Technology Mapping

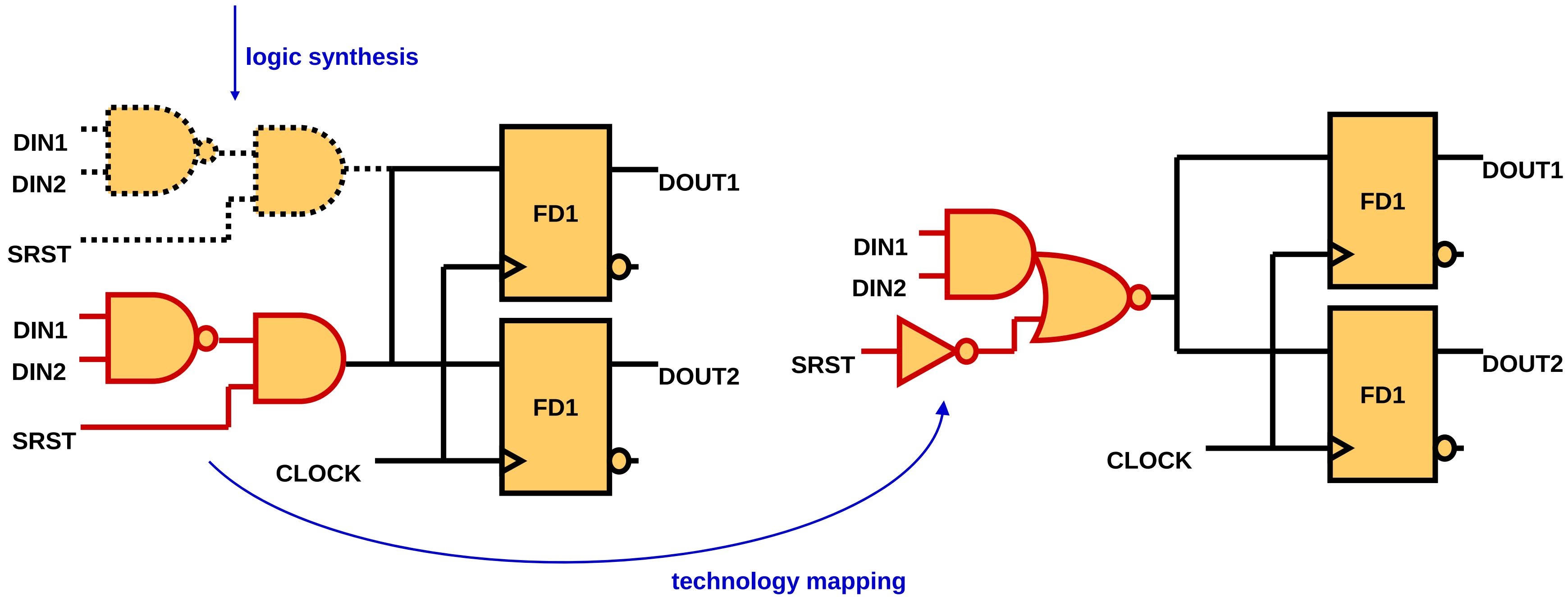
(4541.554 Introduction to Computer-Aided Design)

School of EECS

Seoul National University

Technology Mapping Problem

```
process
begin
    wait until rising_edge(CLOCK);
    if (SRST='0') then
        DOUT1 <= '0';
    else
        DOUT1 <= DIN1 nand DIN2;
    end if;
    DOUT2 <= SRST and (DIN1 nand DIN2);
end process;
```

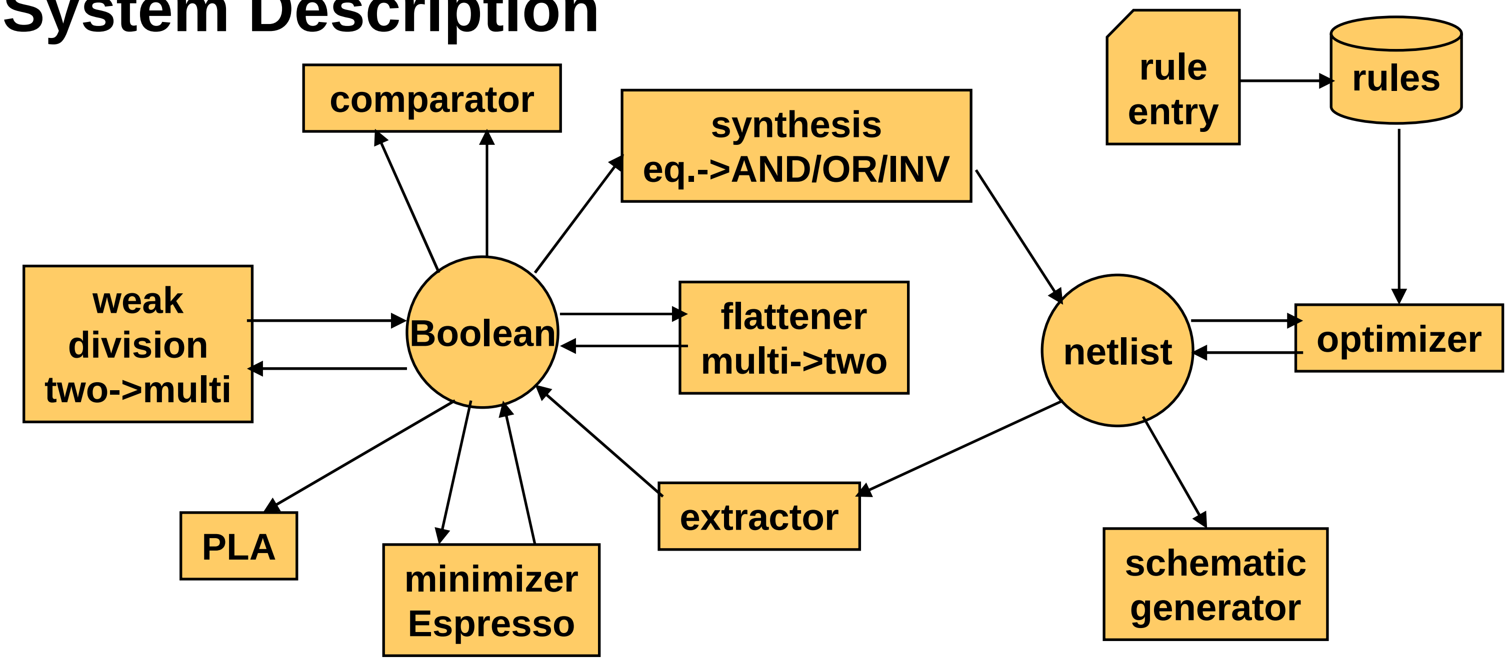


- **Problem**
  - **Given**
    - **Netlist**
    - **Library of components**
    - **Performance goals (area, speed)**
  - **Find**
    - **Best implementation w.r.t. performance goals**

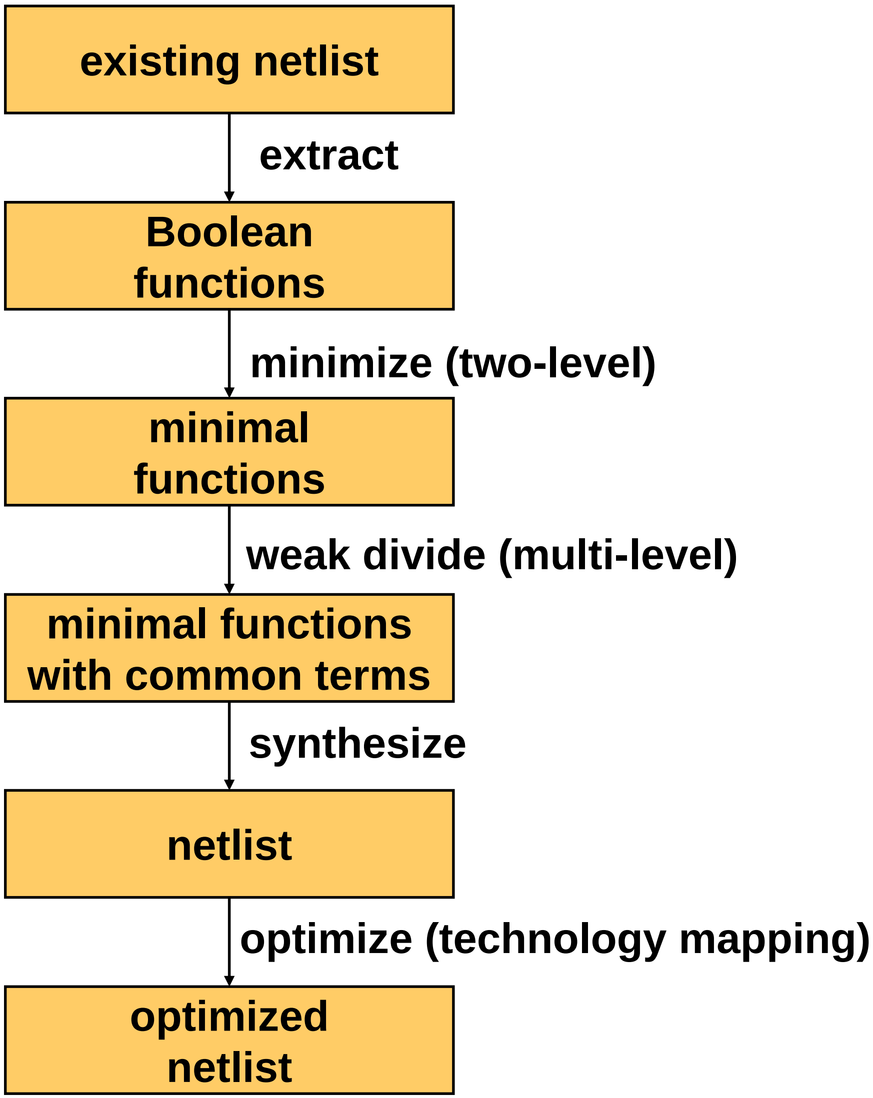
- **Methods**
  - **Rule-based: LSS, SOCRATES**
  - **Algorithmic (graph covering): DAGON, MIS**

Rule-Based Technology Mapping

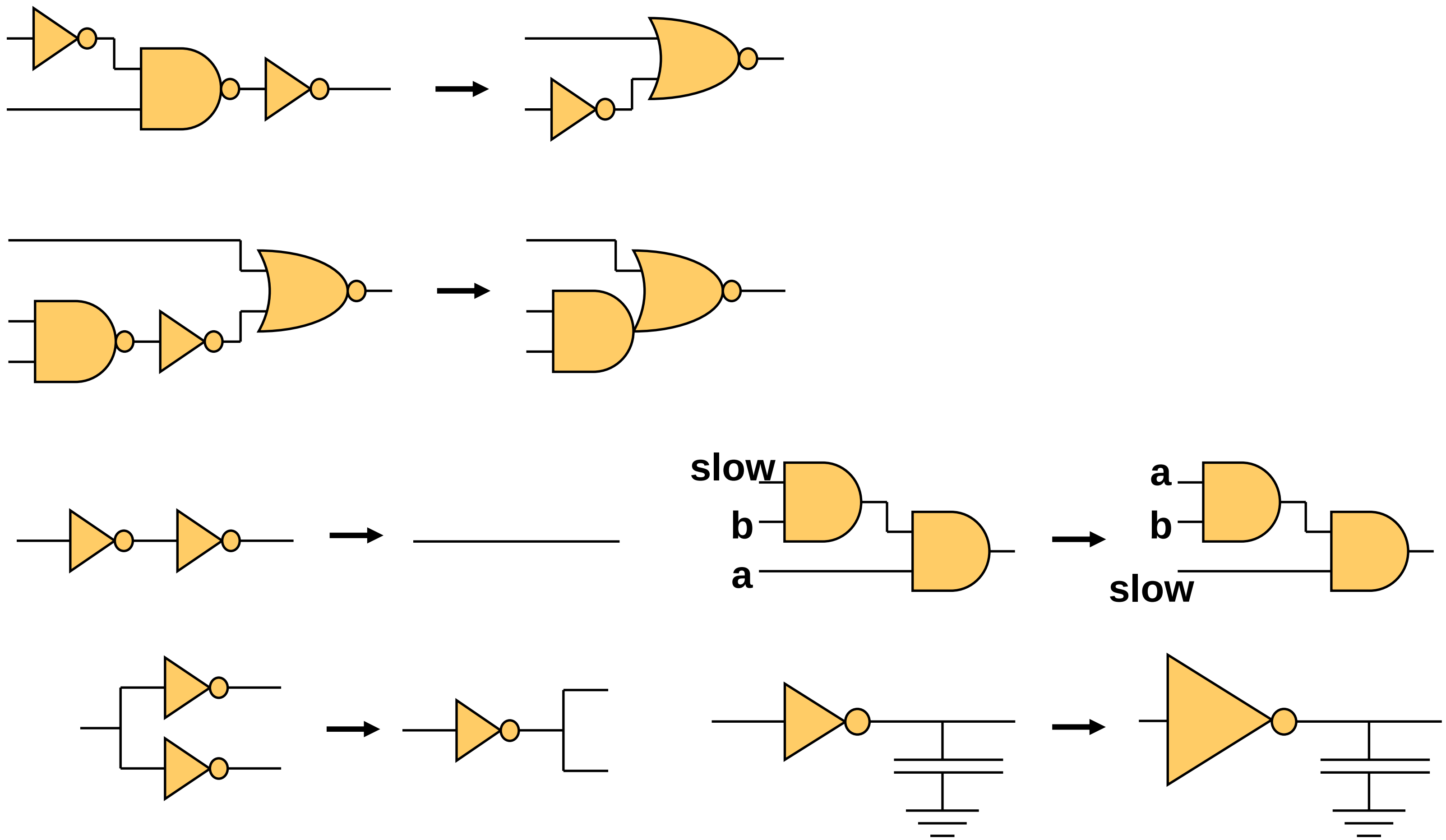
- **SOCRATES**
  - **Synthesis and Optimization of Combinatorics using a Rule-based And Technology-independent Expert System**
  - **A. J. de Geus and W. Cohen, "A Rule Based System for Optimizing Combinational Logic," *IEEE Design & Test of Computers*, Aug. 1985.**
- **System Description**



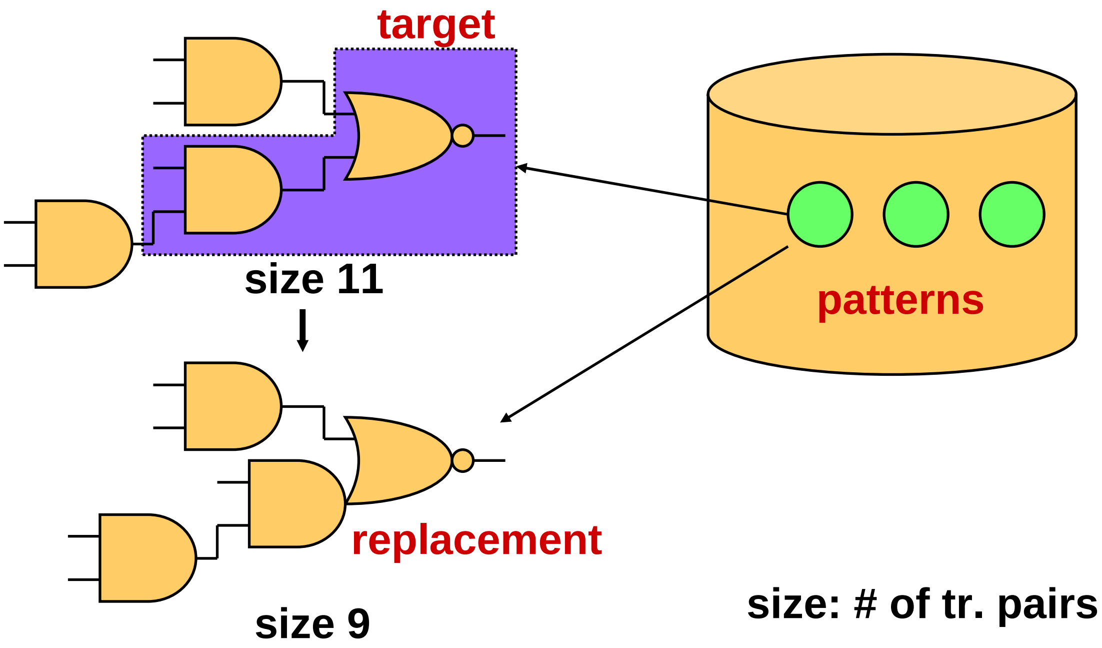
- **Design Scenario**



- **Technology Mapping**
  - **A series of local transformations based on rules**
  - **Example rules:**



- **Main Operations**
  - **Matching: Find a number of rules that apply to the present network**
  - **Cost function evaluation: For each potential rule application, a cost function is computed to determine the quality of the resulting circuit**
  - **Selection: Decide which rule should be applied**
  - **Replacement: Perform network transformation by applying the selected rule**

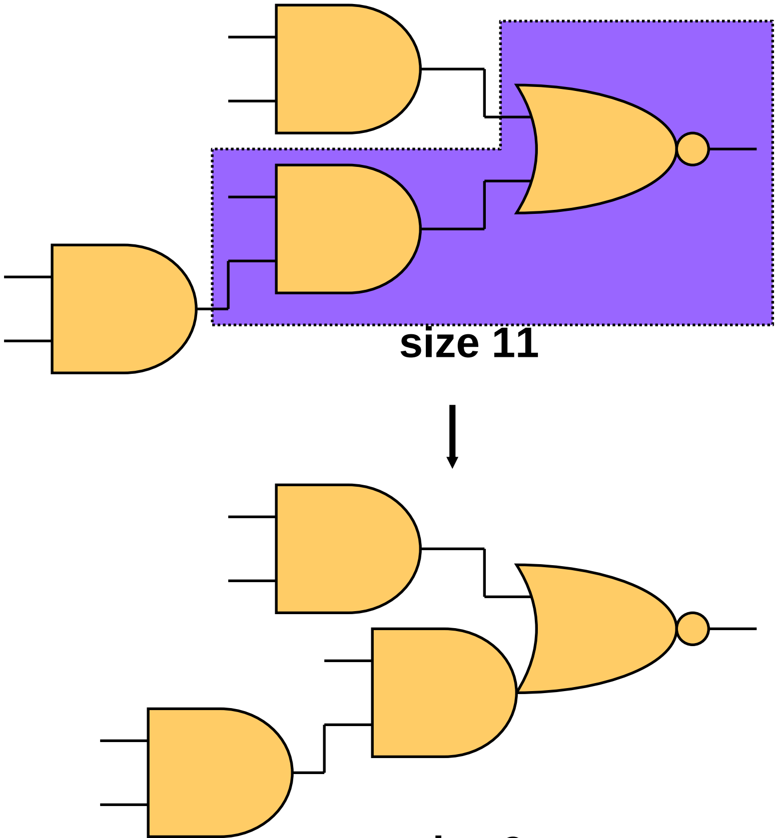


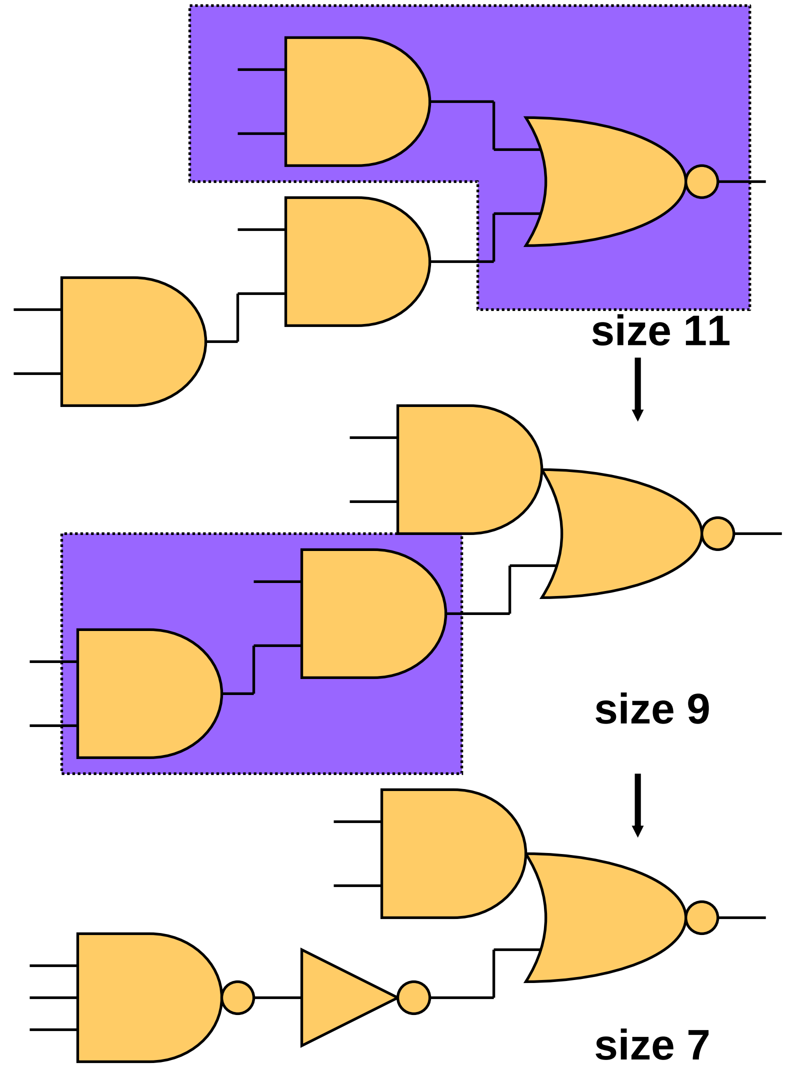
- **Search Strategies**
  - **State space search problem**
  - **Greedy algorithm**

```
Order rules in the knowledge base.
do {
   for each rule R {
      for each gate G {
         if rule R matches at gate G
            if cost improves
               apply_rule (R, G)
      }
   }
} while improving
```

--> local minimum

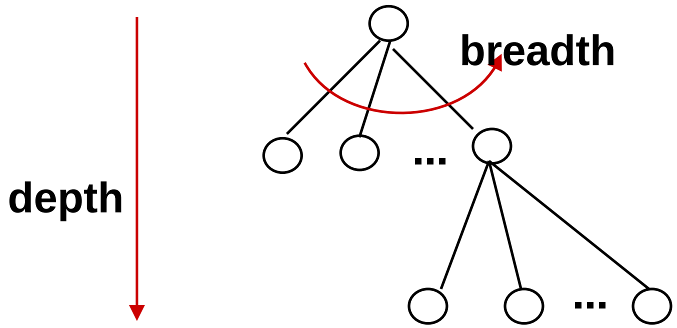
Rule-Based Technology Mapping





– **Look-ahead strategy**

- **Search tree**



- **Complexity of the search**

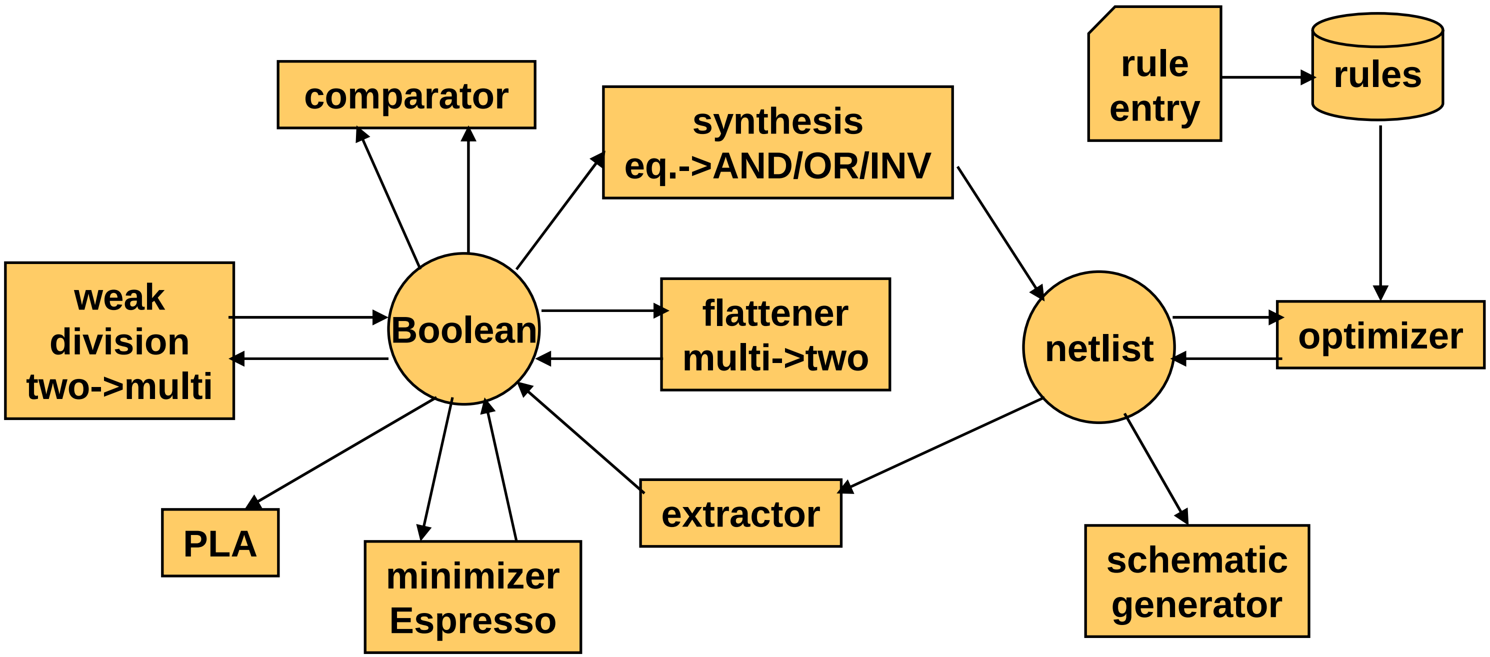
  complexity = breadth^{depth}

  breadth = #gates * #rules

  ex) 100 gates,
  20 rules,
  look-ahead two rule applications
  --> branching factor b = 100*20 = 2,000
  depth d = 2
  --> complexity = 2,000^2 = 4,000,000
  --> prune the tree

- **Pruning**
  - **Limit to first B applicable rules**
  - **Limit the depth to D**
    --> rule application depth = D_{app}
  - **Limit the size of the neighborhood**
    --> only search mutually exclusive transformations
- **Metarules**
  - **Look-ahead is more useful in later phases**
    --> Dynamically vary parameters

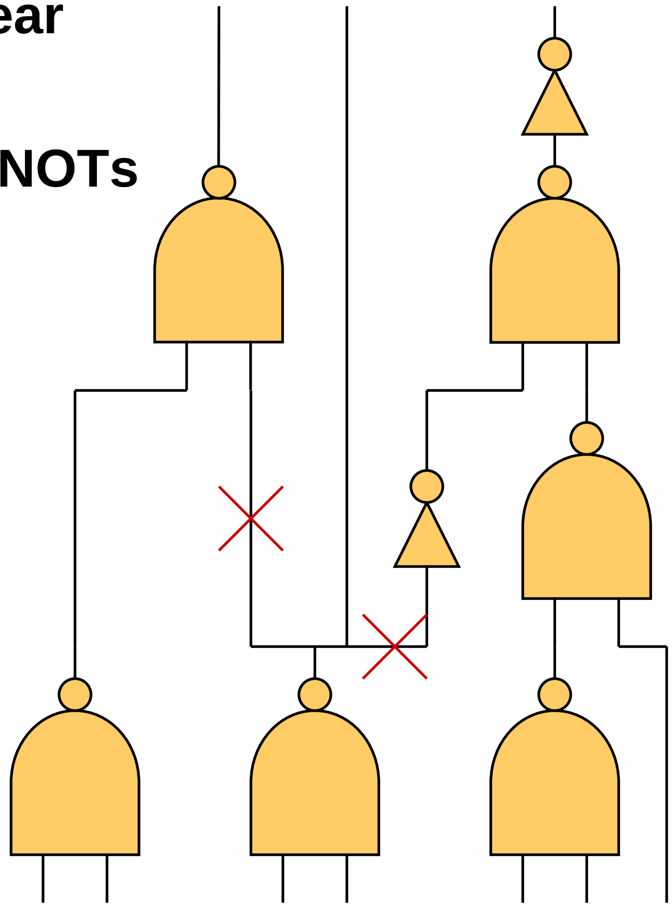
- **Rule Entry**
  - **Two netlists are extracted and compared**



Graph Covering

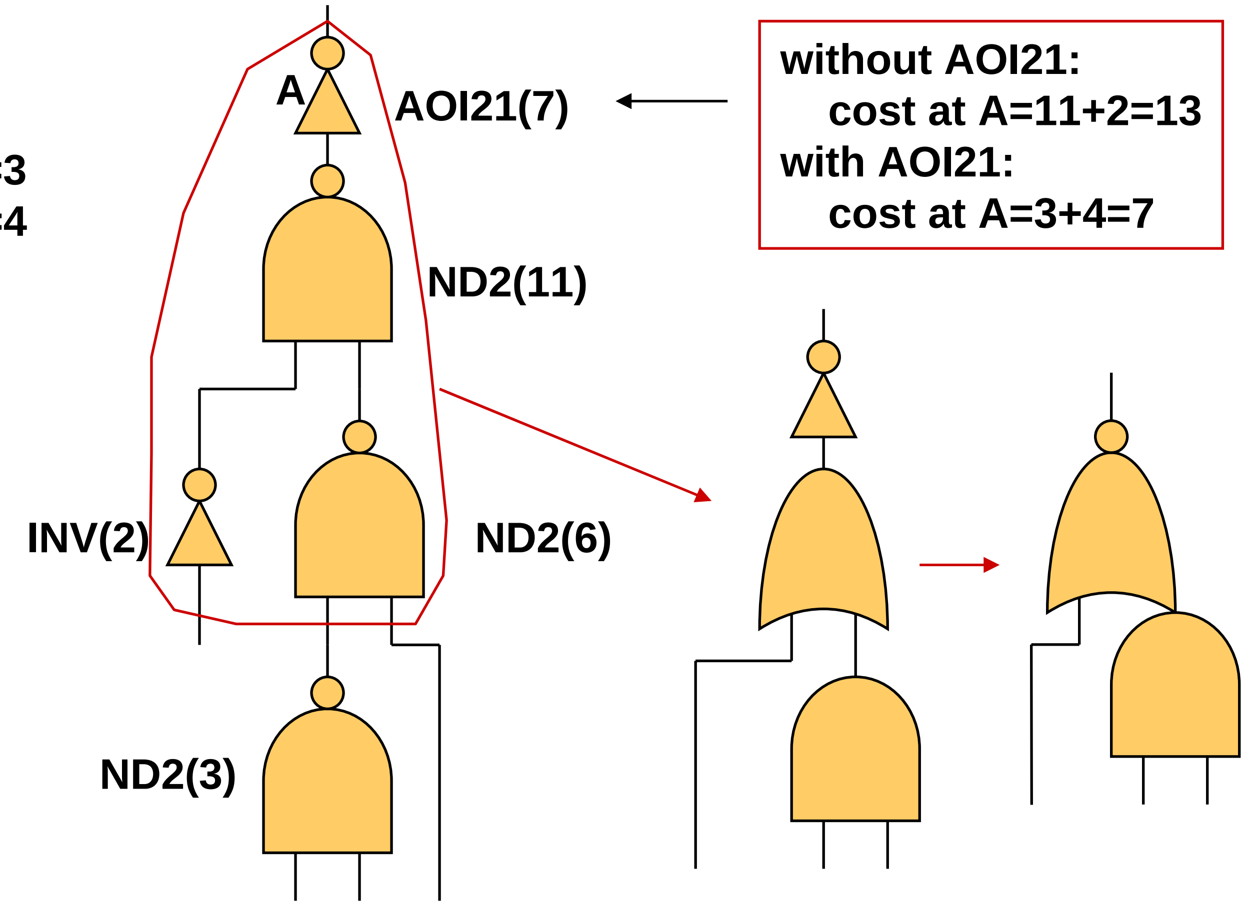
- **DAGON**
  - K. Keutzer, "DAGON: technology binding and local optimization by DAG matching," Proc. 24th Design Automation Conference, 1987
- **Problem**
  - Given
    - Boolean network (represented by a DAG)
    - Library (each cell is a DAG with a cost)
  - Find minimal cost covering of the Boolean network
    - Requires DAG matching (NP-complete)

- **Simplification**
  - **Represent a network by a forest by partitioning DAG**
  - **Partitioning**
    - **If a node has fanout greater than one, cut the graph there**
    - **Gate with fanout>1 becomes a root**
  - **Complexity of the partitioning: linear**
  - **Represent library cells by trees**
  - **Use canonical forms: NANDs and NOTs**
  - **Match trees by trees**
  - **Dynamic programming for finding a minimal cost match**



- **Tree Matching**
  - **Compute all matching at each node**
  - **Select best matching (depth-first)**
    - **Leaf: Cost of a NAND or a NOT**
    - **Internal node: Cost of matching tree + cost of sub-trees**
  - **Example**

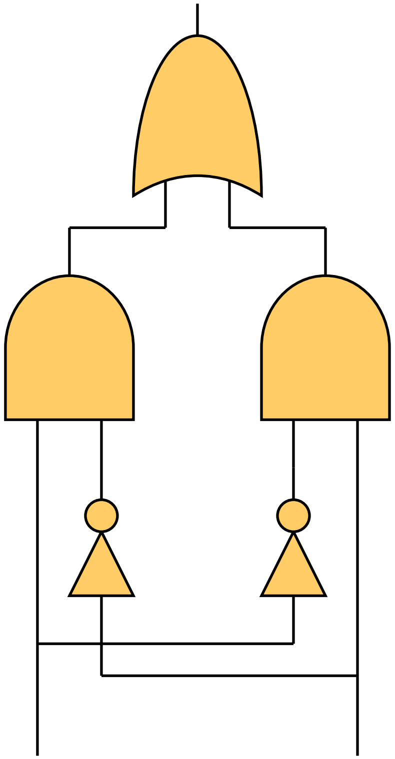
cost(INV)=2
cost(NAND)=3
cost(AOI21)=4



- **Weak Points**
  - **Partitioning network into trees**

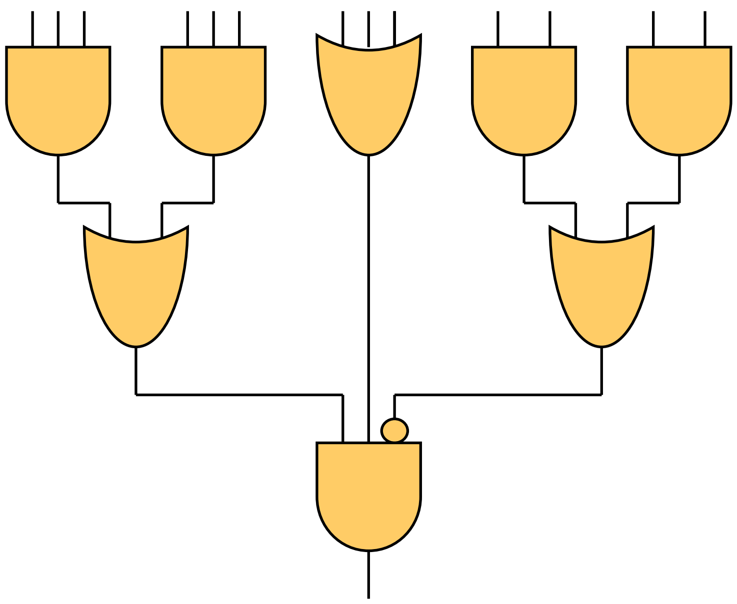
    --> local optimum
  - **Representing cells by trees**

    --> No way of representing XORs with a tree

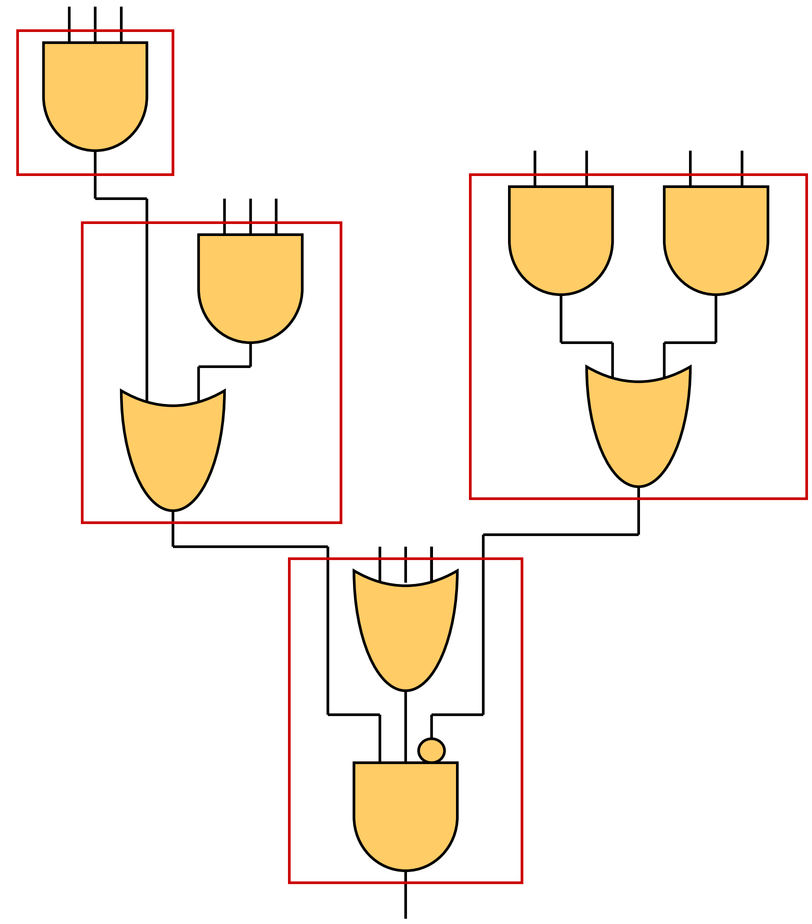


Technology Mapping for FPGAs

- **LUT-based FPGA**
  - **Mapping**

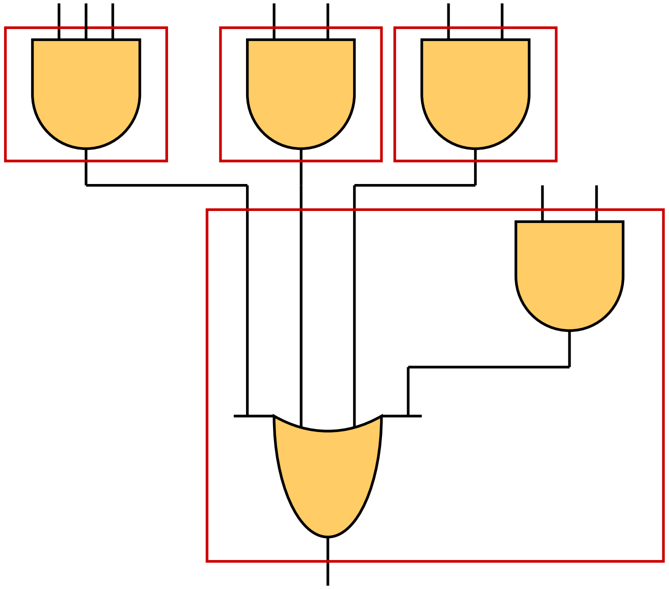


Boolean network

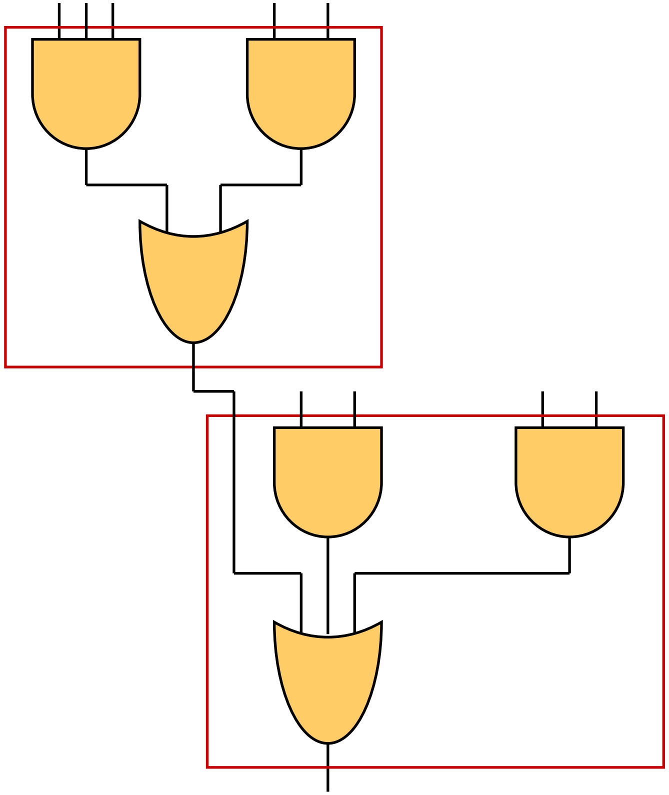


Mapping to 5-input LUTs

– Decomposition

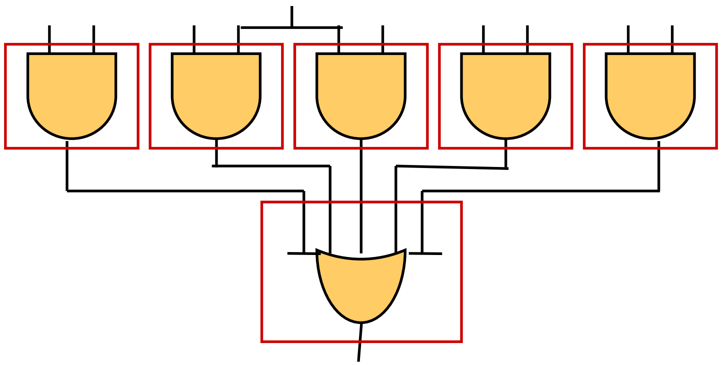


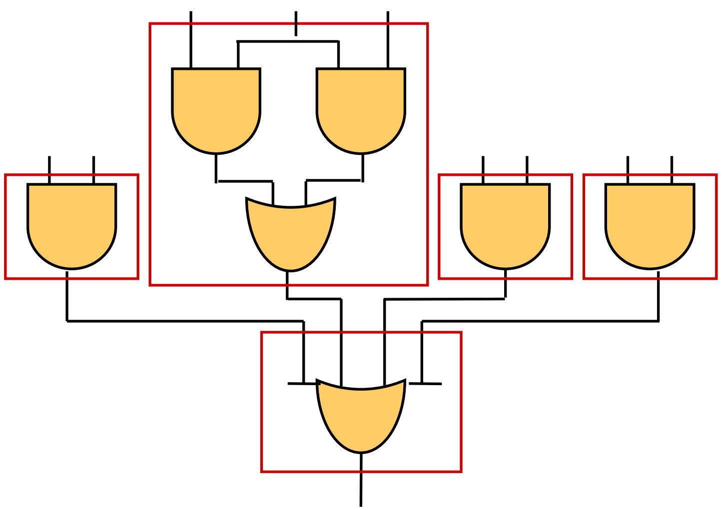
Without decomposition
4 LUTs



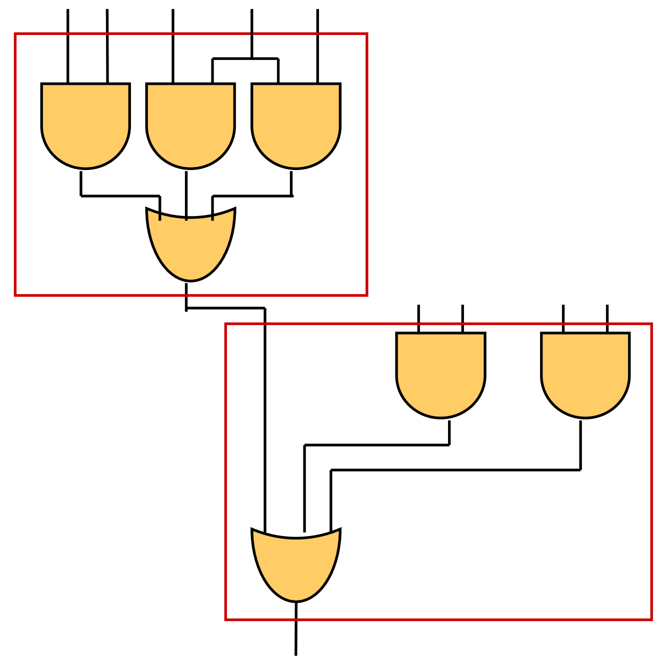
With decomposition
2 LUTs

– **Local reconvergent paths**

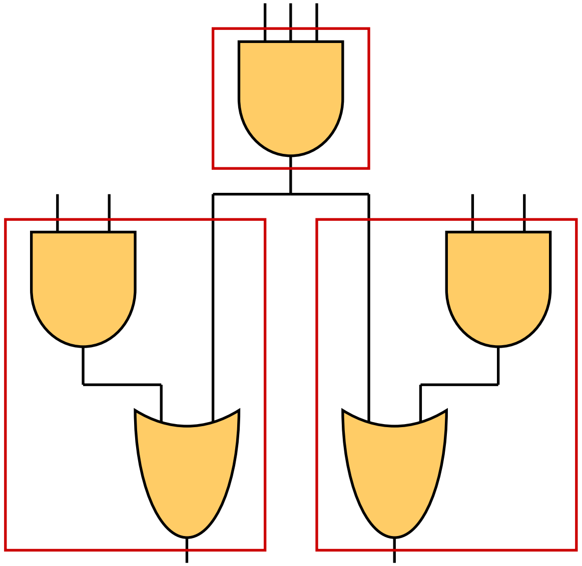




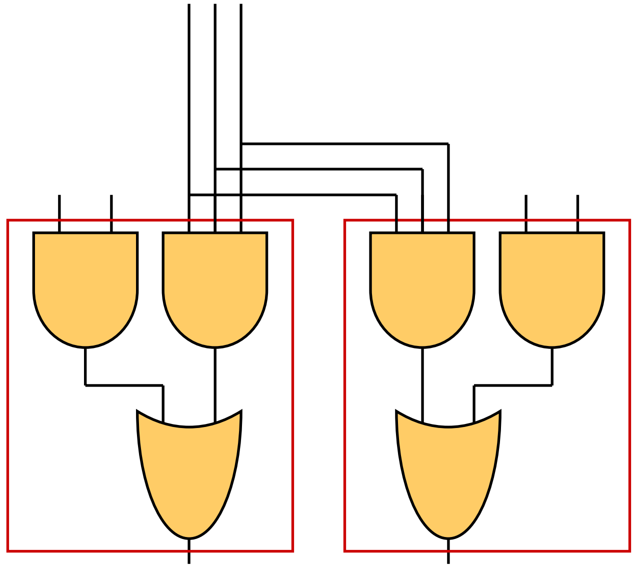
Reconvergent paths realized within one LUT



– **Replication of logic**



Without replicated logic
3 LUTs



With replicated logic
2 LUTs