VETENSKAP OCH KONST

KUNGL TEKNISKA HÖGSKOLAN

Inte

Master Pr

in System-on-Chip Design

L7b: Rule-based optimization

Reading material

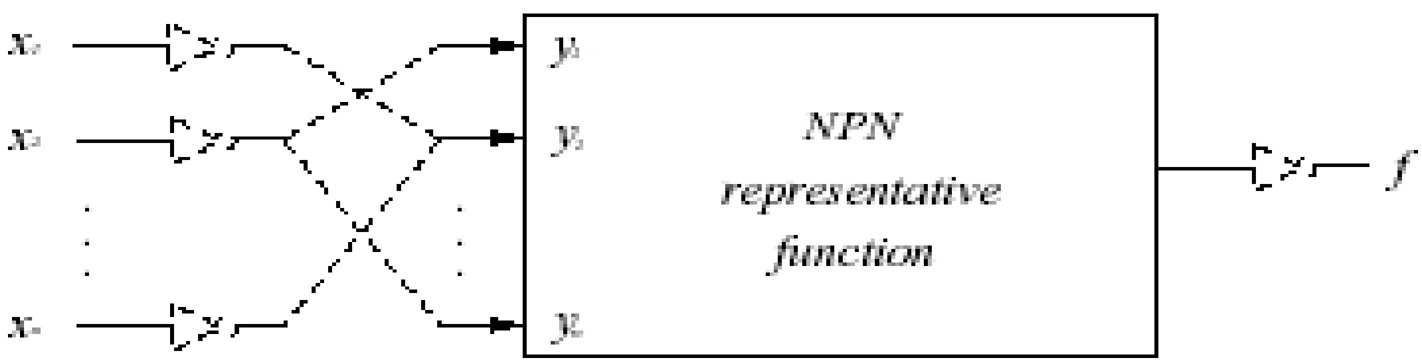
- **Fast multi-level logic optimization using local transformations,** P. Farm, E. Dubrova, *Notes of International Workshop on Logic Synthesis,* May 2003, pp. 120-126.
- **Integrated Logic Synthesis Using Simulated Annealing,** P. Farm, E. Dubrova, A. Kuehlmann), *Notes of International Workshop on Logic Synthesis,* June 7-9, 2006, Vail, Colorado, USA.

Main Idea

- The circuit is optimized by step-wise refinement
- Local transformations based on a set of replacement rules are applied at each step
- The replacement rules are generated with the help of NPN (**N**egation-**P**ermutation-**N**egation) classes of Boolean functions
- The objective is to minimize the number of gates in the circuit

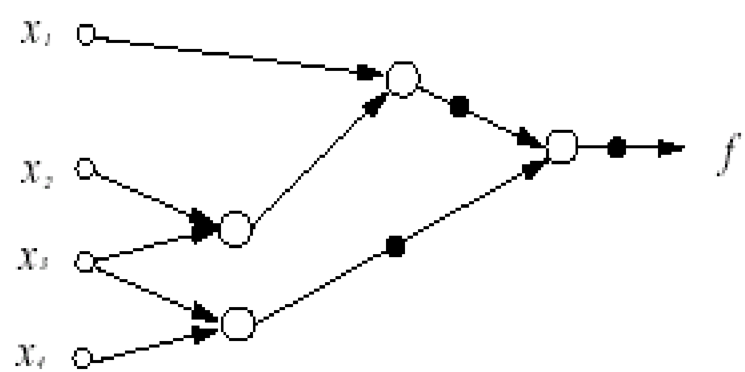
NPN classes

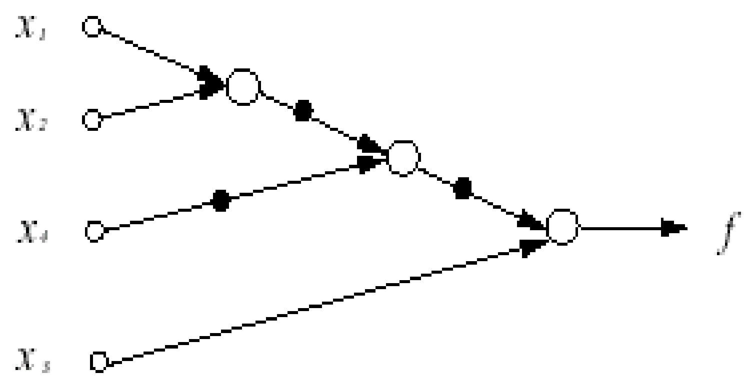
- Each NPN class consists of all functions which differ by:
  - Negation of some input variables $x_1,\ldots,x_n$ and/or,
  - Permutation of some input variables $x_1,\ldots,x_n$ and/or,
  - Negation of the function output



Replacement rules

- Functions belonging to the same NPN class have a minimum AND/INVERTER graph of the same size.
- Rules on how to transform a non-minimum graph to a minimum need to be defined
- Rules for removing redundancies also need to be defined





Rules are derived from axioms and properties of Boolean algebra

A3: $\forall a,b,c \in B$, $a \cdot (b+c) = a \cdot b + a \cdot c$, $a + b \cdot c = (a+b) \cdot (a+c)$

A4: $\forall a \in B$, $a \cdot \mathbf{1} = a$, $a + \mathbf{0} = a$

A5: $\forall a \in B$, $a \cdot a' = \mathbf{0}$, $a + a' = \mathbf{1}$

P1: $\forall a \in B$, $(a')' = a$

P2: $\forall a,b,c \in B$, $a \cdot (b \cdot c) = (a \cdot b) \cdot c$, $(a + b) + c = a + (b + c)$

P3: $\forall a \in B$, $a \cdot \mathbf{0} = \mathbf{0}$, $a + \mathbf{1} = \mathbf{1}$

P5: $\forall a,b \in B$, $a \cdot (a+b) = a$, $a + a \cdot b = a$

P6: $\forall a \in B$, $a \cdot a = a$, $a + a = a$

Completeness of rules

- By using axioms Boolean algebra, we can derive any Boolean expression representing a given Boolean function
- Therefore, by applying them as rules, we can obtain any circuit for any circuit implementing the same function
- Consequently, we can find a minimal circuit starting from any circuit if we search long enough

Simulated Annealing Algorithm

- The order of applying rules can be controlled by a simulated an annealing algorithm
- A *cost* function assigns a cost to each state s of the search space S
- The set of *neighbours* is the set of states which can be obtained from $s \in S$ by a single move
- Two functions are assigned to states:
  - Selection probability
  - Acceptance probability

Simulated Annealing, cont

- Let S be a set of all possible combinational acyclic Boolean circuits implementing a Boolean function f
- At each step, the annealing algorithm considers some neighbor s' $\in$ S of the current state s $\in$ S, and probabilistically decides between moving to state s' of staying in state s'
- The probabilities are selected so that the system ultimately tends to move to state the lower cost

Neighbours

- In our case, the neighbours of a state s are the states which can be obtained from s by applying one of the rules listed on p. 7
- There are also a rules which is related specifically to circuits:
  - Two isomorphic vertices can be merged into one
  - A vertex with a multiple fan-out can be divided into two vertices

Pseudo-code of annealing algorithm

```
algorithm ANNEALING(c1, ε);
    best_c = c1;
    best_cost = ε(c1);
    (tb, te, Δt, N_of_moves) = INITIALIZE(c1, ε);
    t = tb;
    while t > te do
        for N_of_moves do
            c2 = SELECT(c1);
            if ACCEPT(ε(c1), ε(c2), t) then
                c1 = c2;
                cost = ε(c2);
                if cost < best_cost then
                    best_c = c1;
                    best_cost = cost;
        t = t − Δt;
end
```

The annealing schedule

- As the simulation proceeds, the temperature t is gradually reduced
  - Initially, t is set to a high value
  - Then, it is decreased at each step according to some annealing schedule
  - Finally, it becomes 0
- In the way, the system is expected to initially explore a broad region of the search space containing good solutions
- Then, it narrows and at $t = 0$ moves downhill

Convergence to optimum

- If the annealing process is scheduled correctly, it converges to optimum solution asymptotically
- In general, annealing can smoothly trade-off complex, multi-dimensional objective functions
  - Area, delay, power, etc.