# SAT-based Synthesis of Clock Gating Functions Using 3-Valued Abstraction

**Oleg Rokhlenko**

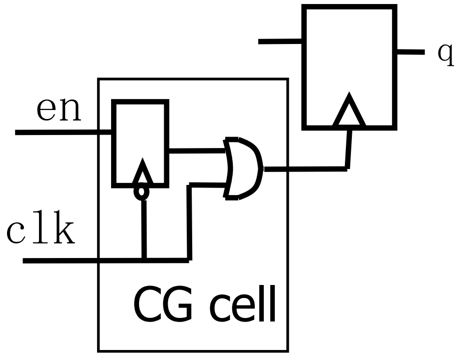
**Joint work with Eli Arbel and Karen Yorav**

IBM Haifa Research Labs

## Clock-Gating

- Saving power by stopping the clock
- Clocks consume up to 50% of dynamic power
- Clock gating
  - Reduces dynamic power consumption
  - Prevents unnecessary switching of parts of the clock network

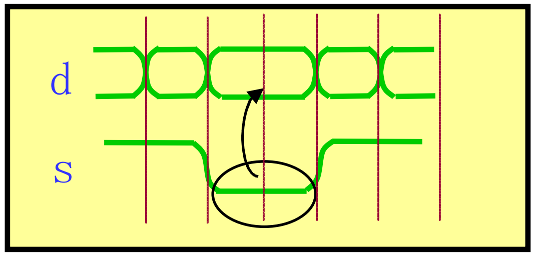
- Fine-grained clock gating
  - Analyze each latch/FF separately
    - Gate-level analysis



# Clock gating analysis approaches

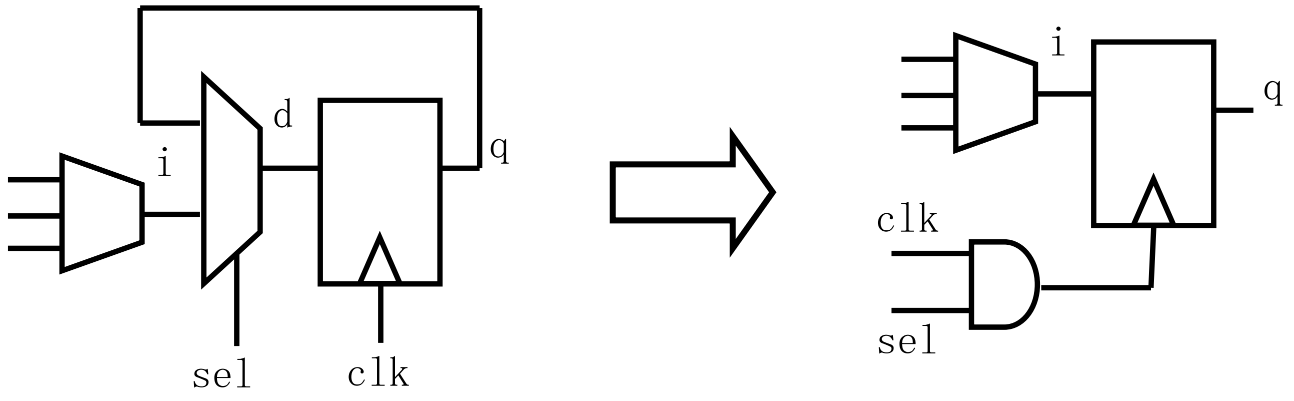
- Structural analysis
  - RTL-coding/gate-level structures
  - Scalable but limited in strength
- Functional analysis
  - Simulation based
    - Partial coverage of design behavior
  - Formal
    - Finds all opportunities but capacity is an issue

```
always @(posedge clk)
      if ( sel )
         d = i
```



## Typical (functional) clock-gating algorithm

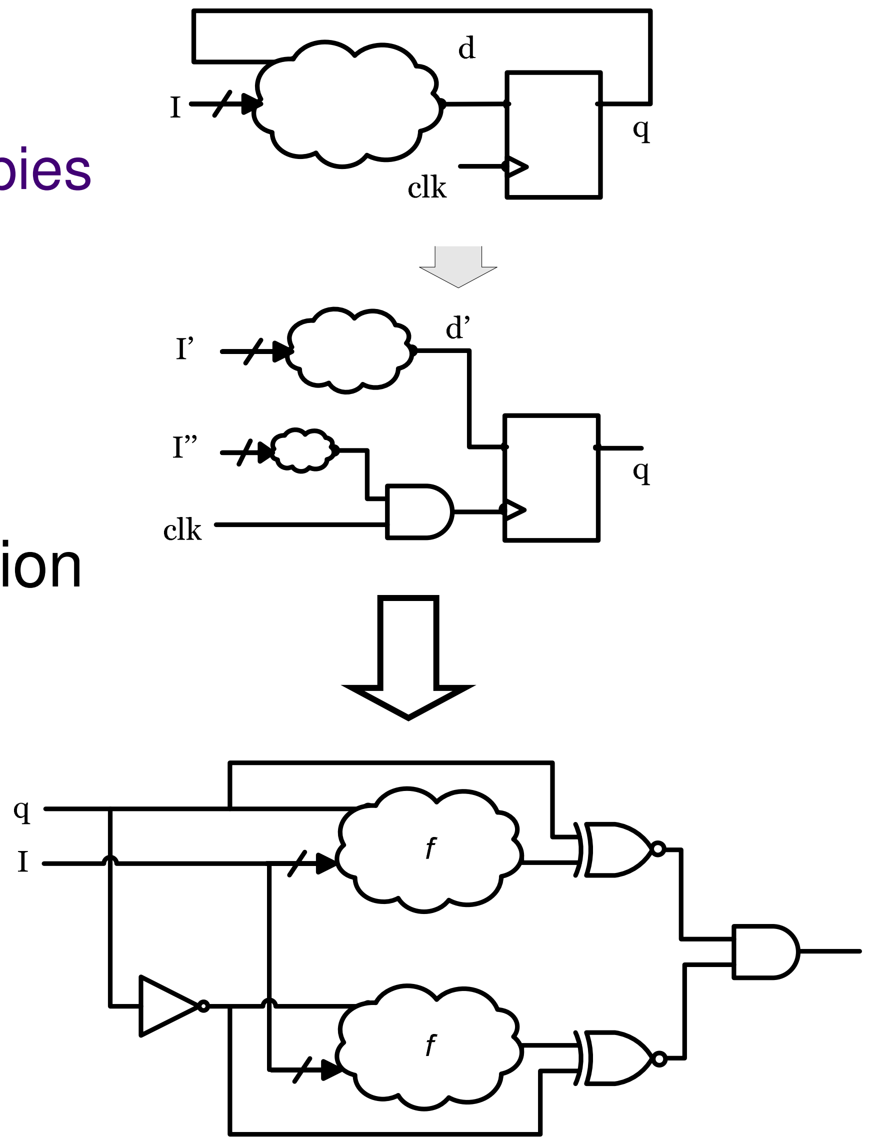
- Feedback Loop Elimination (combinatorial clock gating):
  - Based on hold conditions



- A valid gating function – but may be infeasible
  - E.g. timing/area constraints
- There are other types of clock gating
  - Sequential clock gating, e.g. based on unobservability conditions

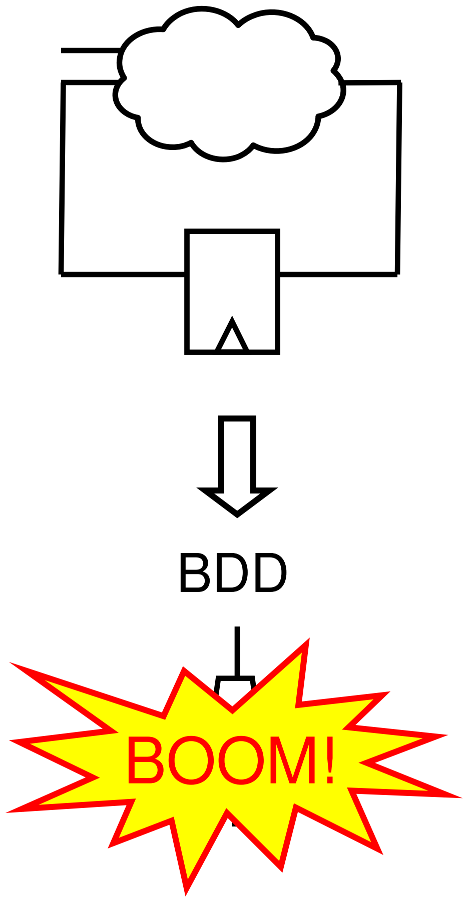
## BDD-based clock-gating

- Build the strongest function
  - typically based on one or more copies of the next-state function
- Minimize the function
  - by building its BDD
- Synthesize a net-list implementation
  - translation from BDD to net-list
- Timing constraint:
  - a CG signal may arrive too late, skewing the clock signal



## BDD-based clock-gating

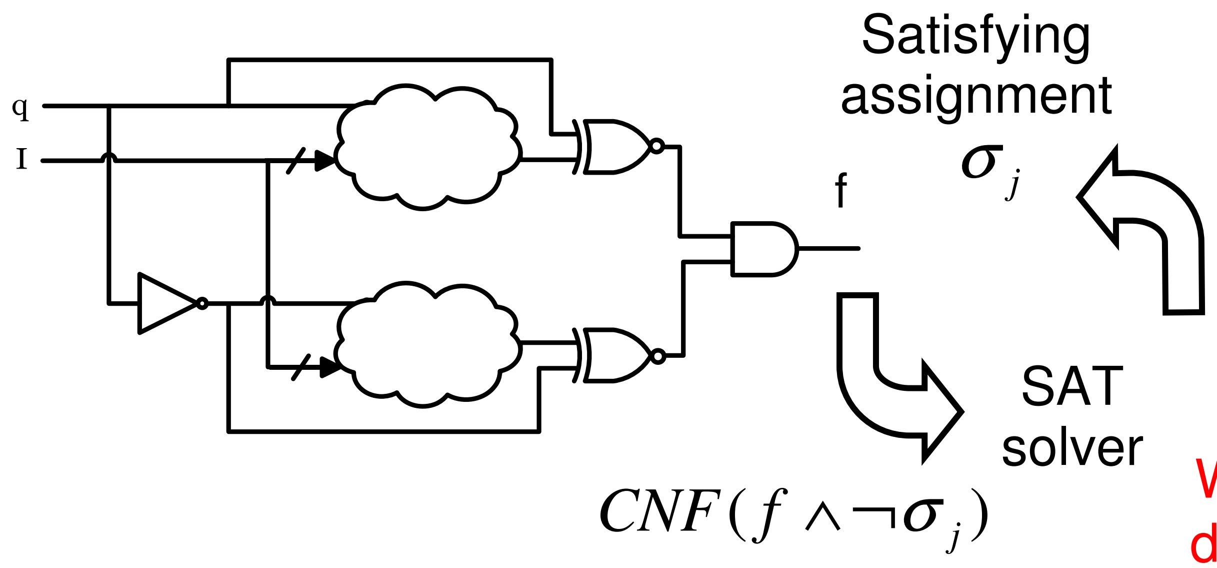
- Build the strongest function
  - typically based on one or more copies of the next-state function
- Minimize the function
  - by building its BDD
- Approximate the function
  - to allow for timing/area constraints
  - e.g. by “trimming” the BDD
- Synthesize a net-list implementation
  - translation from BDD to net-list



# Our contribution: SAT-based clock-gating

**Overview**:

- Send the algorithmic gating function to a SAT-solver
  - each satisfying assignment is a gating opportunity
- Use an “all-SAT-like” algorithm to produce assignments
  - the result is the disjunction of all assignments



?

What about the depth problem?

## Our contribution: SAT-based clock-gating

**Overview**:

- Send the algorithmic gating function to a SAT-solver
  - each satisfying assignment is a gating opportunity
- Use an “all-SAT-like” algorithm to produce assignments
  - the result is the disjunction of all solutions
- Make the solver produce bounded-size clauses
  - directly generating the approximated solution
- Further optimize if needed
  - possibly using BDD-based solutions

# 3-valued logic

- In 3-valued logic the value X stands for *unknown*

| $\wedge$ | 0 | 1 | $X$ |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | $X$ |
| $X$ | 0 | $X$ | $X$ |

| $\vee$ | 0 | 1 | $X$ |
|---|---|---|---|
| 0 | 0 | 1 | $X$ |
| 1 | 1 | 1 | 1 |
| $X$ | $X$ | 1 | $X$ |

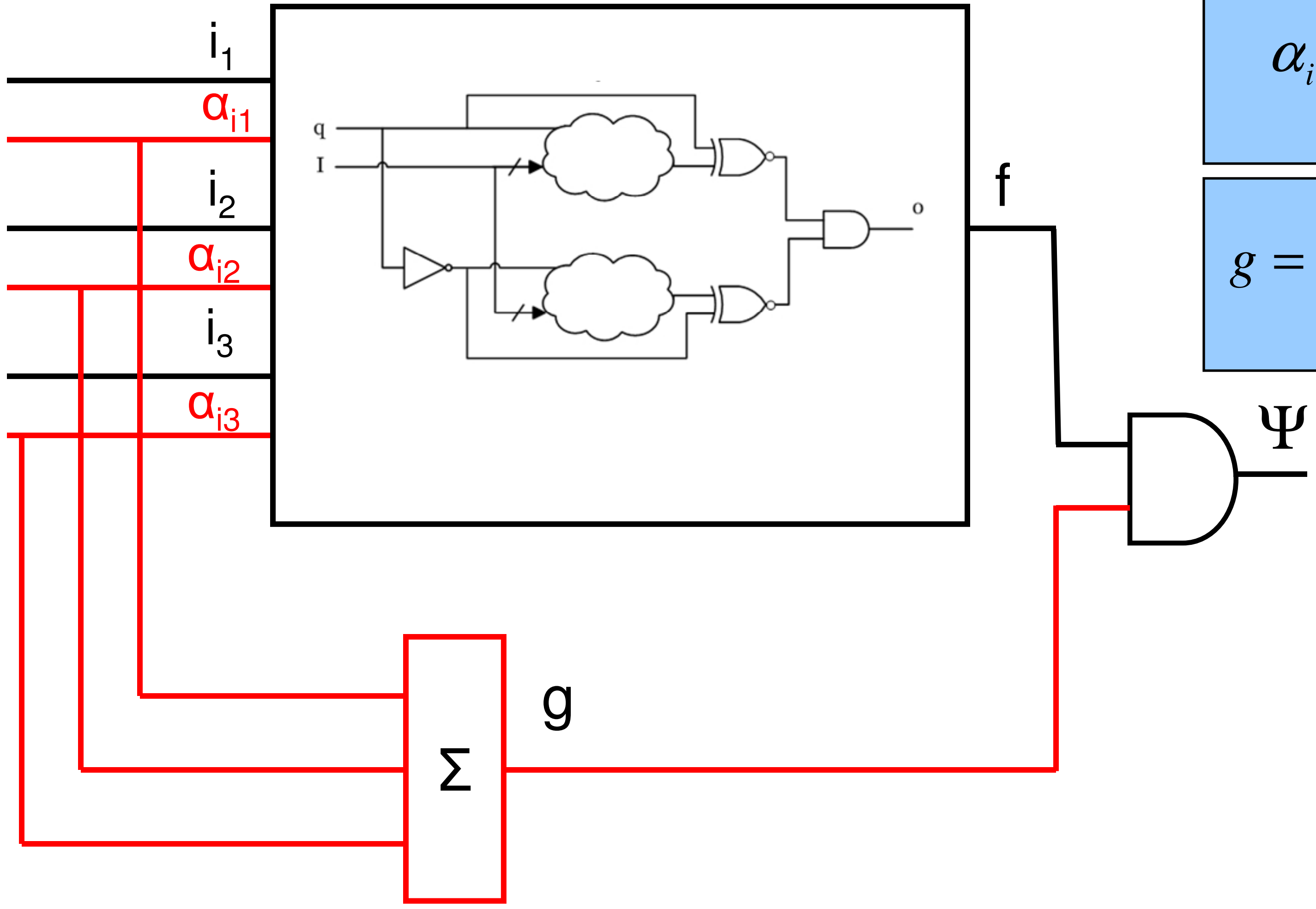
| $\neg$ | |
|---|---|
| 0 | 1 |
| 1 | 0 |
| $X$ | $X$ |

- If for some assignment $\overline{\sigma}$ , $\overline{\sigma}(i) = X$ , $i \in inputs(f)$ then

$$f(\overline{\sigma}) = b \rightarrow \forall i : f(\overline{\sigma}) = b \qquad b \in \{0,1\}$$

- Since it's a one way implication, it's an *approximation*.
  - Xs values imply universal quantification but not every quantification can be done with Xs

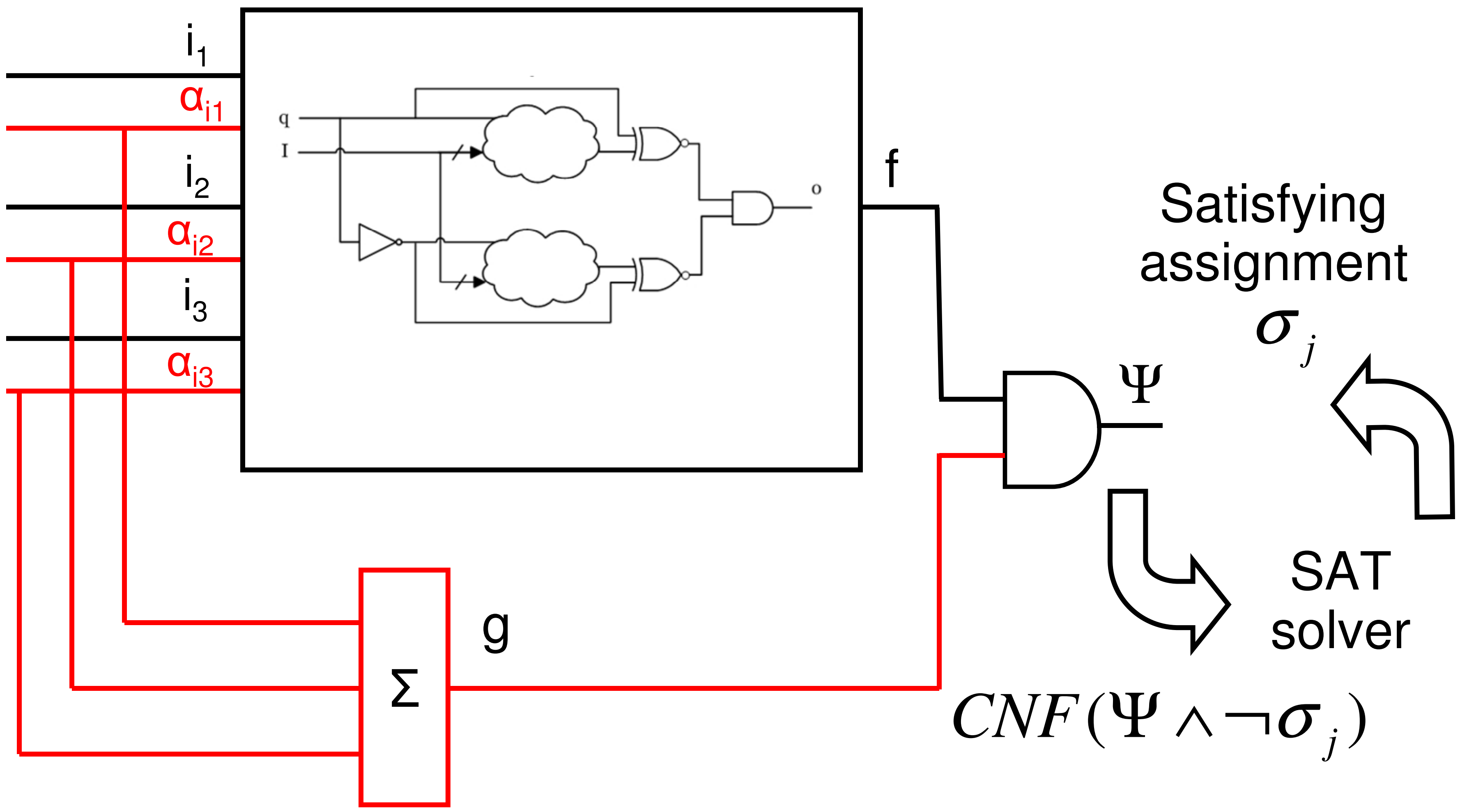
## SAT-based Synthesis of Clock Gating Functions



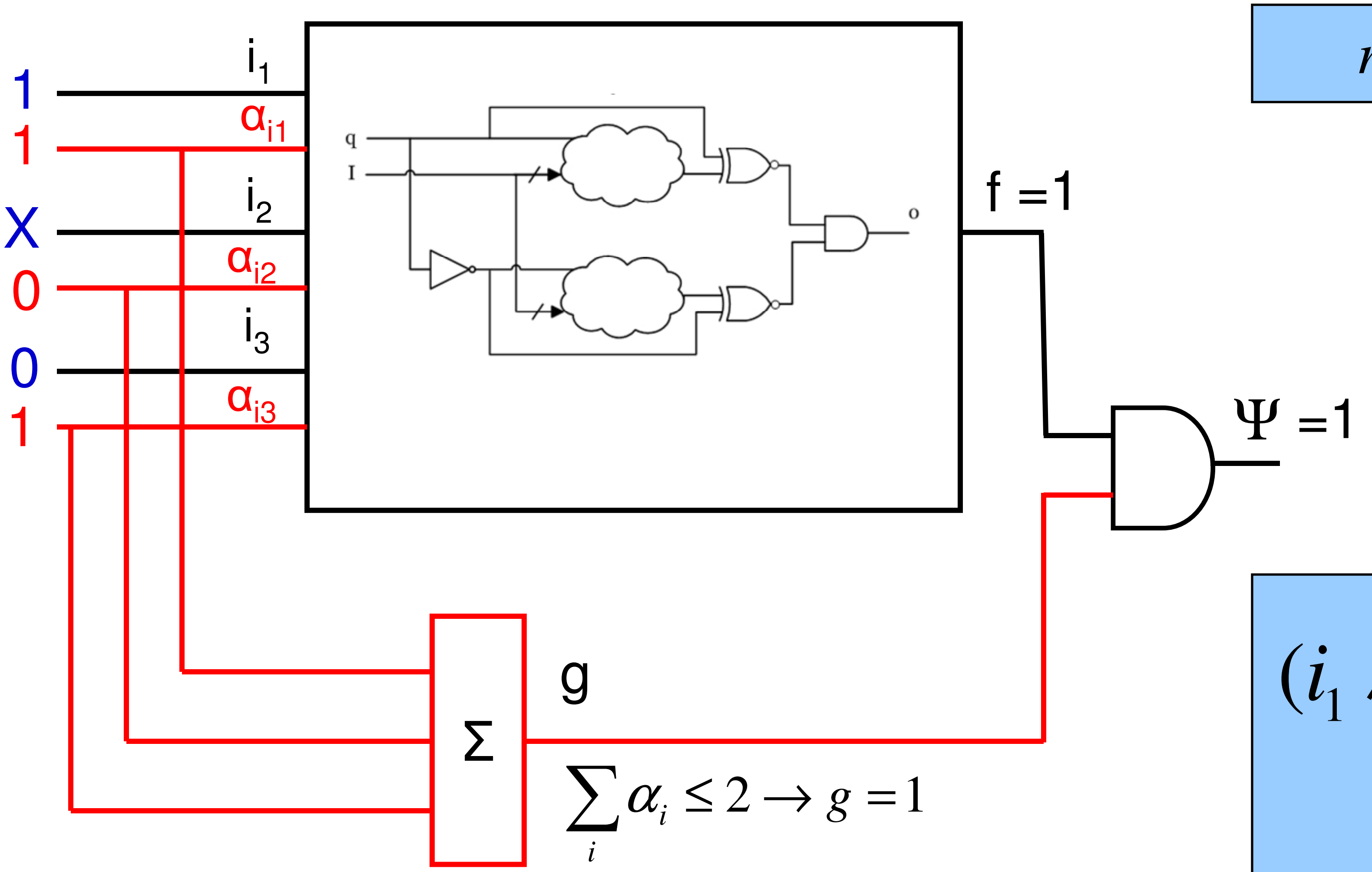
$$\alpha_i = 1 \leftrightarrow i \neq X$$

$$g = 1 \leftrightarrow \sum_i \alpha_i \leq n$$

## SAT-based Synthesis of Clock Gating Functions



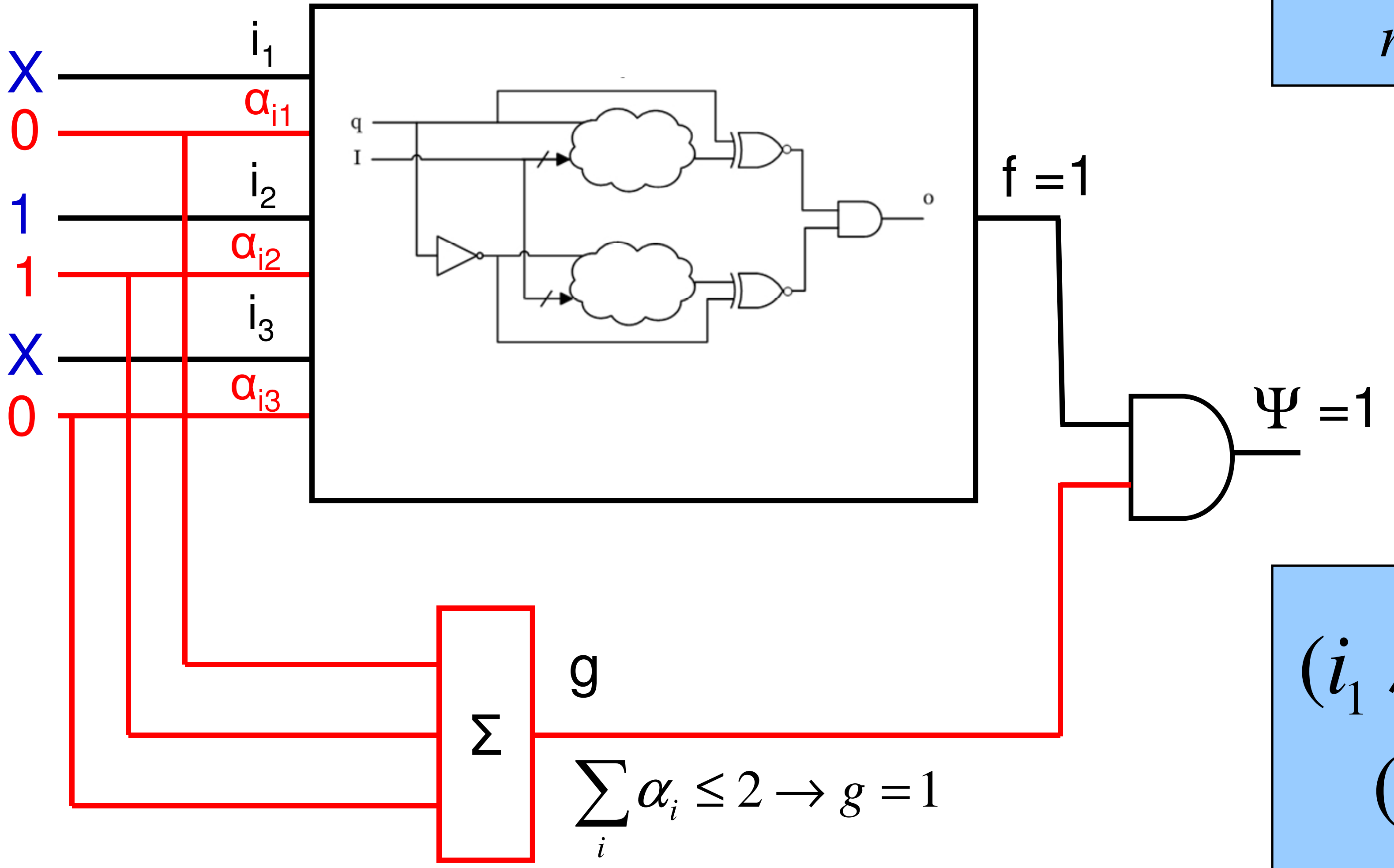
## Example



$n = 2$

$(i_1 \wedge \bar{i}_3)$

## Example



$$n = 2$$

$$(i_1 \wedge \bar{i}_3) \vee (i_2)$$

## Experimentation

Latches on which BDD timed-out

We show only designs with > 10% of hard latches

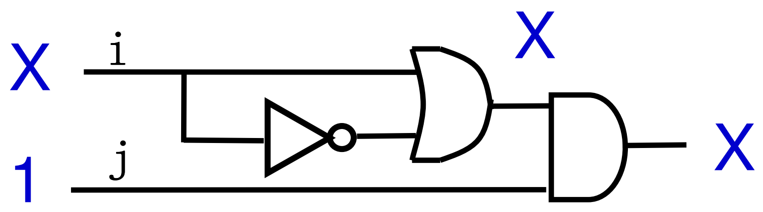
| Design | Candidates | Hard | % Hard | SAT solved | % Solved |
|---|---|---|---|---|---|
| D1 | 585 | 418 | 71.45% | 418 | 100.00% |
| D2 | 576 | 273 | 47.40% | 273 | 100.00% |
| D3 | 397 | 243 | 61.21% | 243 | 100.00% |
| D4 | 1096 | 126 | 11.50% | 126 | 100.00% |
| D5 | 234 | 72 | 30.77% | 72 | 100.00% |
| D6 | 409 | 62 | 15.16% | 62 | 100.00% |
| D7 | 328 | 60 | 18.29% | 60 | 100.00% |
| D8 | 219 | 48 | 21.92% | 48 | 100.00% |
| D9 | 1735 | 251 | 14.47% | 250 | 99.60% |
| D10 | 626 | 134 | 21.41% | 132 | 98.51% |
| D11 | 390 | 54 | 13.85% | 53 | 98.15% |
| D12 | 212 | 99 | 46.70% | 97 | 97.98% |
| D13 | 1580 | 202 | 12.78% | 194 | 96.04% |
| D14 | 2507 | 270 | 10.77% | 259 | 95.93% |
| D15 | 107 | 13 | 12.15% | 10 | 76.92% |
| D16 | 247 | 54 | 21.86% | 38 | 70.37% |
| D17 | 195 | 30 | 15.38% | 5 | 16.67% |

The SAT-based approach succeeded in finding a clock gating condition for more than 73% of all the hard latches.

## Over abstraction

- 3-valued abstraction is an *approximation* of universal quantification, but is not exact.
  - It is possible for there to be a term of size *n* that implies *f* while there exists no satisfying assignment to $\Psi$
- Example: $f = (i \vee \neg i) \wedge j$ $\Rightarrow$ *j* is a legal clock-gating function

  $\sigma = (X,1)$



$\Rightarrow$ For n=1 our approach won't find it

- Solution:
  - use higher values of n than our actual depth limit and then use BDD to optimize further

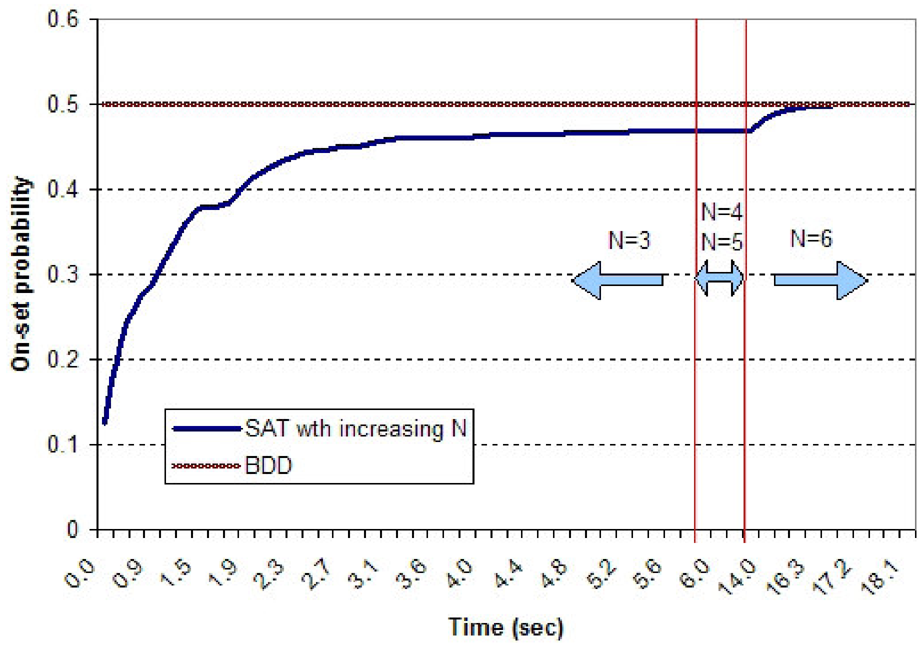
## Experimentation (over abstraction)

- For 27% of hard latches, the SAT solver reported unsatisfiability.
- Using stronger machine with much more time and memory, the BDD-based approach reported the following:
  - 2% - no gating possible
  - 78% - BDD exploded
  - 20% - BDD solved, having an average depth 50.6. After approximation (up to depth=6), only 2% remain having on-set probability > 0.1

The probability of the clock gating function evaluating to ’1’

- All-in-all only 0.5% of hard latches were missed !

# Experimentation (iterative approach)



- Iterative approach for a specific latch, when n = 1 to 6
  - Start with n=1 and increment by one each time we get UNSAT
- After 6 seconds, on-set probability is 93% of the optimum.
- The iterative approach allows us getting the strongest result.

## Conclusion

- Using SAT when BDD fails allows handling much larger designs than before.
- Using 3-valued abstraction, we are able to directly generate the (strongest) approximation.
- Our approach produces partial results even if the computation is not completed within a set time limit.
- Over approximation is present, but we are fine with it. Moreover, extending *n* beyond our target depth overcomes the over-approximation.
- In general, our approach allows universal quantification using SAT.

## Thank you !

**Questions?**