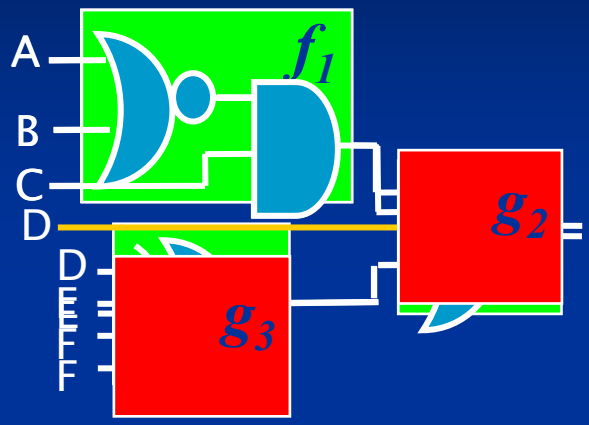
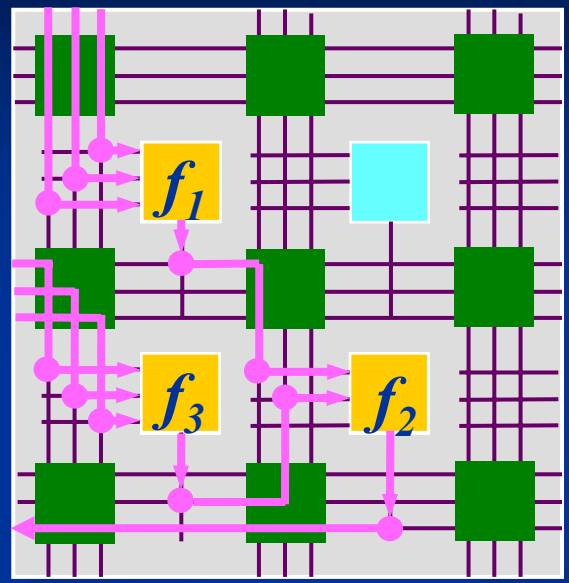
A Fast SPFD-based Rewiring Technique

Pongstorn Maidee and Kia Bazargan
University of Minnesota
USA

Rewiring : What is it & Why Use it?





1) Synthesis → 2) Mapping → 3) Placement → 4) Routing

- If optimization goal is not met at placement,
  - **Feed info back to earlier stage (slow, may not converge)**
  - **Restructure circuit at this stage (circuit rewiring)**

Rewiring Techniques

Methods	How?	speed	quality	FPGA?
ATPG	Check redundancy	fast	OK	Yes
Graph-based	Match subcircuit to prototype	Very fast	OK	No
SPFD	Better way to describe circuit functionality	Slow	Best	Yes
Symbolic	Boolean reasoning	Slow	OK	yes

SPFD-based Rewiring Engine: Problems & Solutions

- Conventional SPFD-based rewiring
  - Use BDD to represent SPFD
    ⇒ BDD size may be large
    ⇒ Runtime/Memory bottleneck.
- Propose:
  - SAT-based SPFD-based rewiring engine ⇒ fast
- Contributions:
  - Use SAT for SPFD-based rewiring ⇒ use less memory
  - Use auxiliary circuits to find SPFD propagation paths
  - Devise a way to use one SAT run to see if a rewiring is valid.

Outline

- Binary functions and their representations.
- Definition of SPFD.
- Why SPFD?
- SPFD computation
- SPFD-based rewiring
- Efficient SPFD rewiring

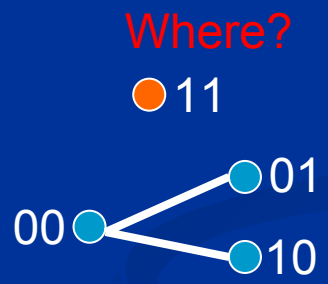
Binary Functions = Bipartite Graphs

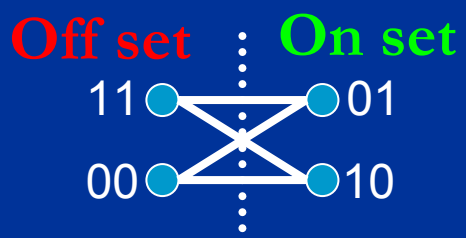
- Completely Specified Function

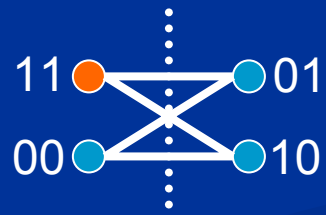
x	y	f
0	0	0
0	1	1
1	0	1
1	1	0

- Incompletely Specified Function (ISF)

x	y	f
0	0	0
0	1	1
1	0	1
1	1	*



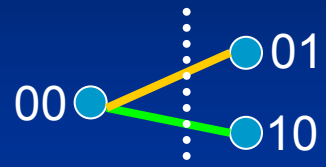




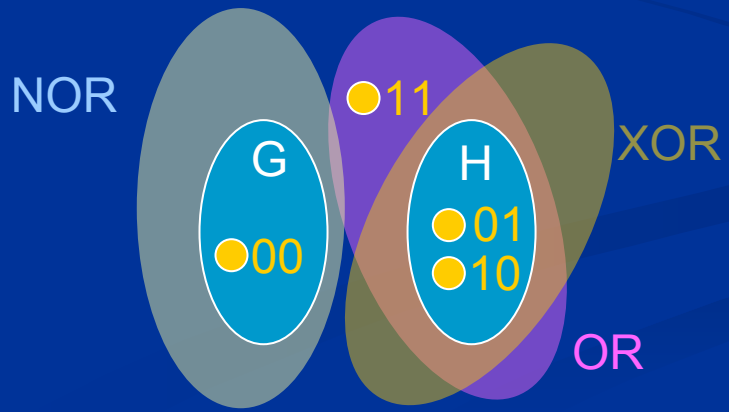
A function can be represented as a bipartite graph.

Set of Pairs of Functions to be Distinguished (SPFD)

- List each edge of a bipartite graph :
  {(00,01),(00,10)}



- If each node is a function : the list becomes
  SPFD = $\{(g_1,h_1),(g_2,h_2),\ldots,(g_n,h_n)\}$.
- f satisfies a pair of functions (g,h),
  if f includes either g or h BUT not both.
- f satisfies SPFD, if f satisfies each pair in SPFD.

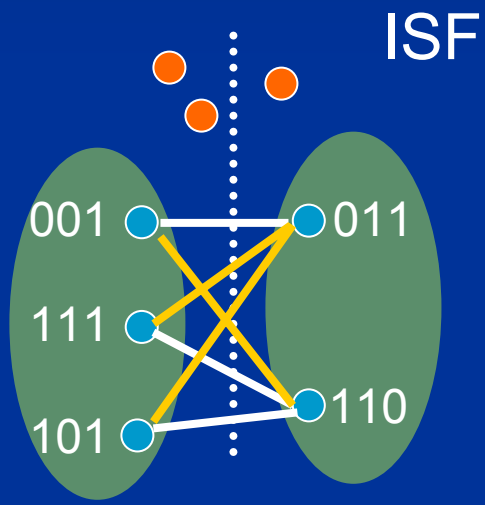


Why SPFD?

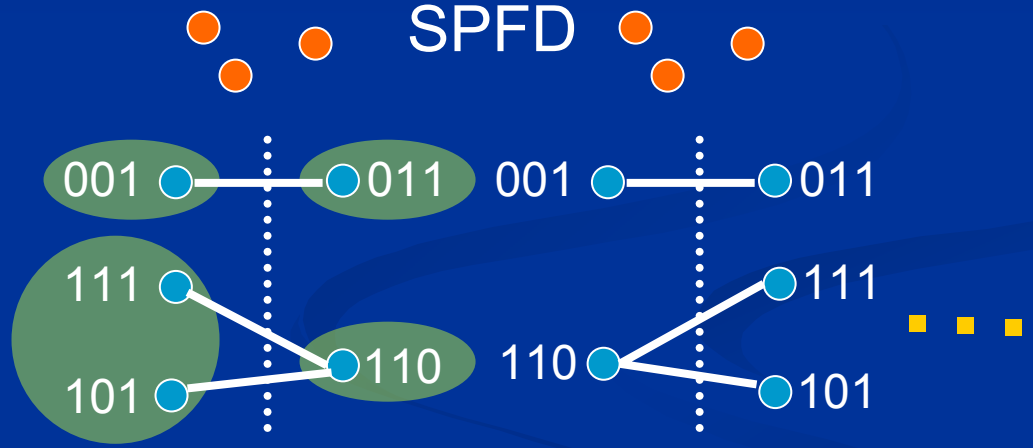
SPFD is more flexible than ISF.

⇓

Rewiring based on SPFD is more powerful than those based on ISF.



$2^4 = 16$ functions



$2^3 * 2^2 = 32$ functions

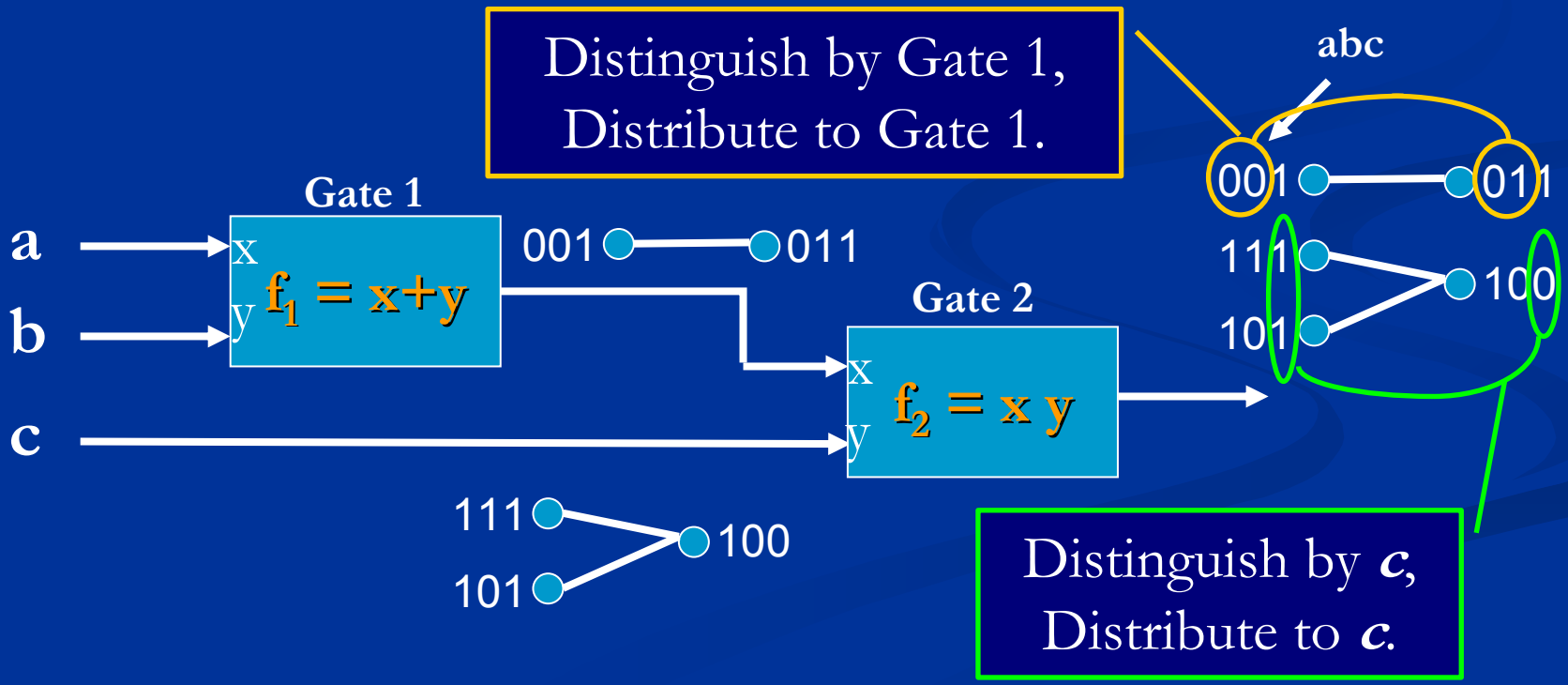
As a requirement for a node to be a valid Boolean function:

- ISF: combine both groups and implicitly add extra edges
- SPFD: no need to combine groups.

A Boolean function at a node (complete bipartite) must be superset of the SPFD

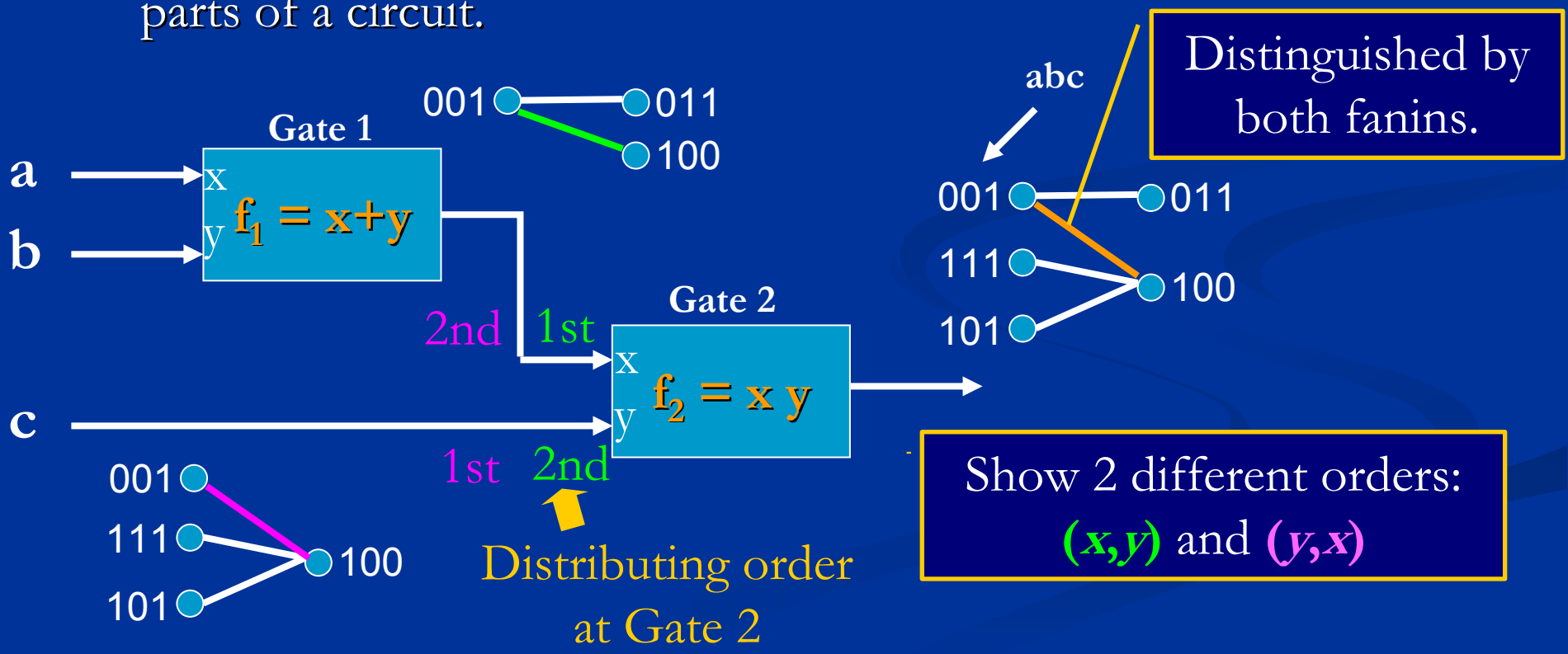
SPFD Computation

- No synthesis tool generates SPFD (as of now)
- Need to compute SPFD on a synthesized network
- SPFD at a PO ***x*** is (f(***x***) , inv(f(***x***)))
- Propagate SPFD backwards



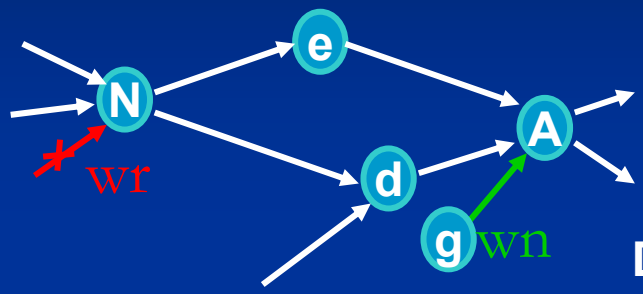
Distributing Order

- What if more than one fanin distinguish a pair ?
- Convention : Distribute to the first fanin in a distributing order.
- Important ? Distributing order determines # pairs flowing in parts of a circuit.



SPFD-based Rewiring

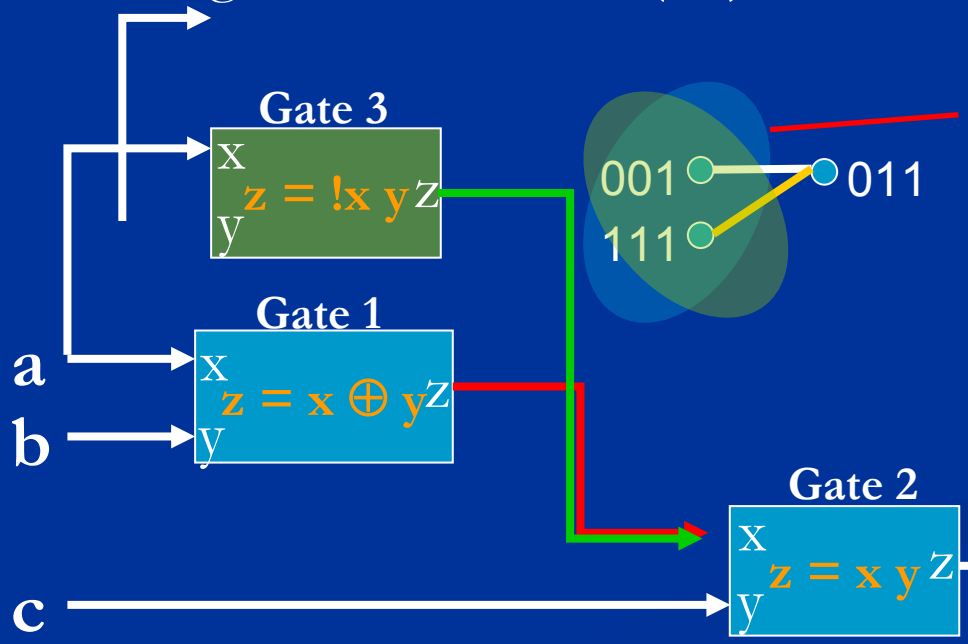
- Where to add a new wire ?



Effect of removing **wr** pass node A.
Add **wn** at A may compensate for that.

Dominator of N = { N , A }

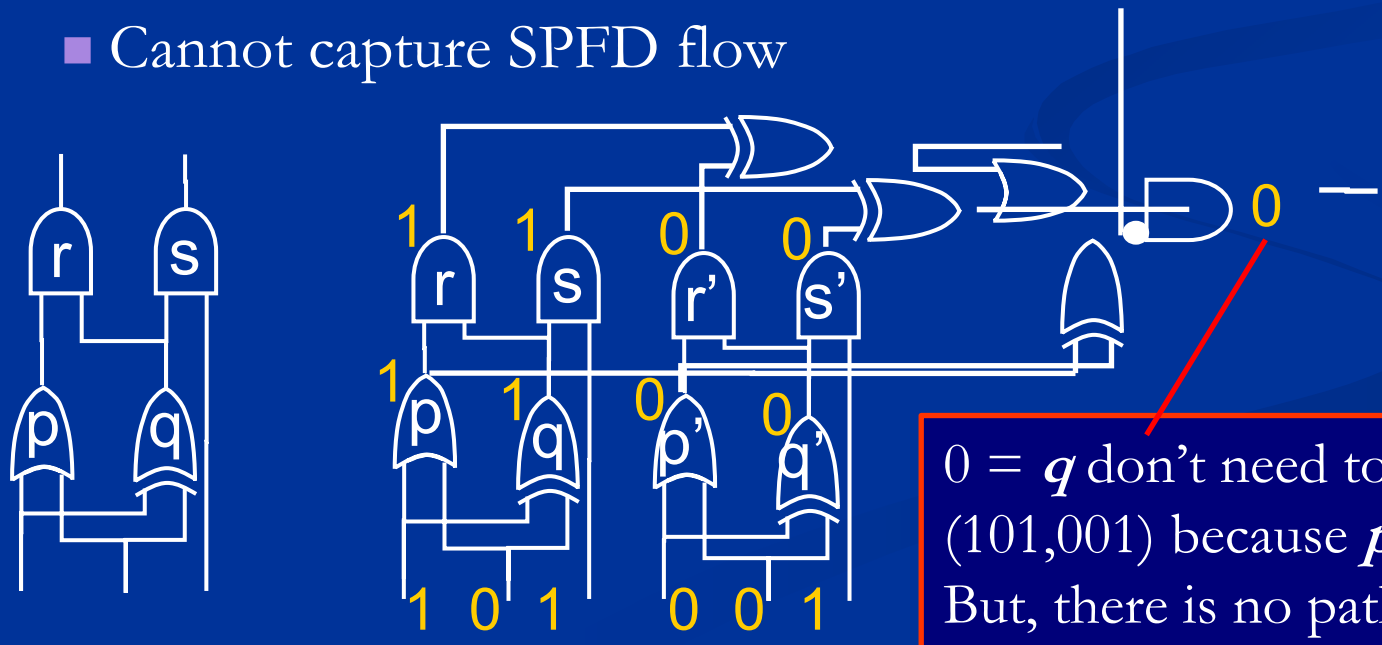
- Rewiring is valid if SPFD(**wr**) ⊆ SPFD(sr(**wn**))



Satisfied by both gates 1 and 3
⇒ replace **(1,2)** with **(3,2)**

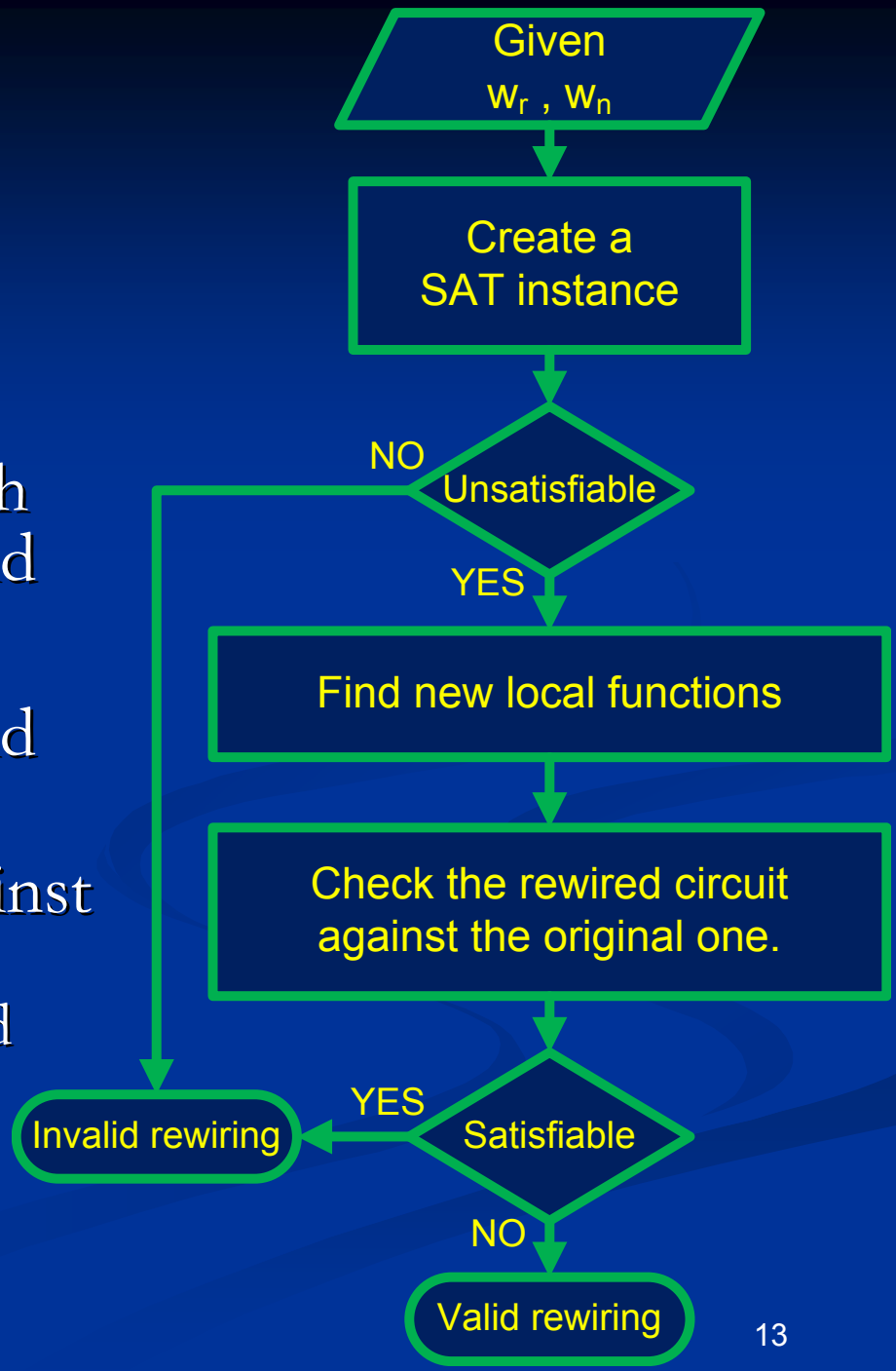
Previous work

- Conventional SPFD-based rewiring
  - Use BDD to represent SPFD ⇒ BDD size becomes bottleneck.
- Use SAT for circuit restructuring
  - Build new node producing the same care minterms as the old node.
  - Don't care is not as flexible as SPFD ⇒ inferior to SPFD-based.
- Previous use of SAT to compute SPFD
  - Cannot capture SPFD flow

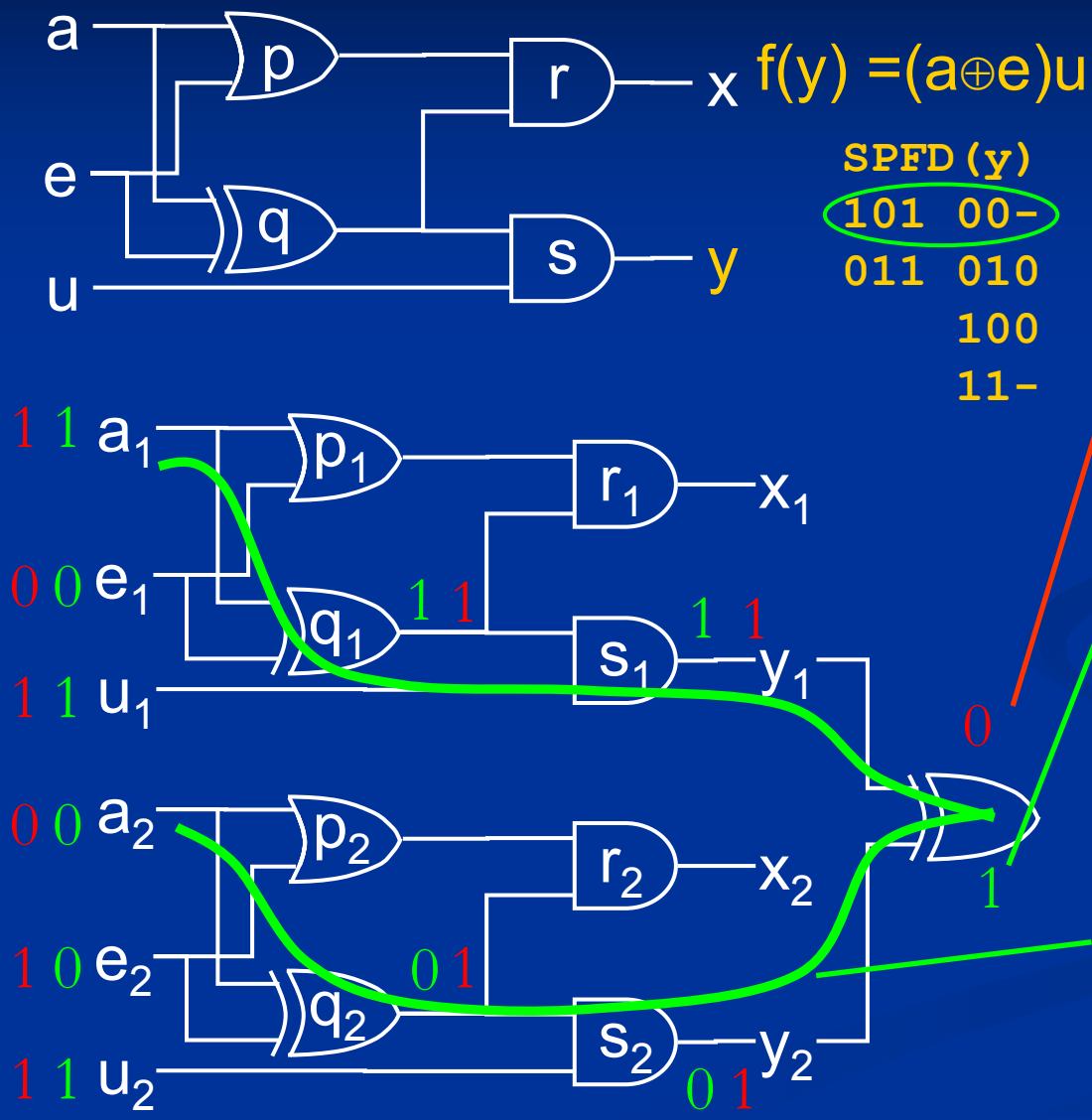


Our Efficient SPFD Rewiring

- For each candidate wire
  - Build rewiring instance such that it is unsatisfiable if valid
  - Check rewiring validity
- Find new function for the valid rewiring
- Check the rewiring circuit against the original circuit.
  (extra pairs of minterms introduced inside the circuit are not captured)



The Pair Distinguishable ?

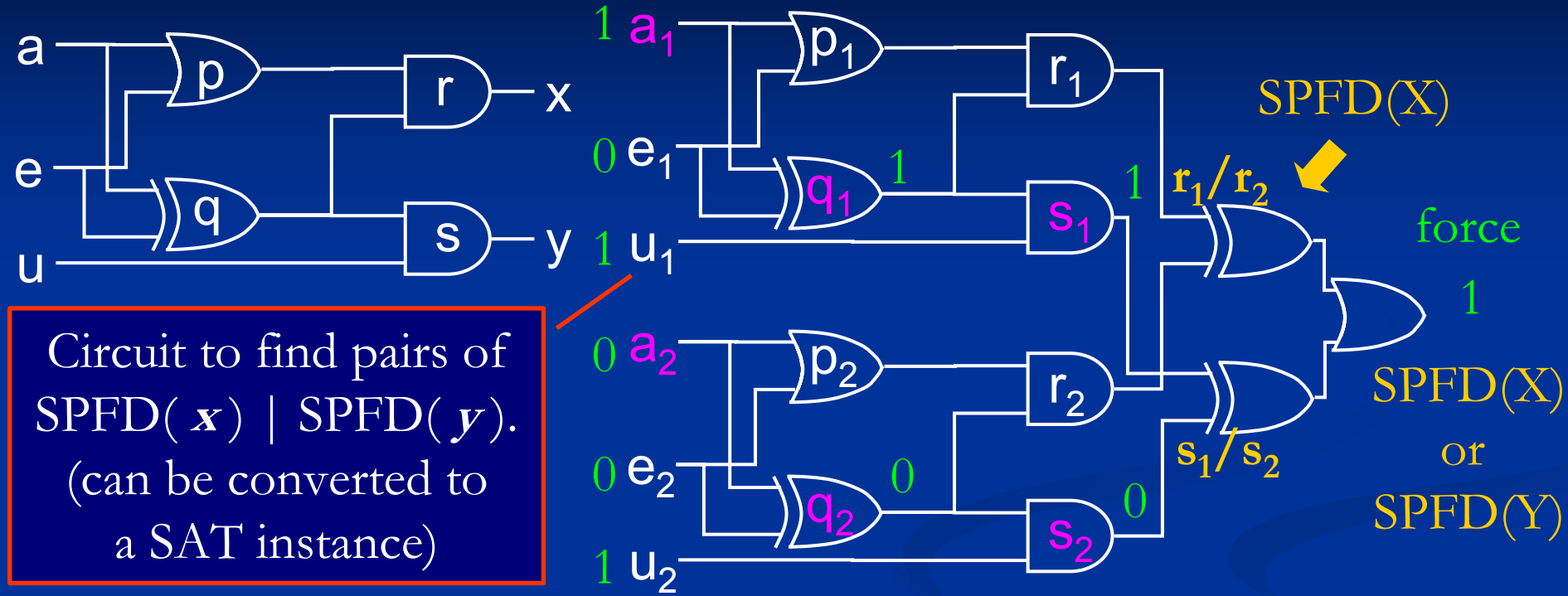


If a pair ∉ SPFD(y), output is **0**.

If a pair ∈ SPFD(y), output is **1**.

If a pair ∈ SPFD(y), ∃ a distinguishable path.

Find Distinguishable Pairs with SAT

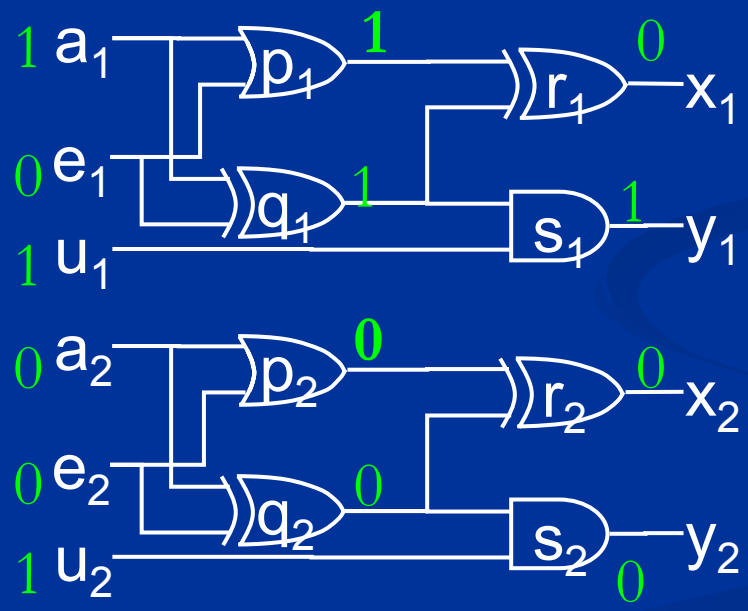


Circuit to find pairs of SPFD(x) | SPFD(y). (can be converted to a SAT instance)

- A pair ∈ SPFD(x) | SPFD(y) if output is 1 ⇒ add forcing clause
  Forcing clause := become 0 if NOT match
  Eg. a_1,e_1,u_1,a_2,e_2,u_2 = 101001 is $(a_1)(\bar{e}_1)(u_1)(\bar{a}_2)(\bar{e}_2)(u_2)$
- Prevent SAT solver to discover the same pair ⇒ add blocking clause
  Blocking clause := become 0 if match
  Eg. a_1,e_1,u_1,a_2,e_2,u_2 = 101001 is $(\bar{a}_1+e_1+\bar{u}_1+a_2+e_2+\bar{u}_2)$

Finding SPFD at an internal node

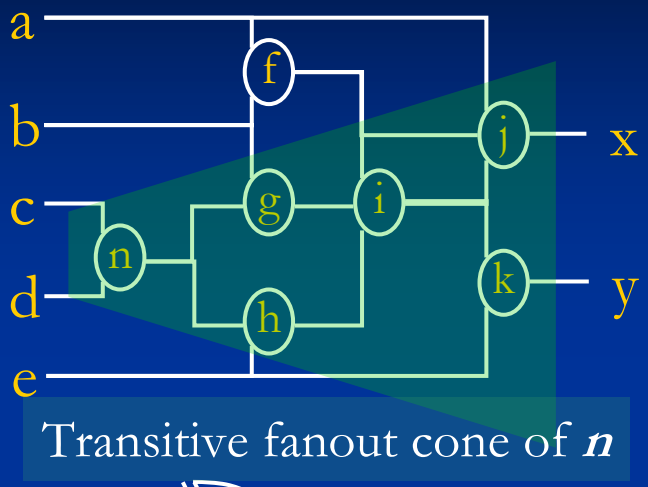
- A pair ∈ SPFD at PO ***x,*** if ***x*** distinguishes the pair.
- (101,001) distinguished by ***p*** and a PO ***y***.
  But, (101,001) ∉ SPFD(***p***)
  because NO distinguishable **path** from ***p*** to a PO.

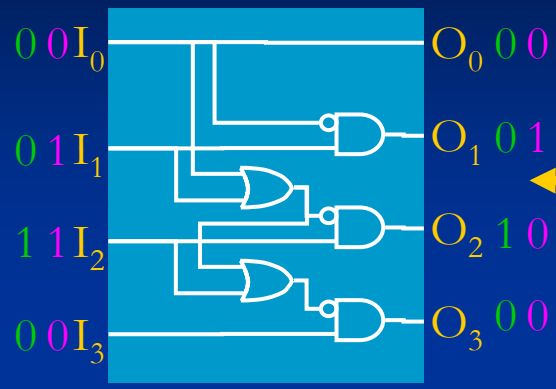


- To find SPFD at an internal node

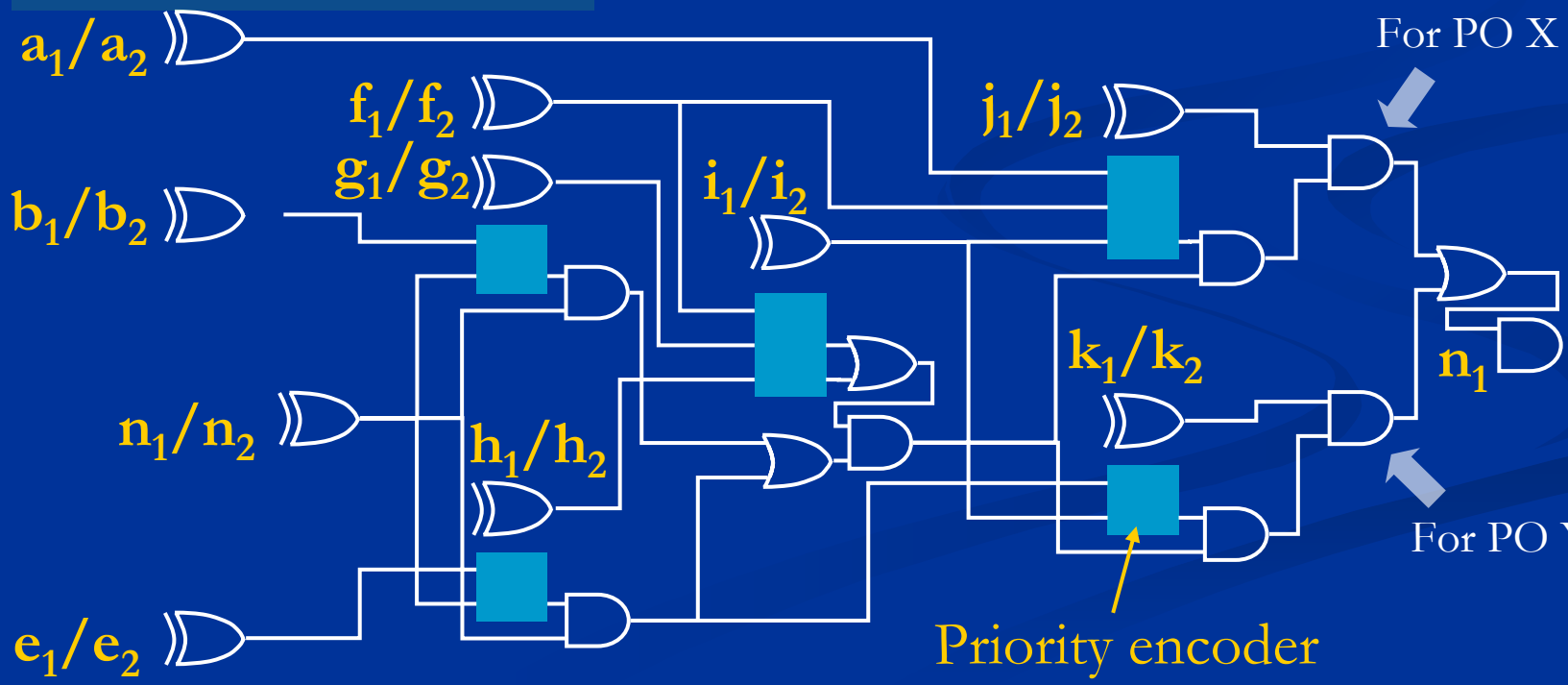
  ⇒ need to add some circuits to trace a distinguishable path

Miter to Find SPFD(n)

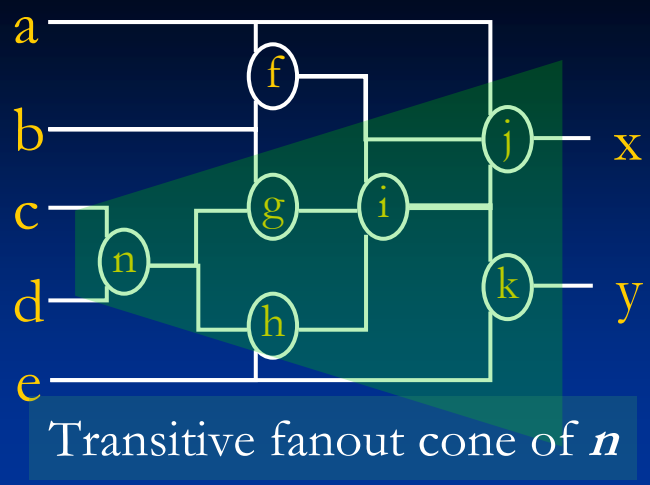




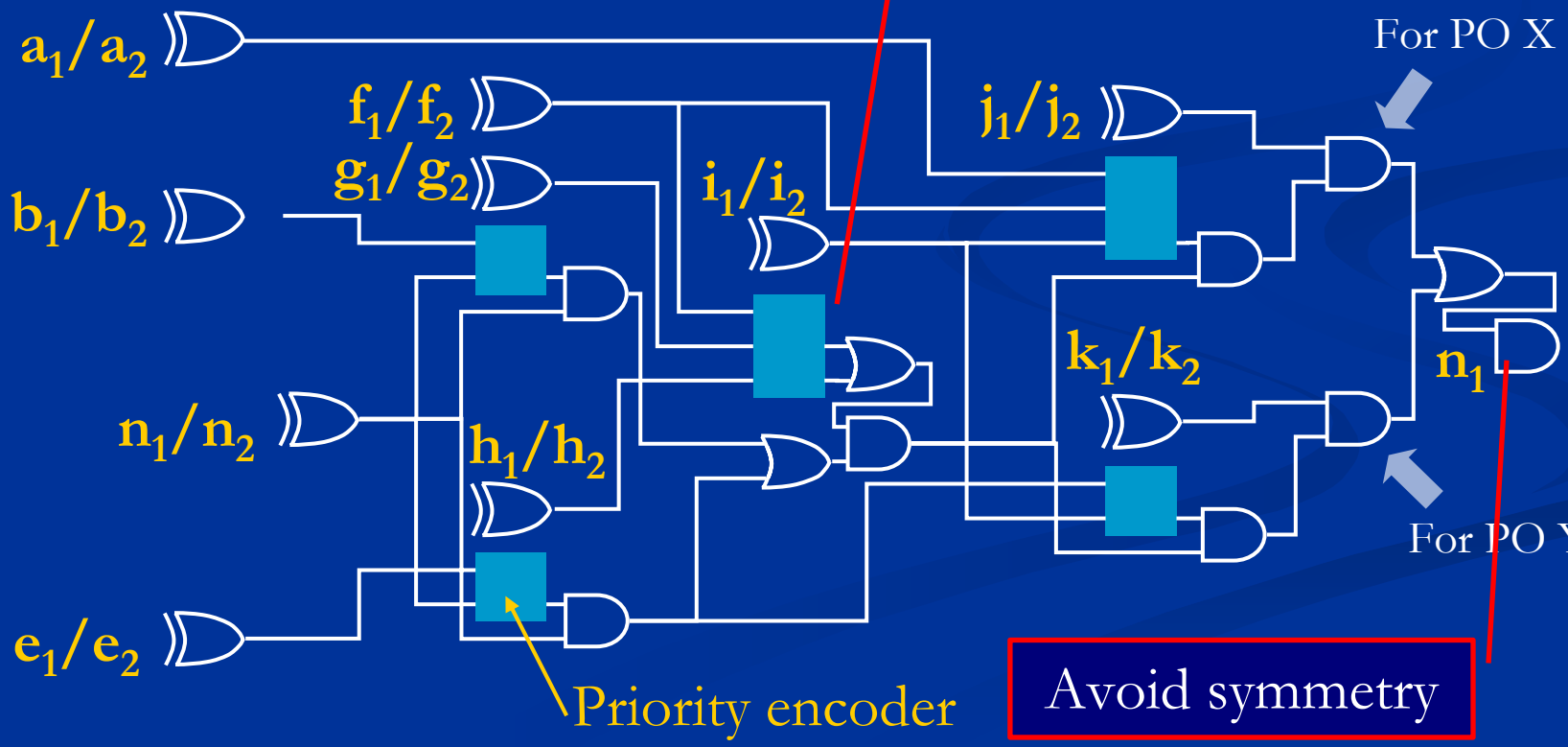
Priority encoder maintains Distributing order



How It Works



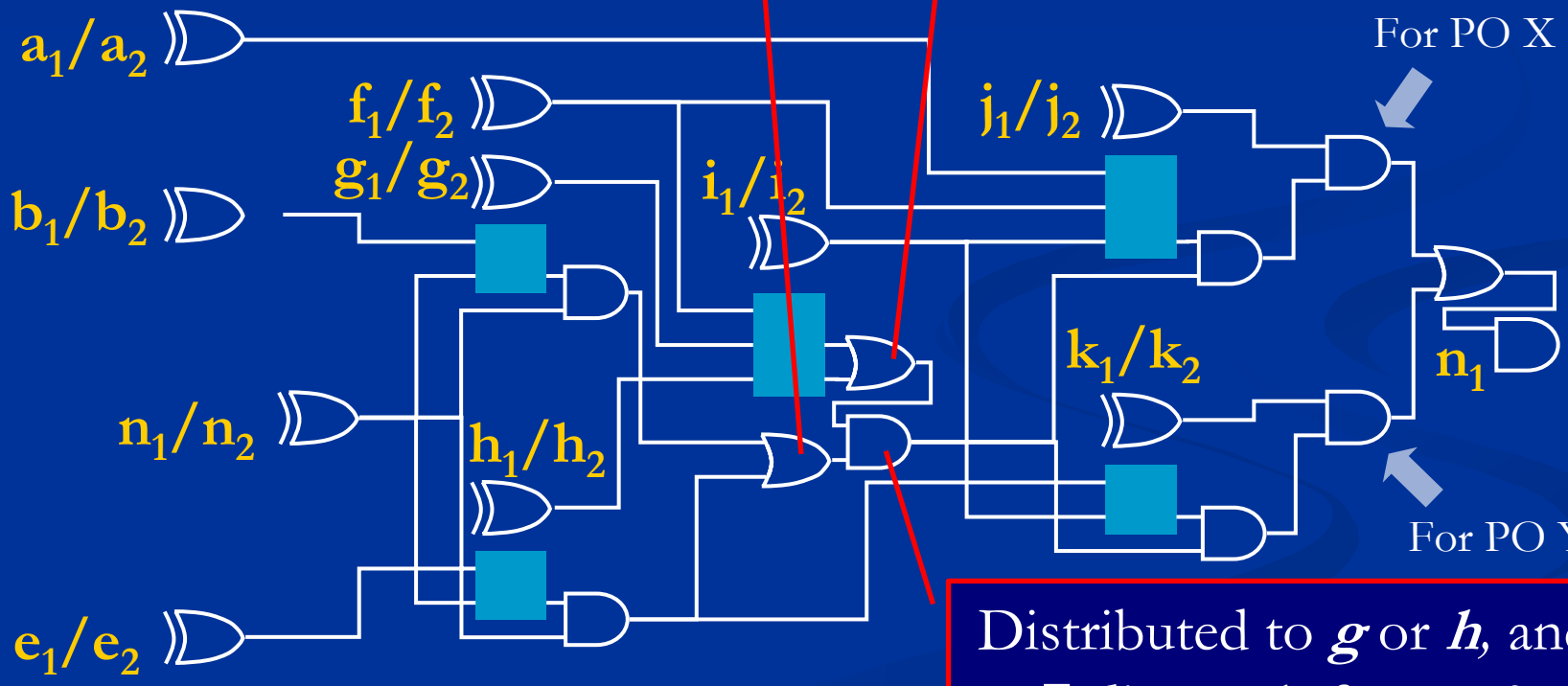
If node ***f*** can distinguish, we are not interested because it won't go to ***n*** (priority encoder makes other outputs 0)



How It Works

Distinguishable path from ***n***?
(output of OR = 1?)

The pair is distributed to either ***g*** or ***h***?
(output of OR = 1?)



Distributed to ***g*** or ***h***, and
∃ dist. path from ***n***?
(output of AND = 1?)

Computing SPFD ?

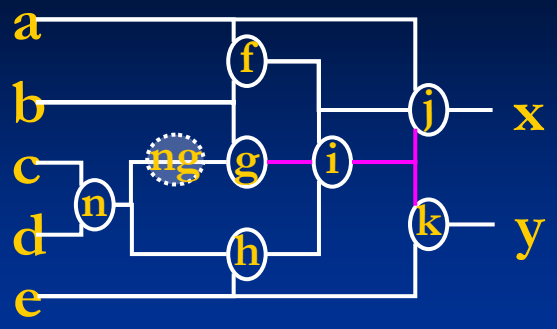
- Conventional rewiring : represent all SPFDs by BDD
- Miter can enumerate all pairs of SPFD(n)
- #pairs of SPFD is very large

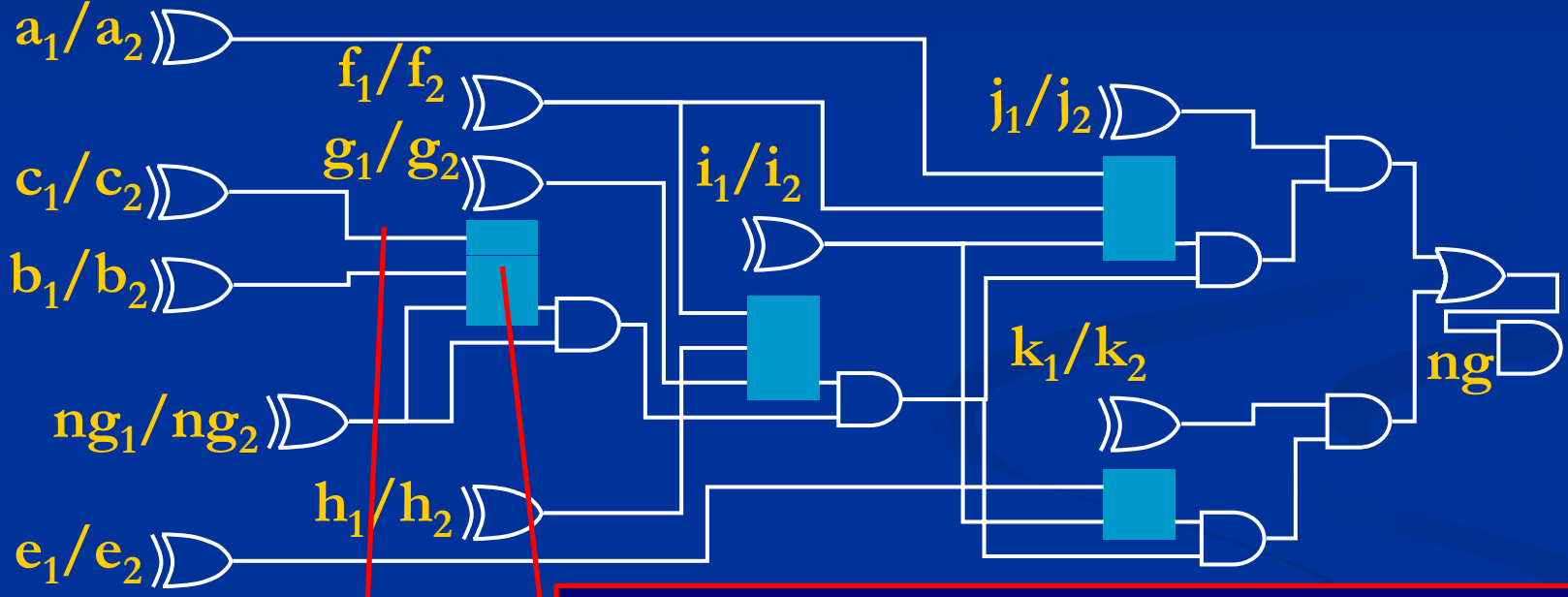
  $\Rightarrow$ takes long time to enumerate
- Can we do rewiring without explicitly enumerating SPFD?

  Yes, we can.

Checking Miter

Check necessary condition for replacing (n,g) with (c,g)



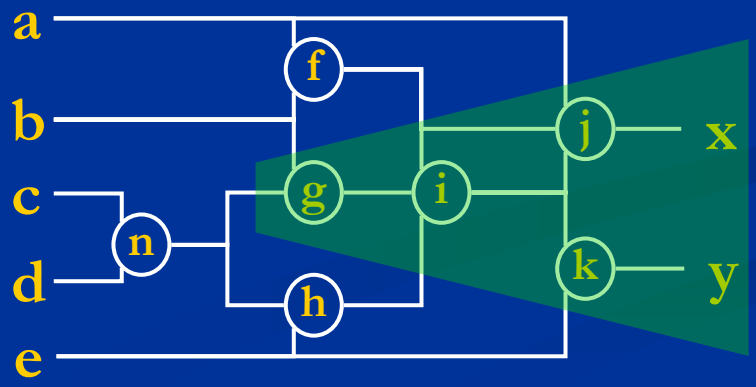


For new wire, as the first input

If both (n,g) and (c,g) distinguish the pair, the pair is distributed to (c,g) ⇒ output 0
⇒ If (c,g) distinguishes all pairs of (n,g), this SAT instance is unsatisfiable.

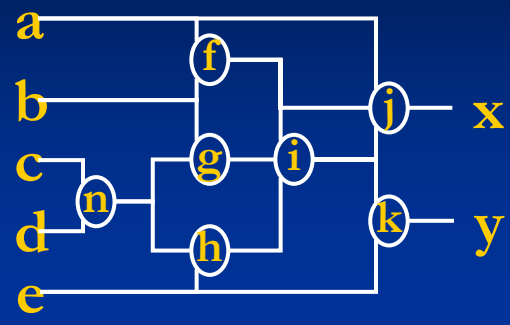
Find New Local Functions

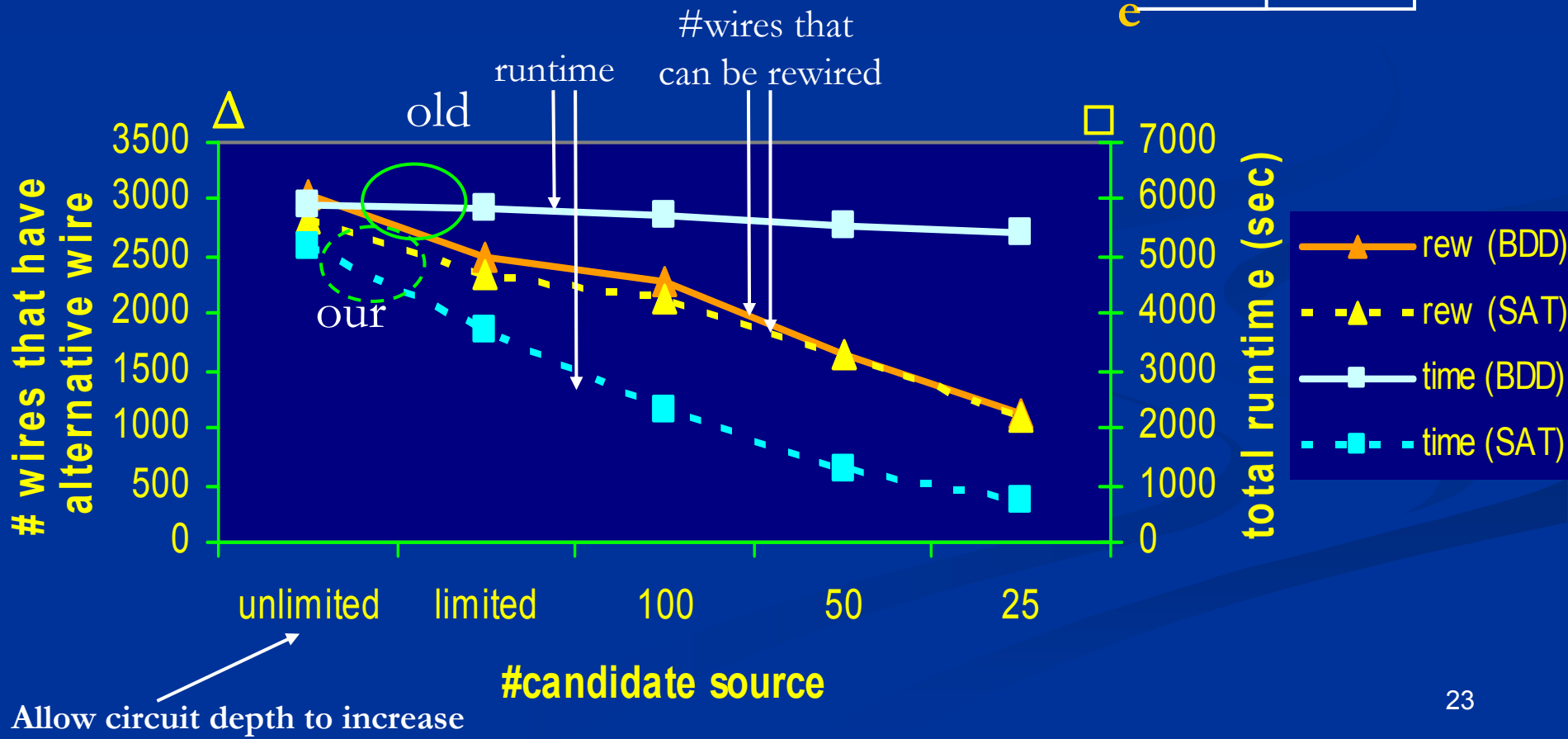
- After rewiring, the flow of SPFD may be different.
  ⇒ Need new functions for nodes ∈ TFO(***g***)
- Care minterms of local function at ***g***
  := Minterms need to propagate SPFD through ***g***.
  ⇒ Build miter to enumerate SPFD(***g***) and record needed minterms.
  SLOW
- More efficient way
  - Key points :
    - Most care minterms at ***j*** were discovered while enumerating SPFD(***g***).
    - Many input vectors are mapped to the same local minterm at ***g*** : no enumeration.
  - Using a miter, at a node, forcing a minterm on one copy and discover another minterm on the other copy.



Experimental Results

- Pick one wire to be removed in turn.
- BDD implementation doesn't finish 3 largest MCNC benchmarks, SAT finishes all.



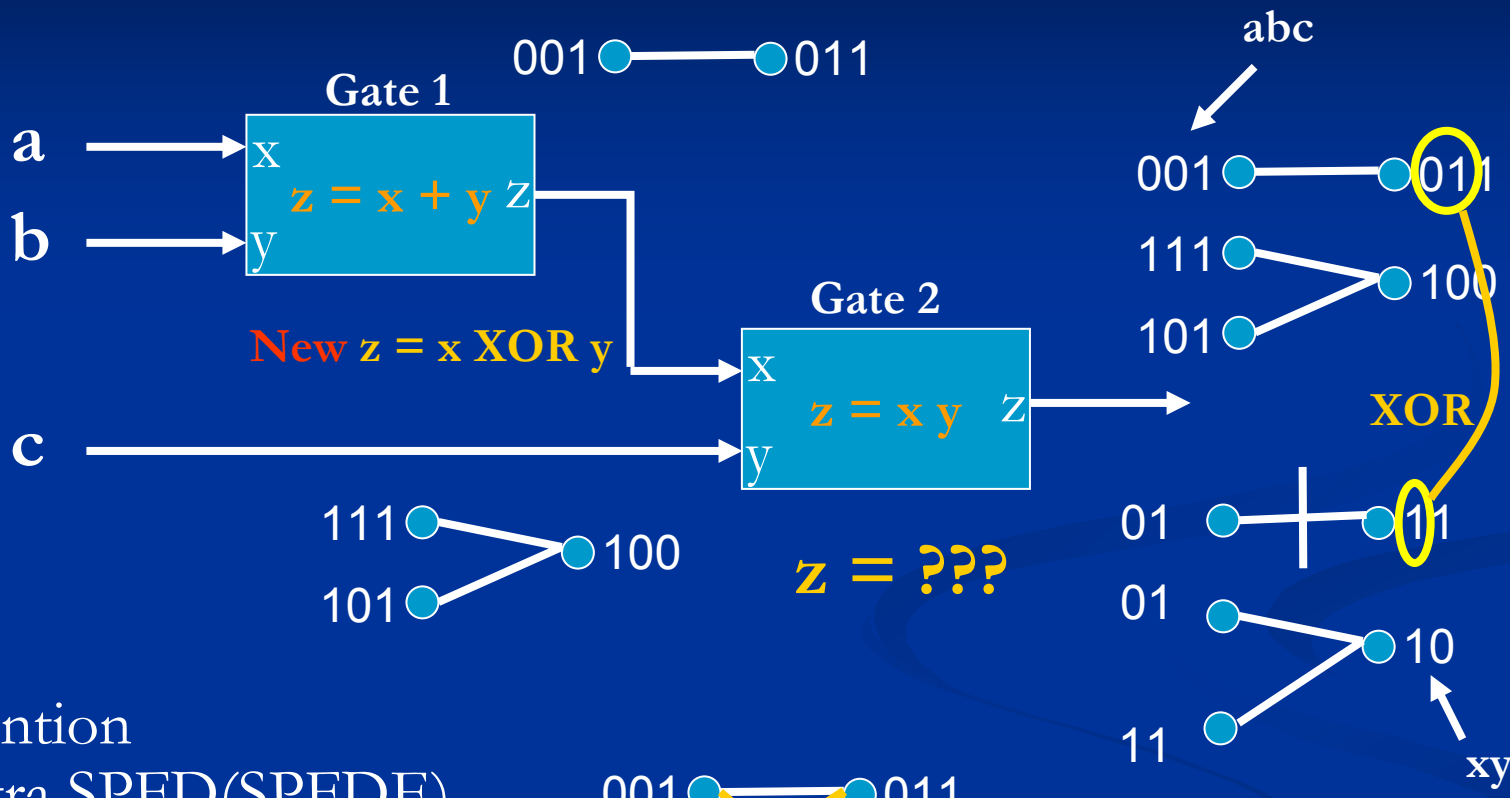


Conclusions

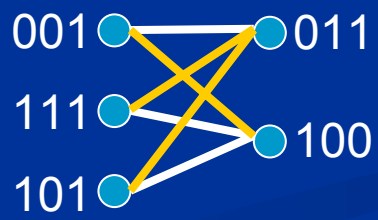
- Used SAT instead of BDD ➔ fast
- Ensure SPFD flow using additional gates and priority encoders.
  - Distinguishable path going from PO to a node
- Devise a SAT instance to determine if the rewiring is invalid in one SAT run (quick screening):
  - Priority encoder ensures distinguishable path goes only through the new wire
- Very fast, comparable quality to BDD

Thank you

Limitation of the Miter



- Prevention
  - Extra SPFD(SPFDE)
  - Required SPFD(SPFDR)



Possible because all edges in SPFD are known

- **A miter cannot capture SPFDE.**