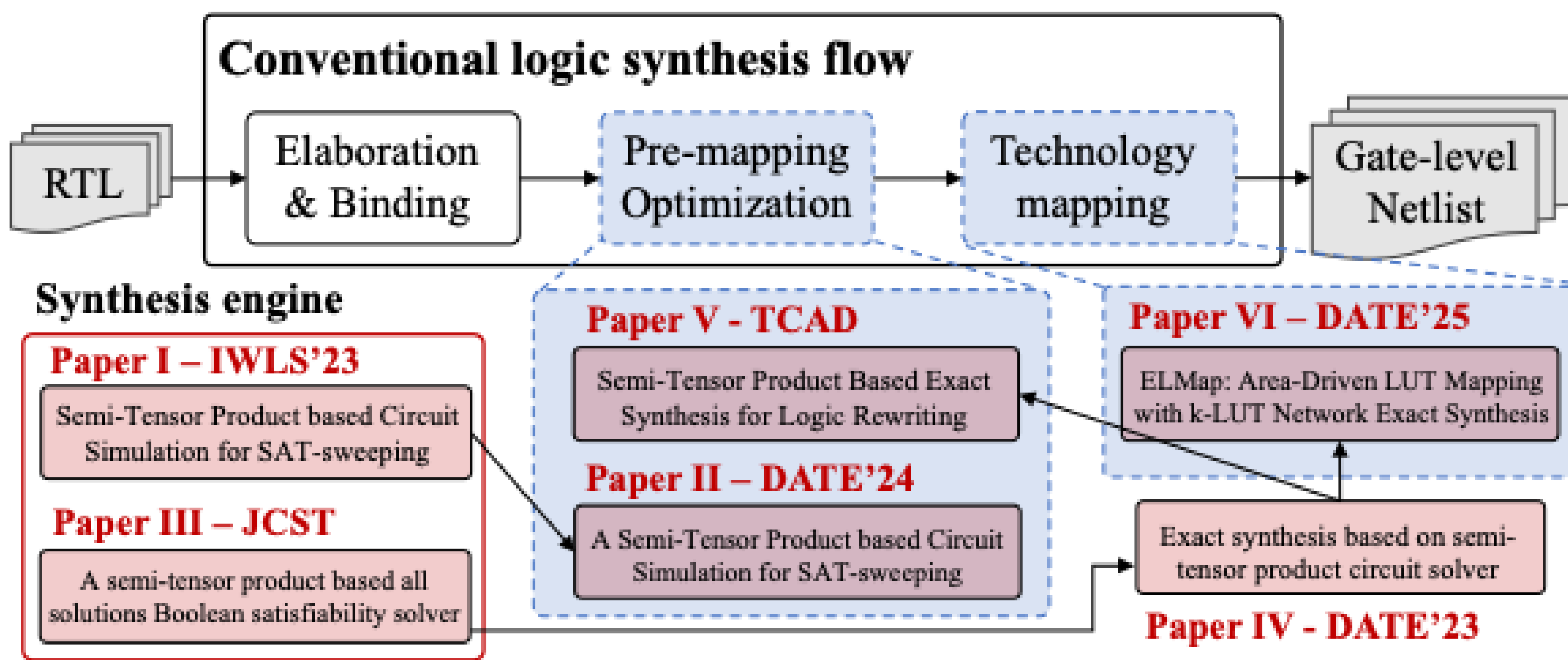


Semi-Tensor Product of Matrices and Its Application in Logic Synthesis

Hongyang Pan¹, Fan Yang¹, Zhufei Chu²

¹ State Key Lab of Integrated Circuits and Systems, Fudan University, ² Ningbo University

Overview



Semi-Tensor Product of Matrices

- Given two matrices $A \in (m, n)$ and $B \in (p, q)$, matrix multiplication require $n = p$. In general, $AB \neq BA$.
- The *semi-tensor product* (STP) of real matrices was proposed by Cheng^[1], which is a generalization of ordinary matrix multiplication and has *quasi-commutativity* under certain conditions.
- We extend the STP of real matrices to *logic representation*, and then some new logic engine under *vector operator* are proposed by using STP-based logical reasoning.

Given two matrices $A \in (m, n)$ and $B \in (p, q)$, the STP of A and B is defined as

$$A \ltimes B = (A \otimes I_{t/n}) \cdot (B \otimes I_{t/p})$$

\otimes : Kronecker product I_t : Identity matrix t : least common multiple of n and p

Example:

$$A = \begin{bmatrix} 1000 \\ 0111 \end{bmatrix} \quad B = \begin{bmatrix} 1101 \\ 0010 \end{bmatrix} \quad (A/B \in (2, 4)) \quad A \ltimes B = A (B \otimes I_2) = \begin{bmatrix} 1000 & 1010 & 0010 \\ 0111 & 0101 & 0001 \end{bmatrix} = \begin{bmatrix} 1010 & 0010 \\ 0101 & 1101 \end{bmatrix}$$

$$A = \begin{bmatrix} 10 & 00 \\ 01 & 11 \end{bmatrix} \quad B = \begin{bmatrix} 11 & 01 \\ 00 & 10 \end{bmatrix} \quad A \ltimes B = \begin{bmatrix} 1010 & 0010 \\ 0101 & 1101 \end{bmatrix}$$

For a Boolean variable, we can denote $T = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ and $F = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$

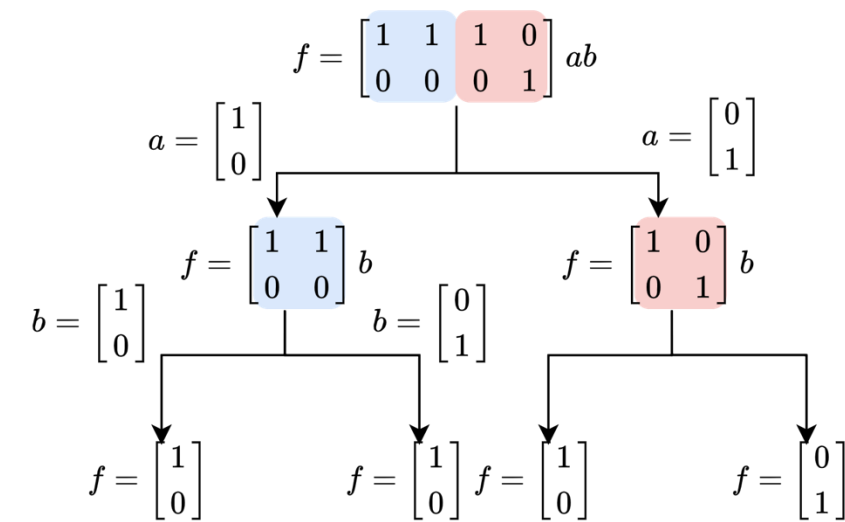
For a Binary Boolean operator, the operation can be represented by a structural matrix

- For example, the conjunction \wedge (AND) is represented as $M_c = M_\wedge = \begin{bmatrix} 1000 \\ 0111 \end{bmatrix}$
 - The first row is actually the **truth table (TT)** of the operator, the second row is the inversion of the first row
- $$M_c = M_\wedge = \begin{bmatrix} 1000 \\ 0111 \end{bmatrix} \text{ (AND)} \quad M_d = M_\vee = \begin{bmatrix} 1110 \\ 0001 \end{bmatrix} \text{ (OR)}$$
- $$M_i = M_\rightarrow = \begin{bmatrix} 1011 \\ 0100 \end{bmatrix} \text{ (Implication)} \quad M_e = M_\leftrightarrow = \begin{bmatrix} 1001 \\ 0110 \end{bmatrix} \text{ (Equivalence)}$$

a	b	f
0	0	0
0	1	0
1	0	0
1	1	1

From logic expression to STP computation

- Boolean variables a and b , logic expression $f = a + b$
- The corresponding STP form is $f = a + b = M_d a b = \begin{bmatrix} 1110 \\ 0001 \end{bmatrix} a b$
- If σ is a binary operator, then $P \sigma Q = M_\sigma P Q$
- M_σ is a structural matrix of σ (R-ary), then $M_\sigma \in (2 \times 2^R)$



Power-reducing: $M_R = \begin{bmatrix} 10 \\ 00 \\ 00 \\ 01 \end{bmatrix}$ Variable swapping: $M_w = \begin{bmatrix} 1000 \\ 0010 \\ 0100 \\ 0001 \end{bmatrix}$

$A^2 = M_R A$ $A B = M_w B A$

Matrix swapping: $A M = (I_2 \otimes M) A$

Person A said that person B is a liar, person B said person C is a liar, and person C said that both persons A and B are liars. Who is the liar?

1. Define Boolean variables:

A: person A is honest
 $\neg A$: person A is a liar
 B / C similarly

2. Operations

The logic expression of the statement is

$$(A \leftrightarrow \neg B) \wedge (B \leftrightarrow \neg C) \wedge (C \leftrightarrow \neg A \wedge \neg B)$$

3. Matrix Form

$$M_c^2 M_e A M_n B M_e B M_n C M_e C M_n A M_n B$$

4. Canonical Form

$$L(A, B, C) = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & 1 \end{bmatrix} A B C$$

We have only one satisfying solutions where L is true if

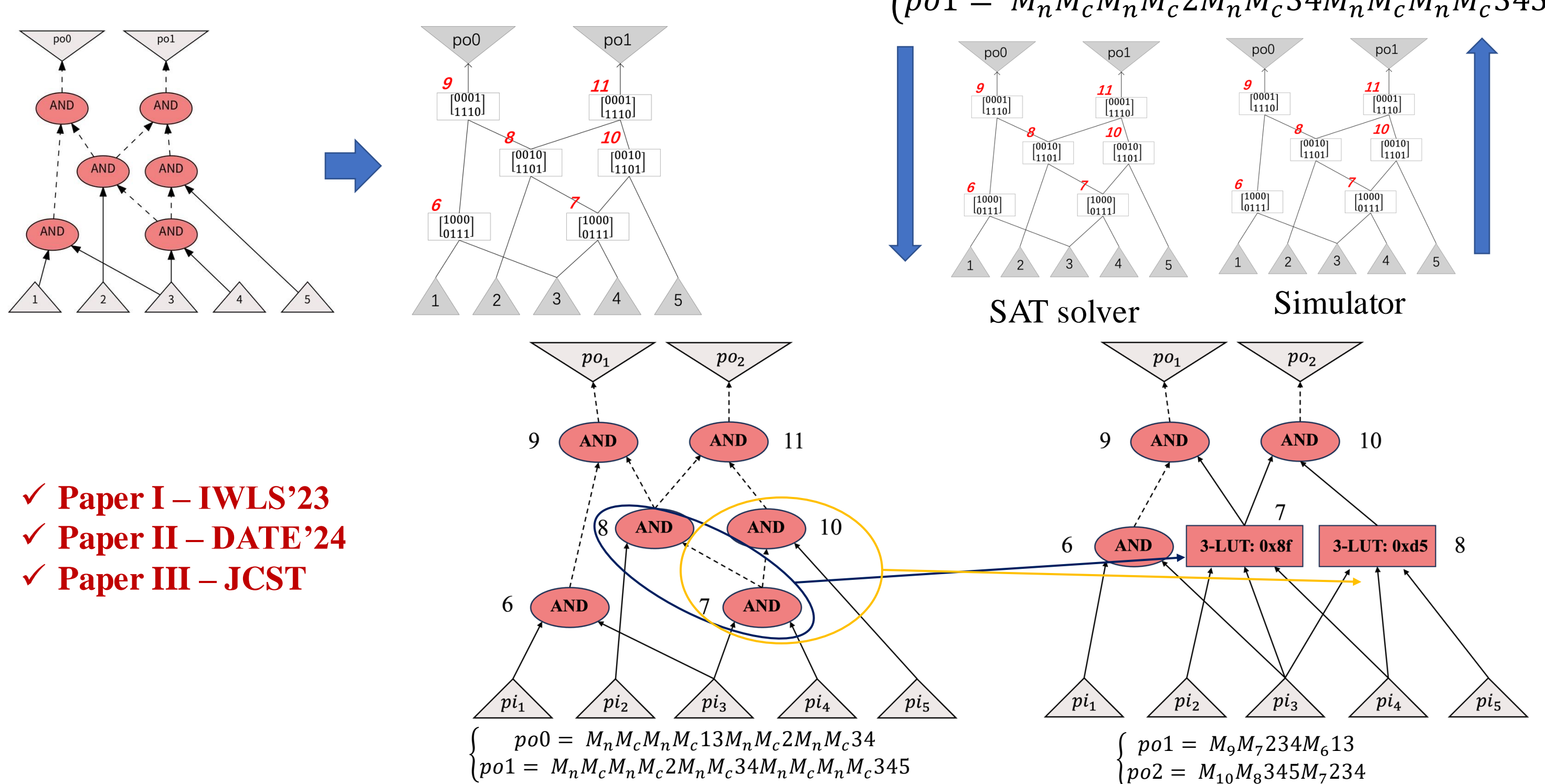
$$A = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, B = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, C = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

5. Solution

So the final solution is that A and C are liars.

Synthesis Engines

Matrices are seen as primitives in the logic network



- ✓ Paper I - IWLS'23
- ✓ Paper II - DATE'24
- ✓ Paper III - JCST

STP-based Exact Synthesis

SAT-based exact synthesis essentially solves the two tasks

- Find logic network topology (DAG)
- Assign Boolean operators to vertices

STP-based exact synthesis is a proposal to mitigate or avoid the task 1

- providing additional constraints
- Convert SAT solving to matrix computation

- ✓ Paper IV - DATE'23
- ✓ Paper V - TCAD
- ✓ Paper VI - DATE'25

$$XXXX M_x = \begin{bmatrix} 1011111 \\ 0100000 \end{bmatrix} \downarrow$$

$$XXXX = \begin{bmatrix} 1011 \\ 0100 \end{bmatrix} M_x = \begin{bmatrix} 1000 \\ 0111 \end{bmatrix} \text{ or}$$

$$XXXX = \begin{bmatrix} 1110 \\ 0001 \end{bmatrix} M_x = \begin{bmatrix} 0111 \\ 1000 \end{bmatrix}$$

$$XXXX I_2 \otimes M_x = \begin{bmatrix} 1000 & 1111 \\ 0111 & 0000 \end{bmatrix} \downarrow$$

$$XXXX = \begin{bmatrix} 1011 \\ 0100 \end{bmatrix} M_x = \begin{bmatrix} 1000 \\ 0111 \end{bmatrix} \text{ or}$$

$$XXXX = \begin{bmatrix} 1111 \\ 1000 \end{bmatrix} M_x = \begin{bmatrix} 0111 \\ 1000 \end{bmatrix}$$

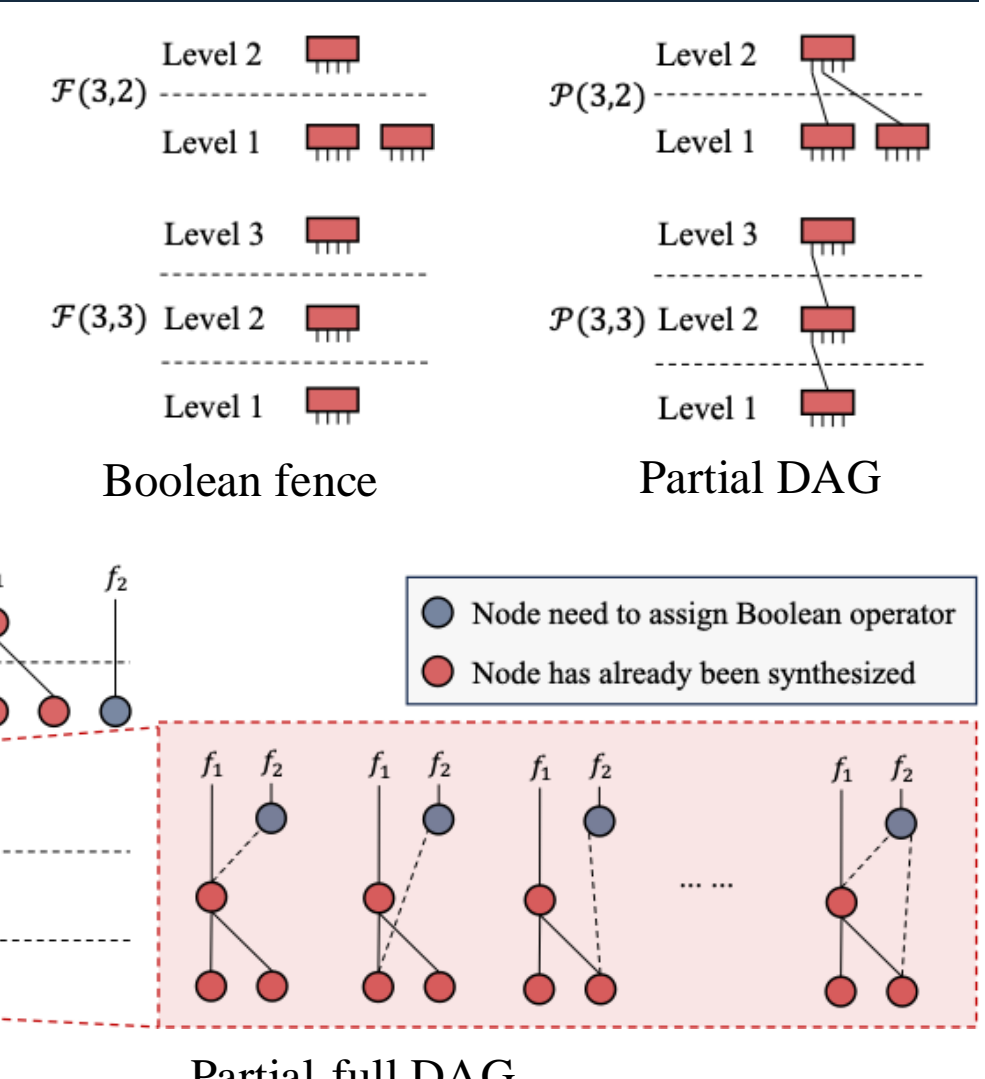
$$\text{Power-reducing}(M_R) = \begin{bmatrix} 10 \\ 00 \\ 00 \\ 01 \end{bmatrix}$$

$$XXXX M_R = \begin{bmatrix} 1000 & 1111 \\ 0111 & 0000 \end{bmatrix} \downarrow$$

$$XXXX = \begin{bmatrix} 1000 & xxxx & xxxx & 1111 \\ 0111 & xxxx & xxxx & 0000 \end{bmatrix}$$

$$XXXX I_2 \otimes M_R = \begin{bmatrix} 1000 & 1111 \\ 0111 & 0000 \end{bmatrix} \downarrow$$

$$XXXX = \begin{bmatrix} 10xxxx00 & 11xxxx11 \\ 01xxxx11 & 00xxxx00 \end{bmatrix}$$



$$\text{Variable swapping}(M_w) = \begin{bmatrix} 1000 \\ 0010 \\ 0100 \\ 0001 \end{bmatrix}$$

$$XXXX M_w = \begin{bmatrix} 1000 & 1111 \\ 0111 & 0000 \end{bmatrix}$$

$$XXXX = \begin{bmatrix} 1011 & 0011 \\ 0100 & 1100 \end{bmatrix}$$

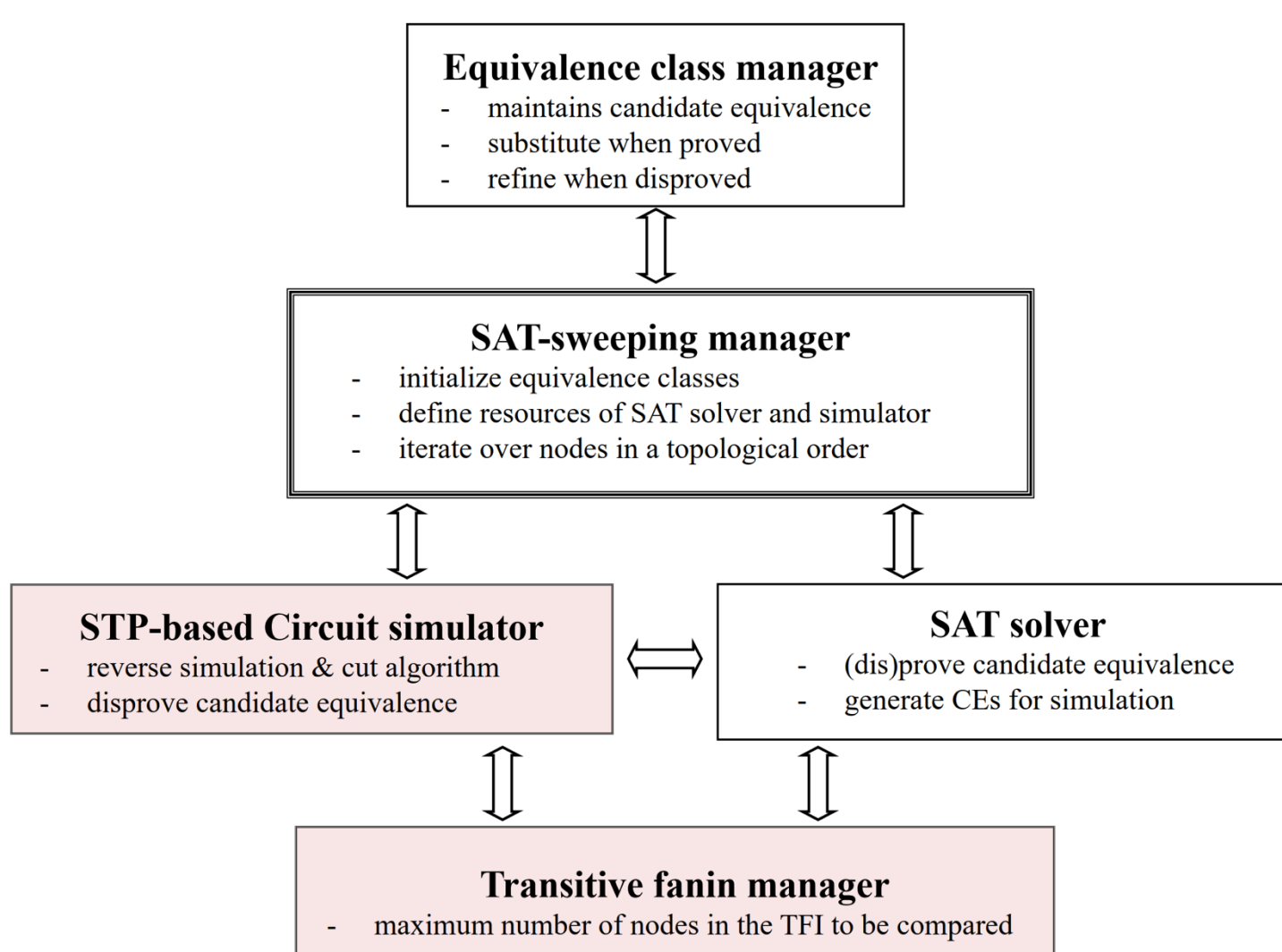
$$XXXX I_2 \otimes M_w = \begin{bmatrix} 1000 & 1111 \\ 0111 & 0000 \end{bmatrix}$$

$$XXXX = \begin{bmatrix} 1000 & 1111 \\ 0111 & 0000 \end{bmatrix}$$

STP-based exact synthesis (matrix factorization)

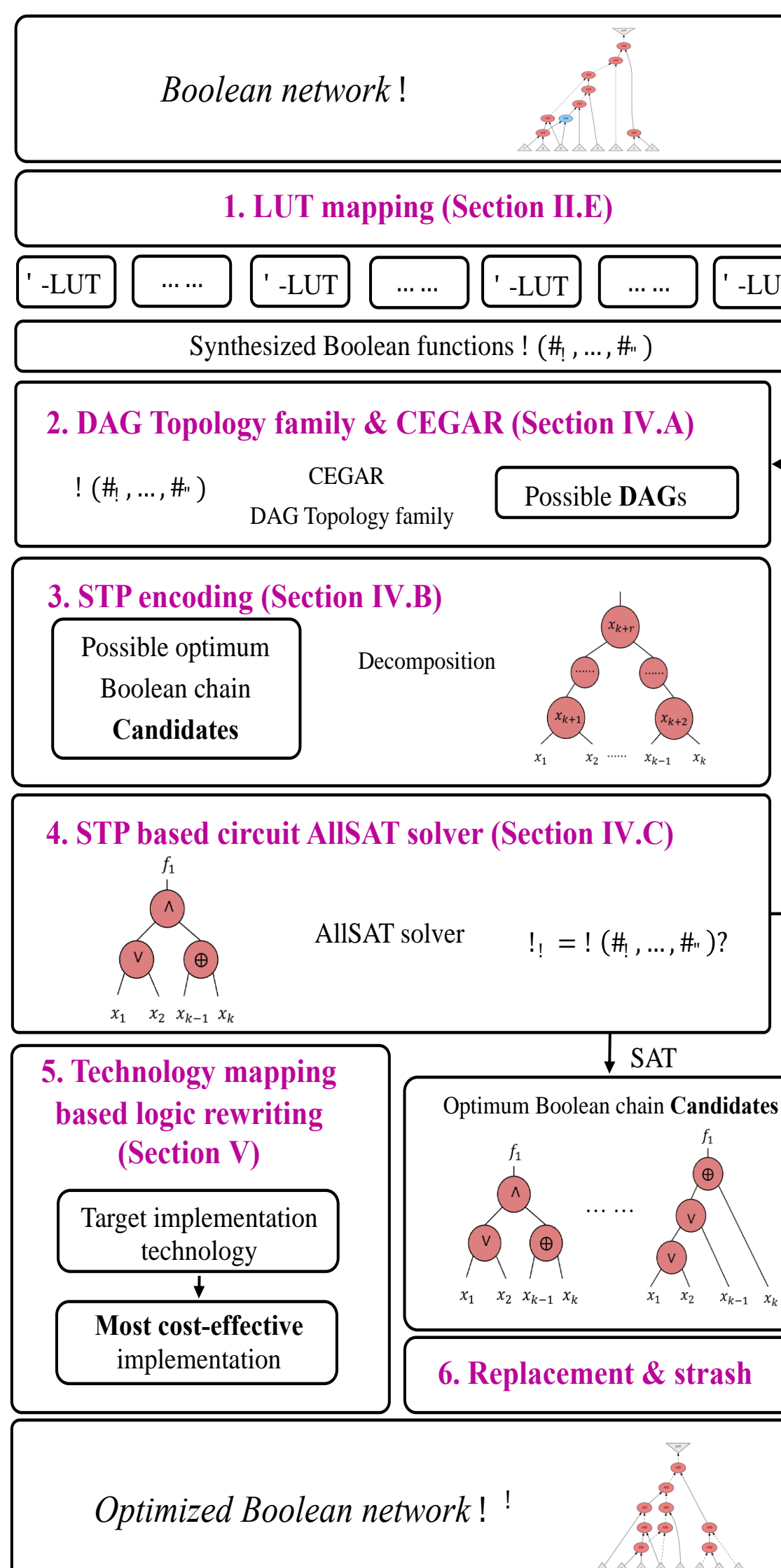
Applications

1. SAT-sweeping (Paper II - DATE'23)



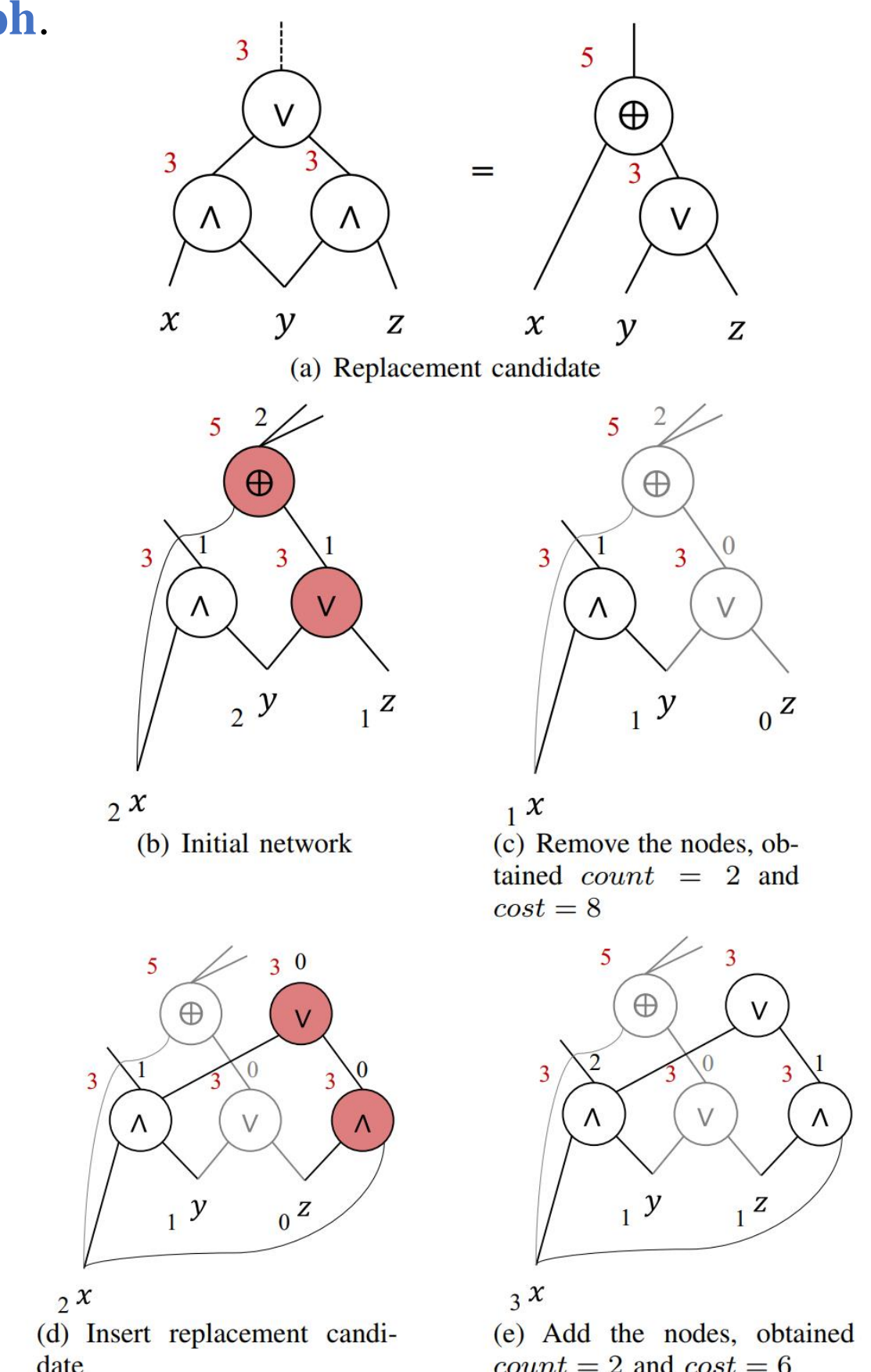
- Simulation patterns are derived from **SAT-guided initial pattern generation**.
- Sort the list of gates to be processed by SAT-sweeping in **inverse topological order**.
- Process the gates that have been sorted. The current gate is referred to as [candidate], as it is a candidate gate for **deletion and replacement** with a topological preceding gate.
- Translated into SAT solving problem.
unDET: the candidate is marked as don't touch, which provides maximum runtime speedup.
unSAT: substitute the nodes.
SAT: we obtain the CE from the solver, propagate the CE through the STP-based simulator, and then refine the equivalence classes accordingly.

2. Logic rewriting (Paper V - TCAD)



- Potential network replacements for each k-LUT are **computed online** using STP-based exact synthesis.
- We also compute **the cost of every nodes by reading a target implementation technology**.
- Replacements that reduce the overall network size are stored in a **conflict graph**.

A maximal set of **non-conflicting replacements** can be selected from the conflict graph, which can **maximize the achieved gain and minimize the cost of implementation**, and applied to rewrite the Boolean network.



3. Logic rewriting (Paper VI - DATE'25)

Algorithm 1: Area-driven LUT mapping algorithm.

Input: Input network A, LUT input size k , number of cut choices n
 Output: LUT network L

```

1 foreach  $v \in A$  in topological order do
2    $J_k, \dots, J_{k+n} \leftarrow \text{cut\_enu}(k, \dots, k+n)$ 
3    $\text{best\_subLUT}_v \leftarrow \text{exact}(J_k, \dots, J_{k+n})$ 
4 end
5 foreach  $v \in A$  in reverse topological order do
6    $\text{best\_LUT} \leftarrow \text{compare\_area}(\text{best\_subLUT}_v)$ 
7    $L_v \leftarrow \text{LUT\_map}(\text{best\_LUT}, A_v)$ 
8   visited(best_LUT)
9 end
    
```

ELMap consists of two main components: area-driven LUT mapping (ALM), exact synthesis of k-LUT network (ELN).

- ALM generates multiple decomposed subcircuits and optimizes the area of the LUT netlist by **dynamically selecting** the optimal sub-LUT.
- ELN uses exact k-LUT synthesis to store multiple **optimal sub-LUT netlists** for each node.

