Parallel and Reconfigurable VLSI Computing (4)

FPGA Synthesis Flow

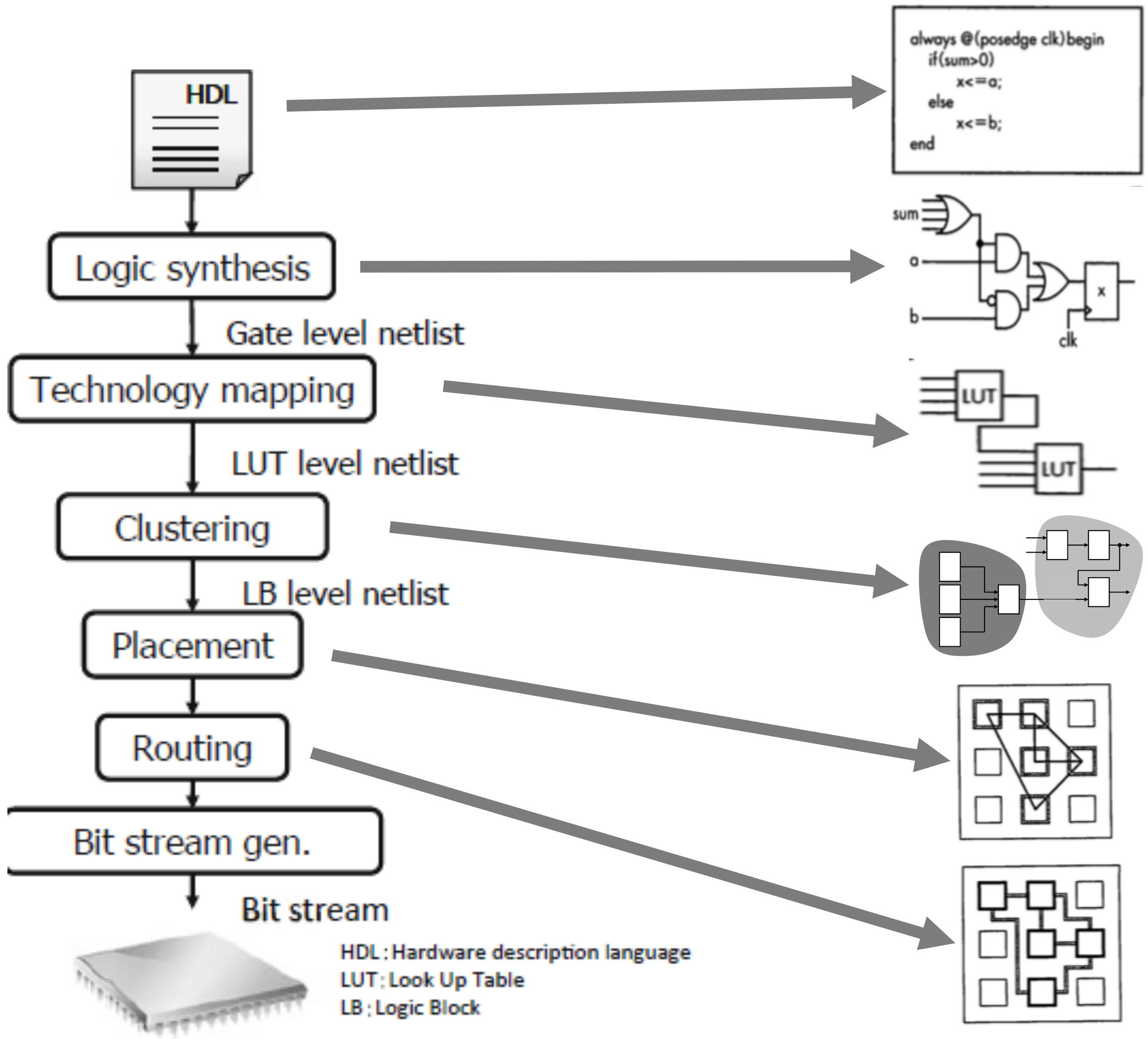
Hiroki Nakahara

Tokyo Institute of Technology

Outline

1. Synthesis Flow
2. Technology Mapping
3. Clustering
4. Place-and-Routing
5. Low Power Design
6. Conclusion

Synthesis Flow



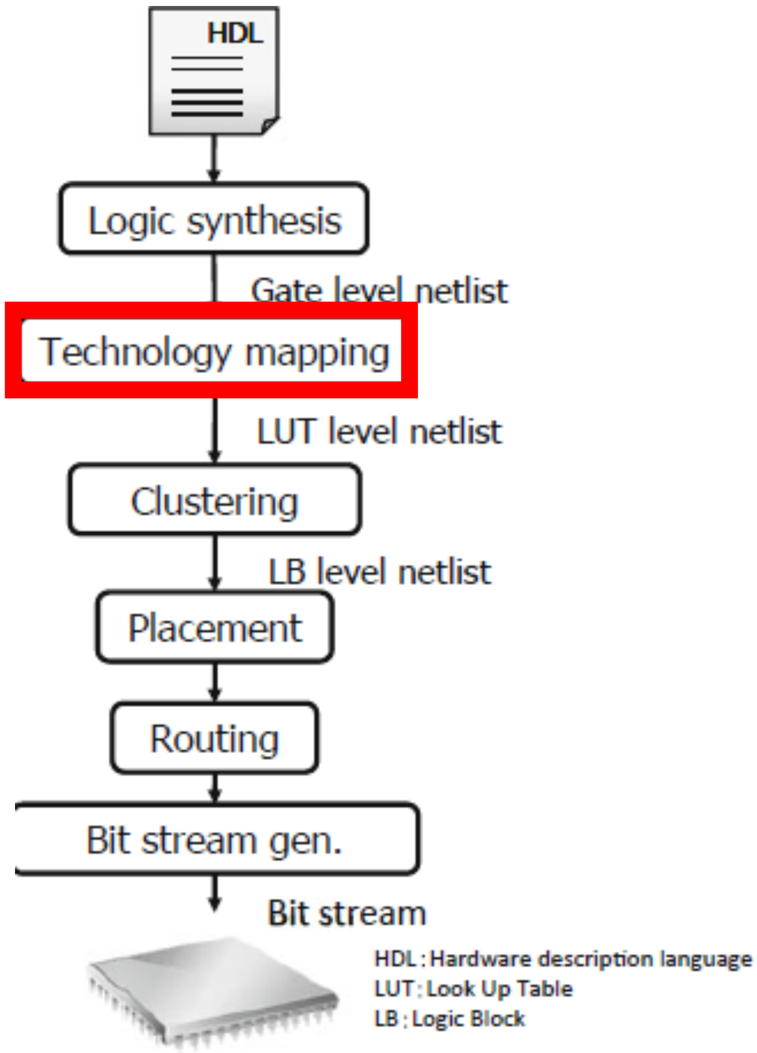
Related Work: VTR (Verilog-to-Routing) Project

- Open-source CAD tools for FPGA architecture and CAD research

https://github.com/verilog-to-routing/vtr-verilog-to-routing

- Enable the investigation of new FPGA architectures and CAD algorithms, which are not possible with closed-source tools
- The VTR design flow takes as input a Verilog description of a digital circuit, and a description of the target FPGA architecture
  - Elaboration & Synthesis (ODIN II)
  - Logic Optimization & Technology Mapping (ABC)
  - Packing, Placement, Routing & Timing Analysis (VPR)

Technology Mapping

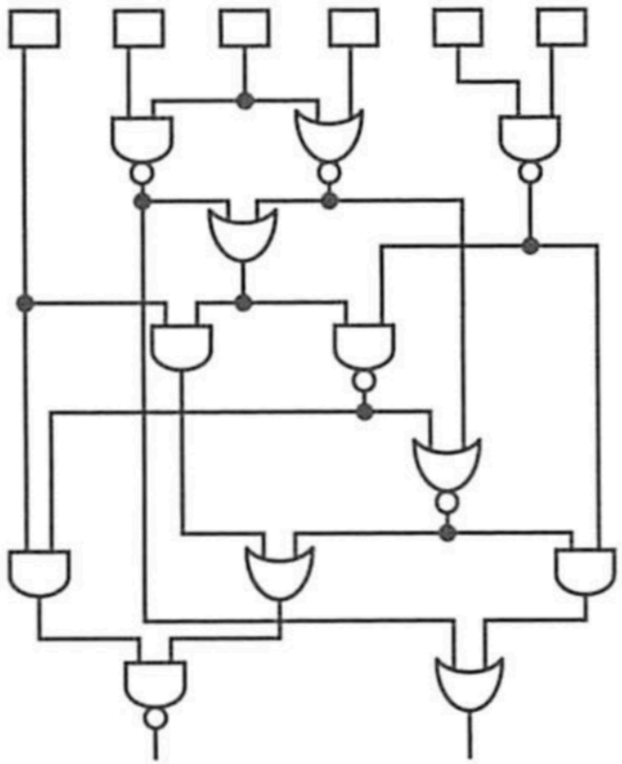


FlowMap

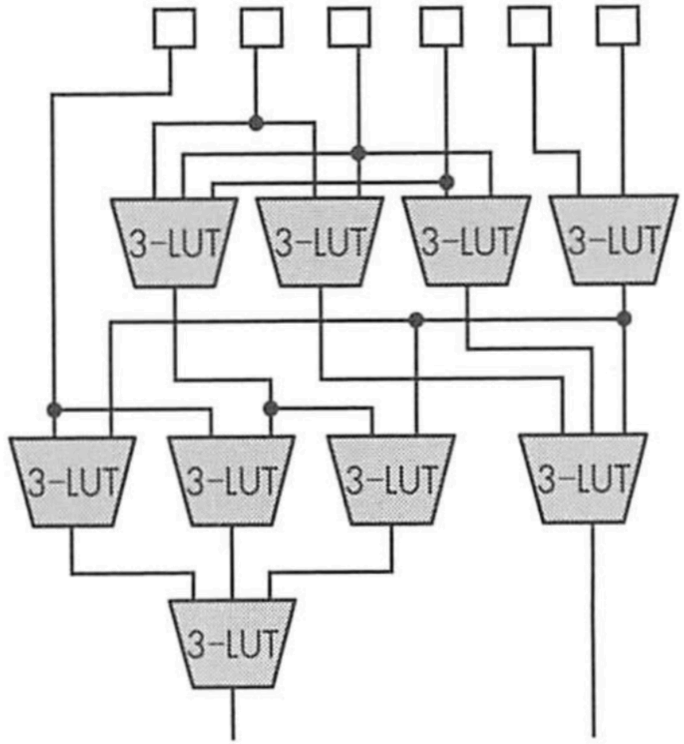
- Labeling and Cut
- Mapping

Technology Mapping

- Convert a given Boolean netlist into an LUT netlist

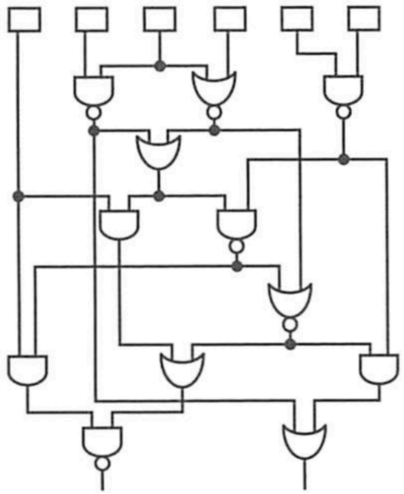


Boolean netlist

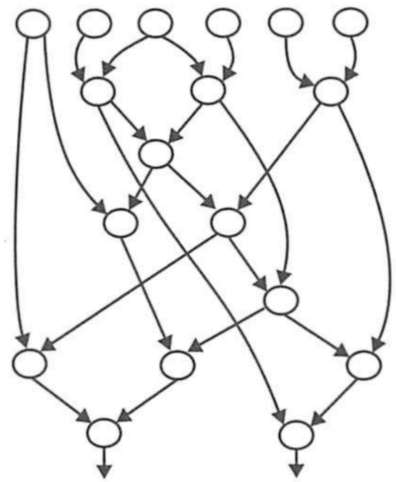


LUT netlist

FlowMap Process



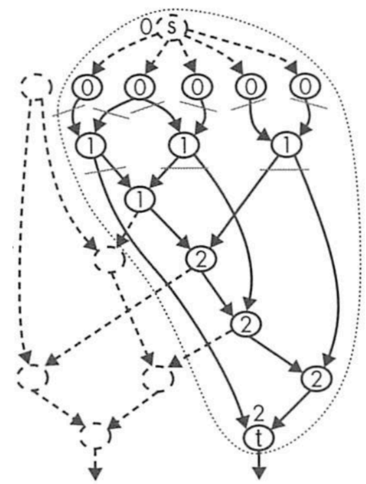
Boolean netlist



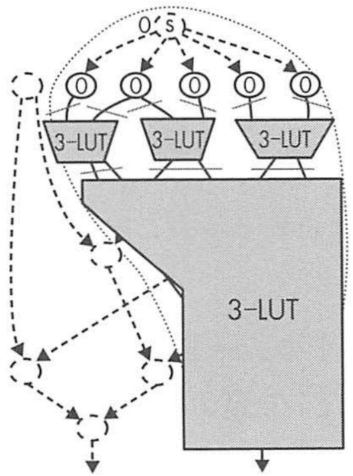
DAG

Labeling

Cut

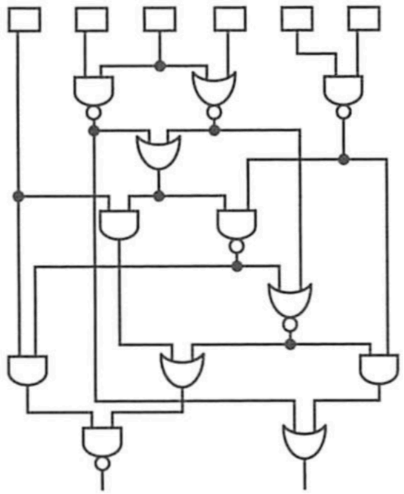


Mapping

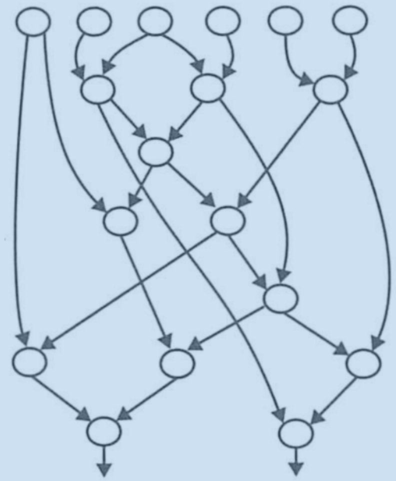


LUT netlist

FlowMap Process



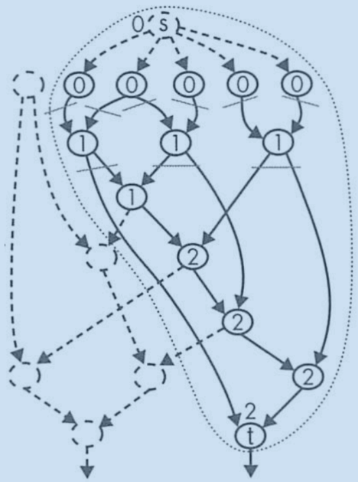
Boolean netlist



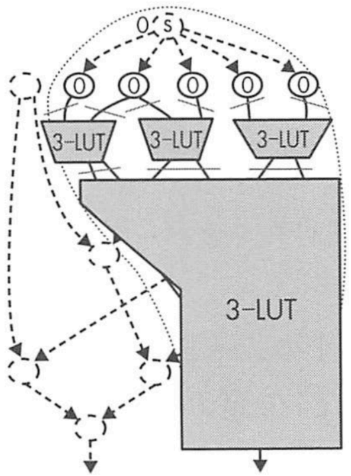
DAG

Labeling

Cut



Mapping



LUT netlist

FlowMap Algorithm

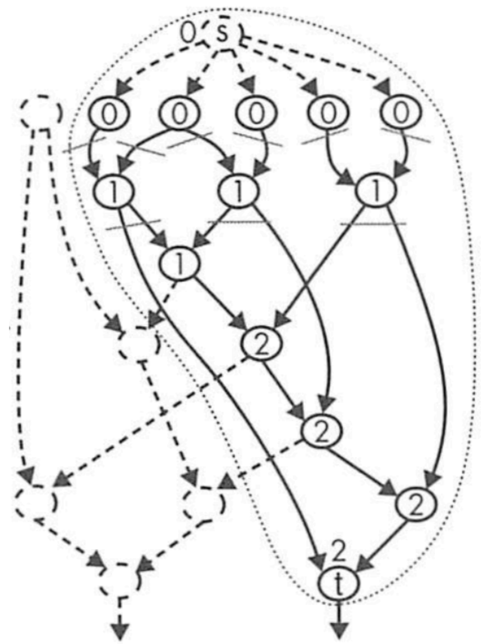
1) Extract a given Boolean network dependent on an output node t

2) Assign the input label to 0

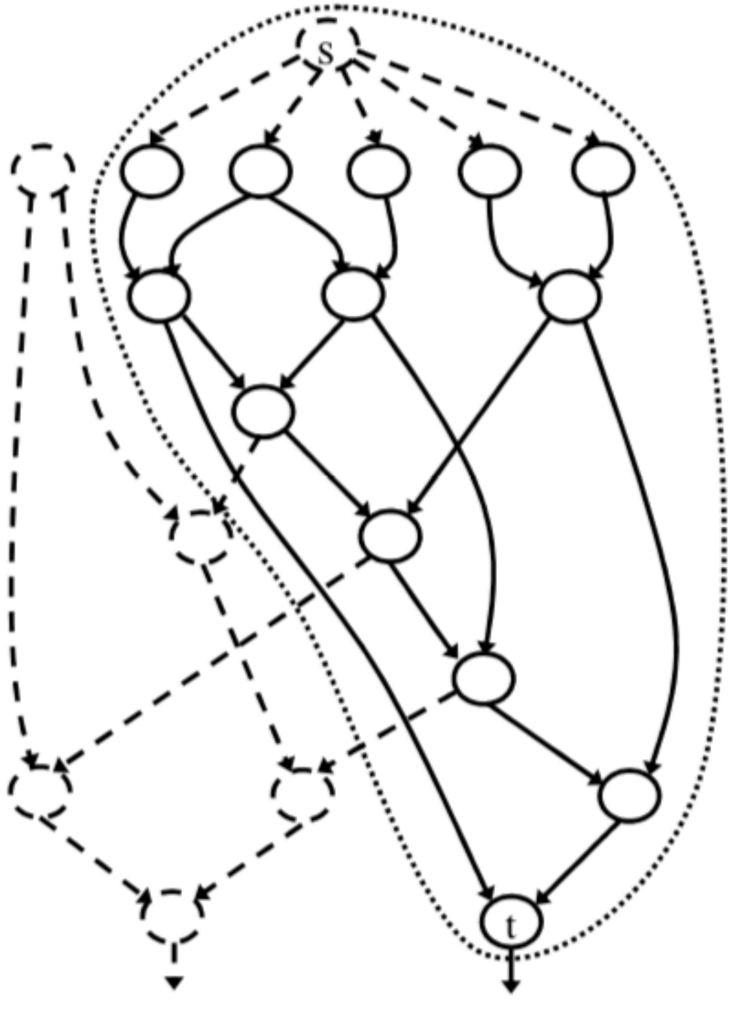
3) Label the node to which the already labeled node is input

4) Look for the range that can be covered by the k-LUT and place a cut on the input

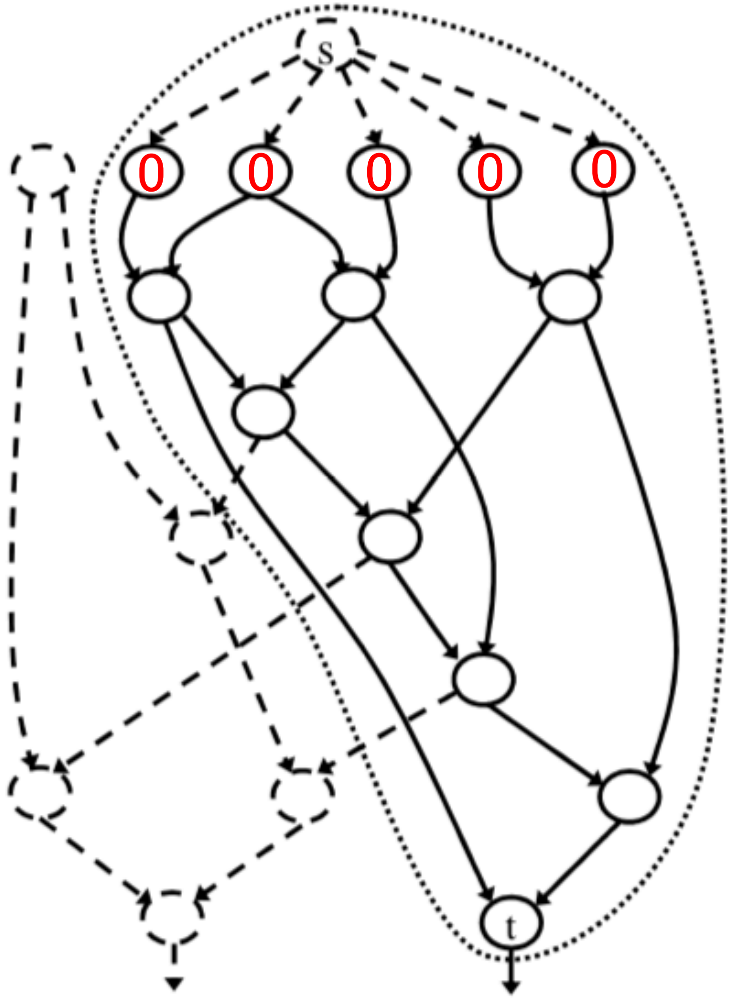
5) repeat 3) and 4)



Extract a Single-output Network

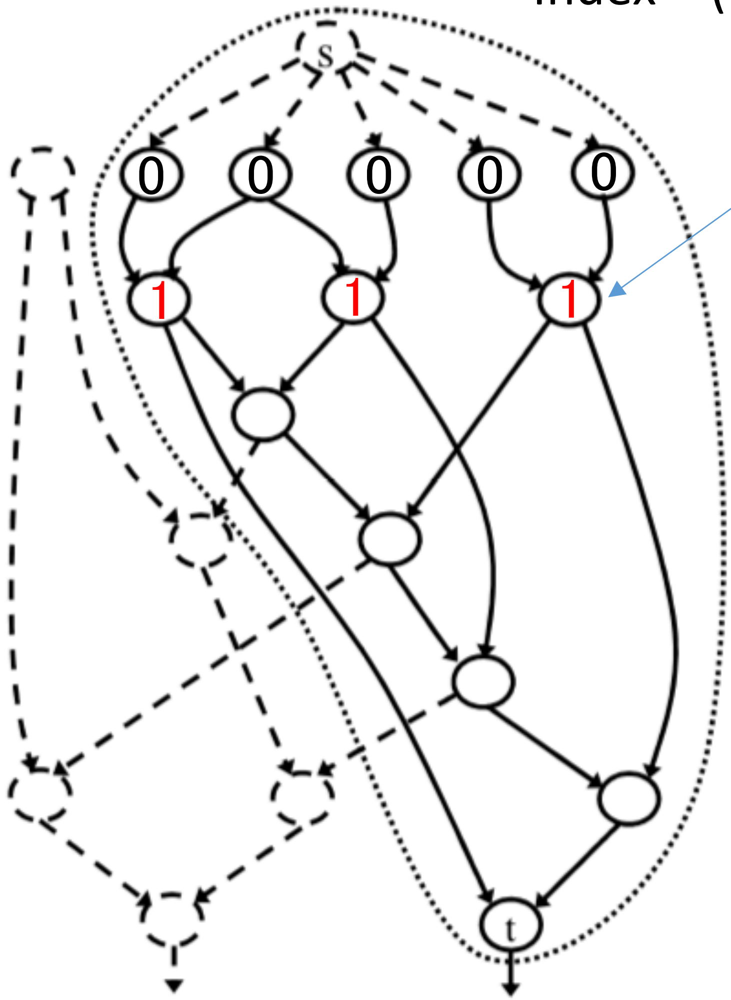


Assign the Input Label to 0

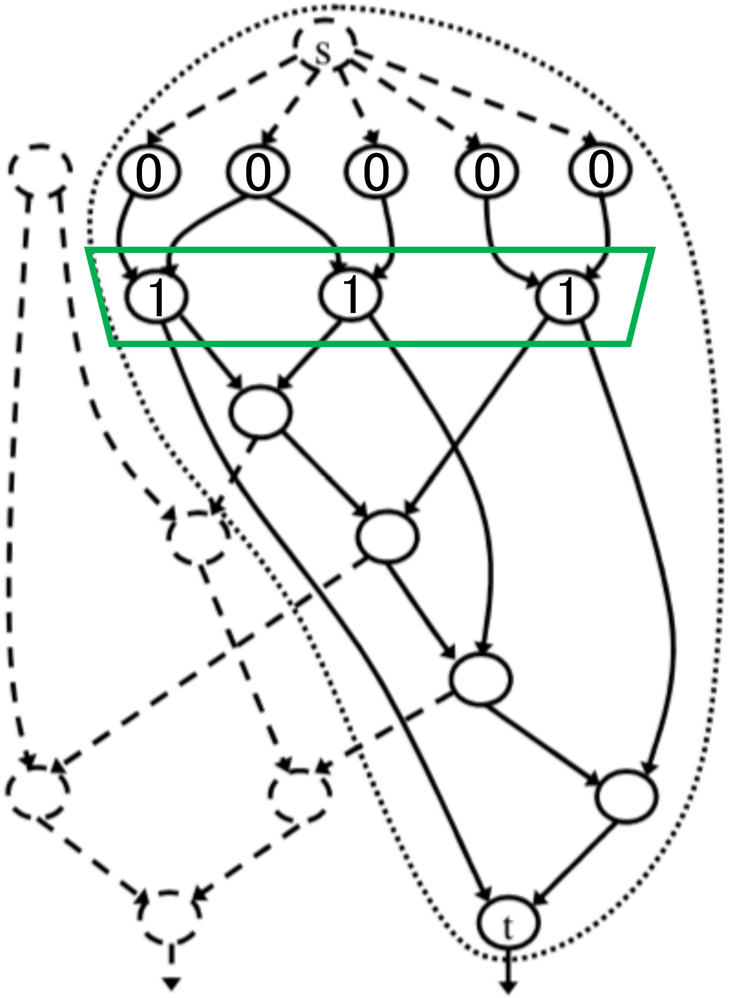


Labeling the Index by Topological Order

Index = (Max. of above the cut) + 1



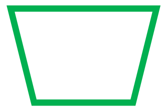
Covering by the k-Input LUT (k-LUT)



3−LUT:

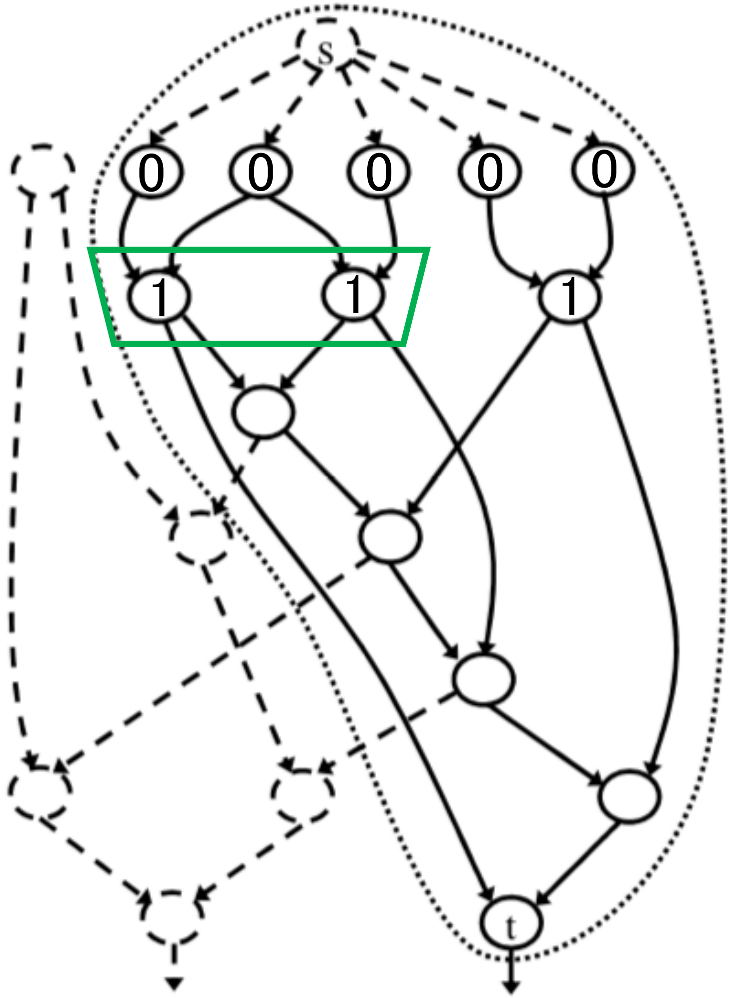
#in is less than 4,

and #out is one.

 is infeasible,

since #in=5 and #out=3.

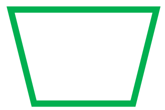
Covering by the k-Input LUT (k-LUT)



3−LUT：

#in is less than 4,

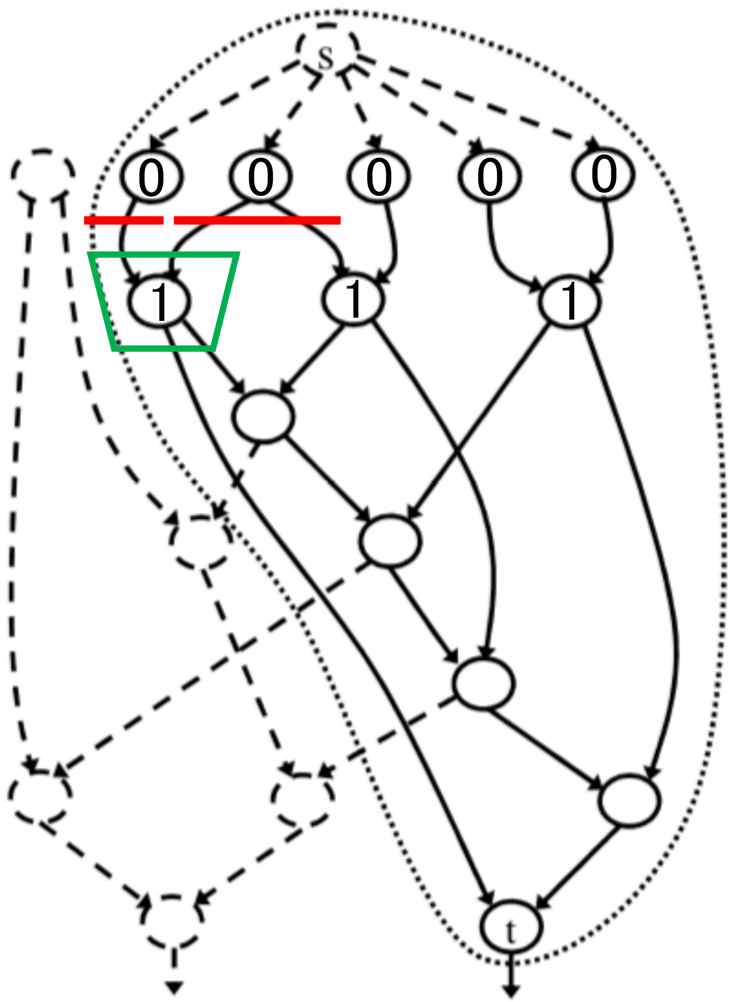
and #out is one.



is infeasible,

since #in=3 and #out=2.

Covering by the k-Input LUT (k-LUT)



3−LUT：

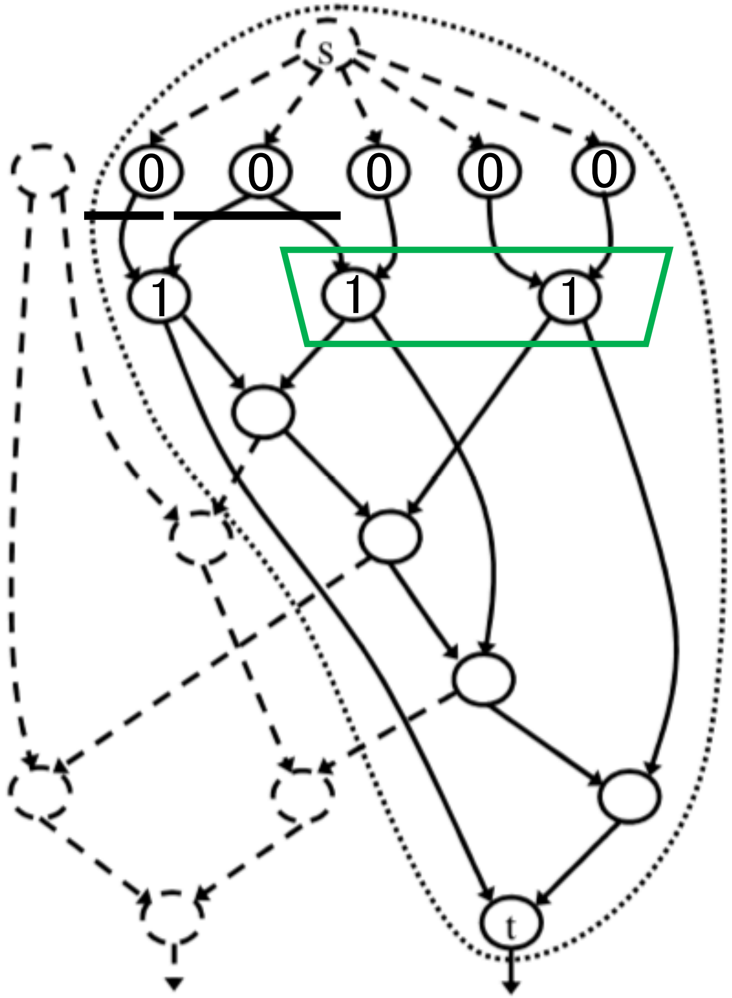
#in is less than 4,

and #out is one.

is feasible,

since #in=2 and #out=1.

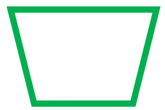
Covering by the k-Input LUT (k-LUT)



3−LUT :

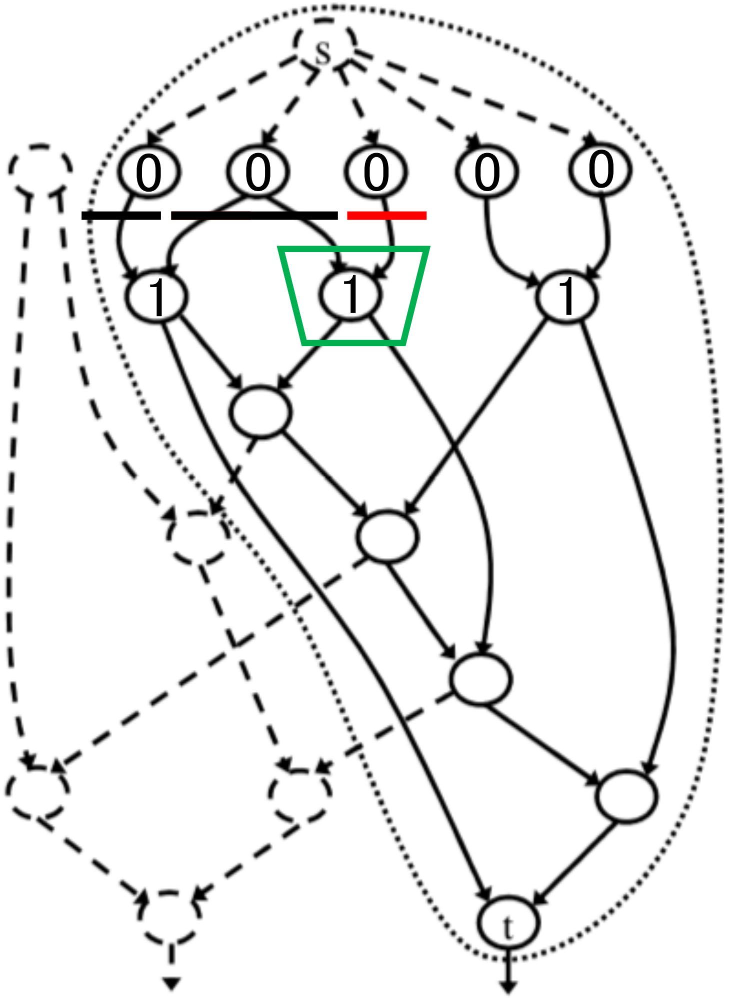
#in is less than 4,

and #out is one.

 is infeasible,

since #in=4 and #out=2.

Covering by the k-Input LUT (k-LUT)



3−LUT：

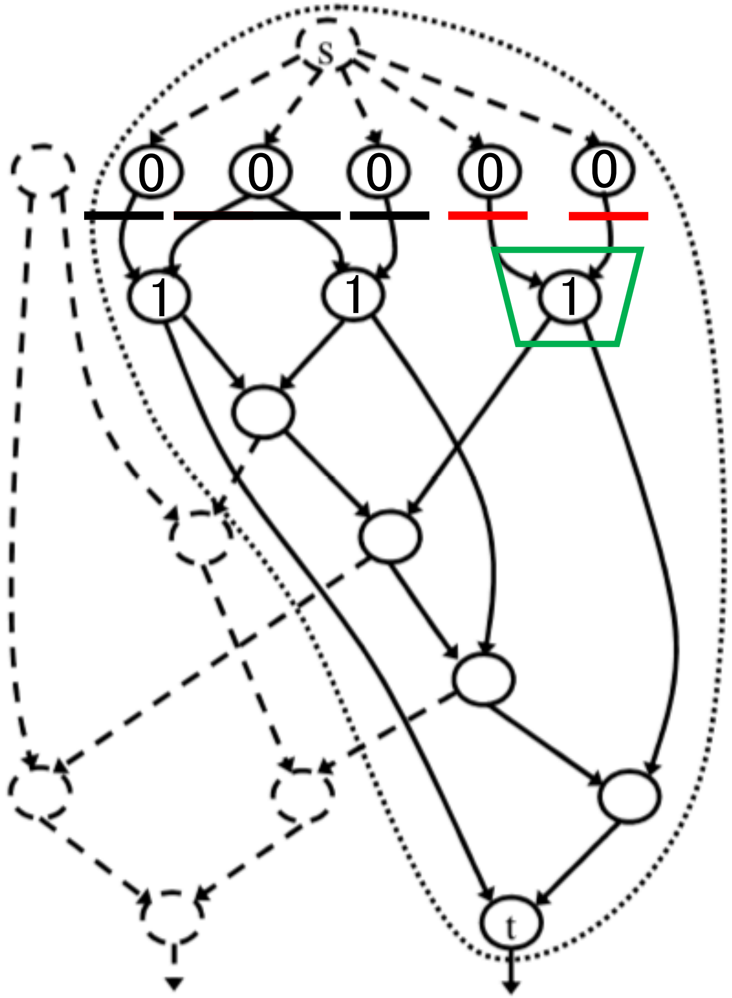
#in is less than 4,

and #out is one.

is feasible,

since #in=2 and #out=1.

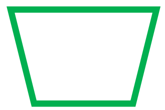
Covering by the k-Input LUT (k-LUT)



3−LUT :

#in is less than 4,

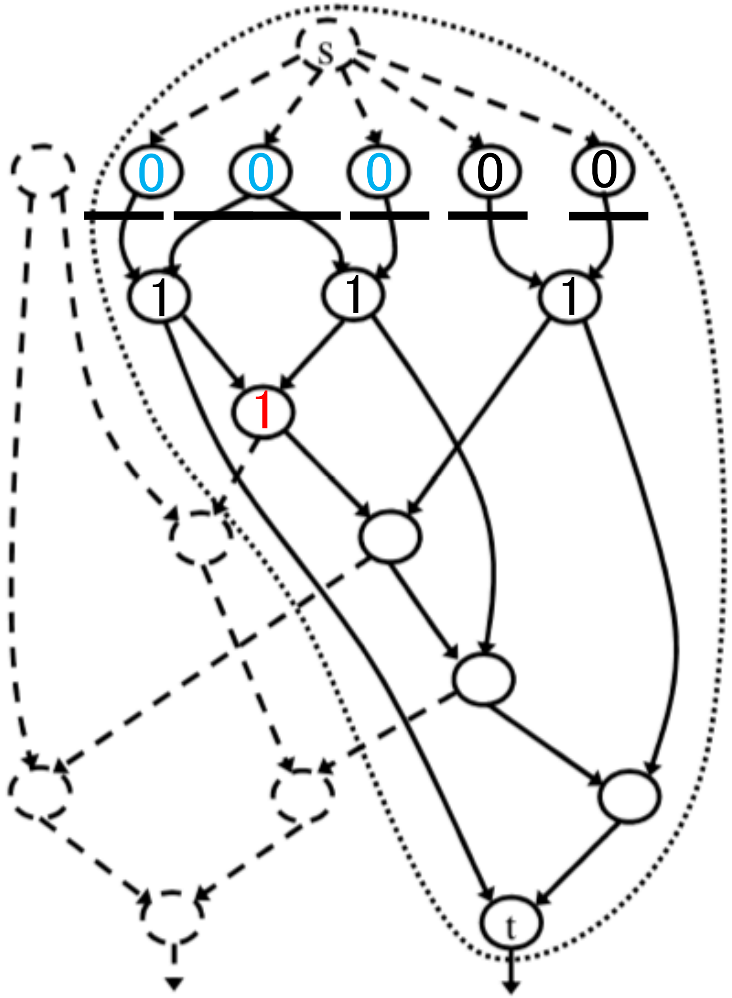
and #out is one.

 is feasible,

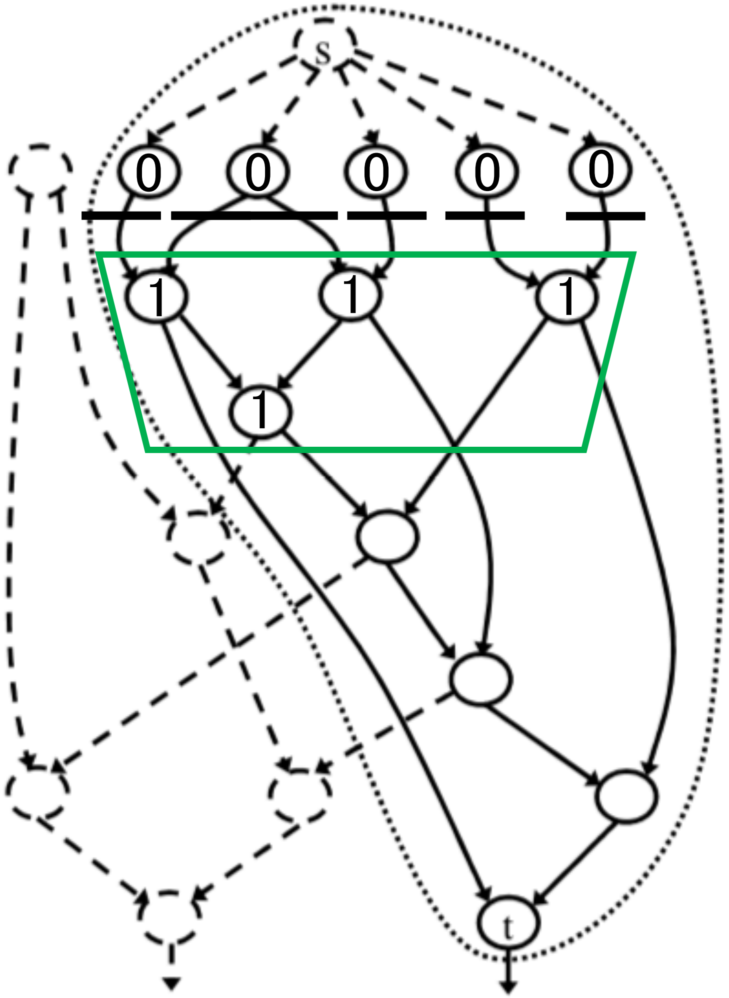
since #in=2 and #out=1.

Labeling

- Calculate the index of the node whose input is the already assigned



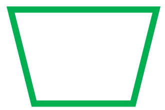
Covering by the k-Input LUT (k-LUT)



3−LUT :

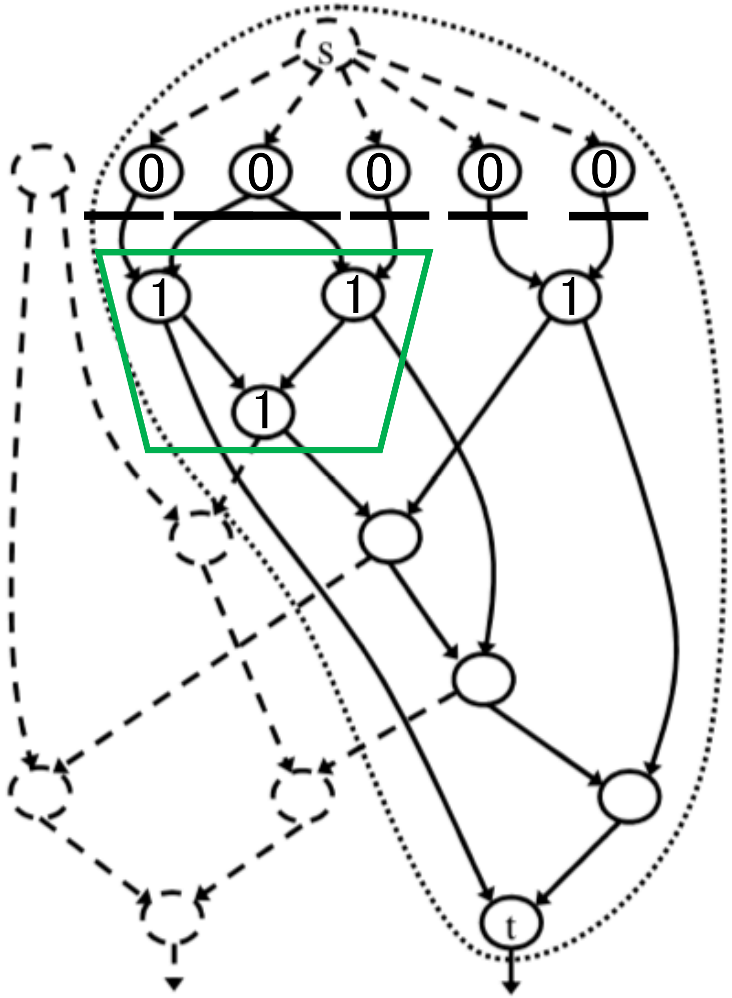
#in is less than 4,

and #out is one.

 is infeasible,

since #in=5 and #out=4.

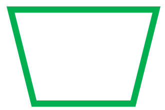
Covering by the k-Input LUT (k-LUT)



3−LUT：

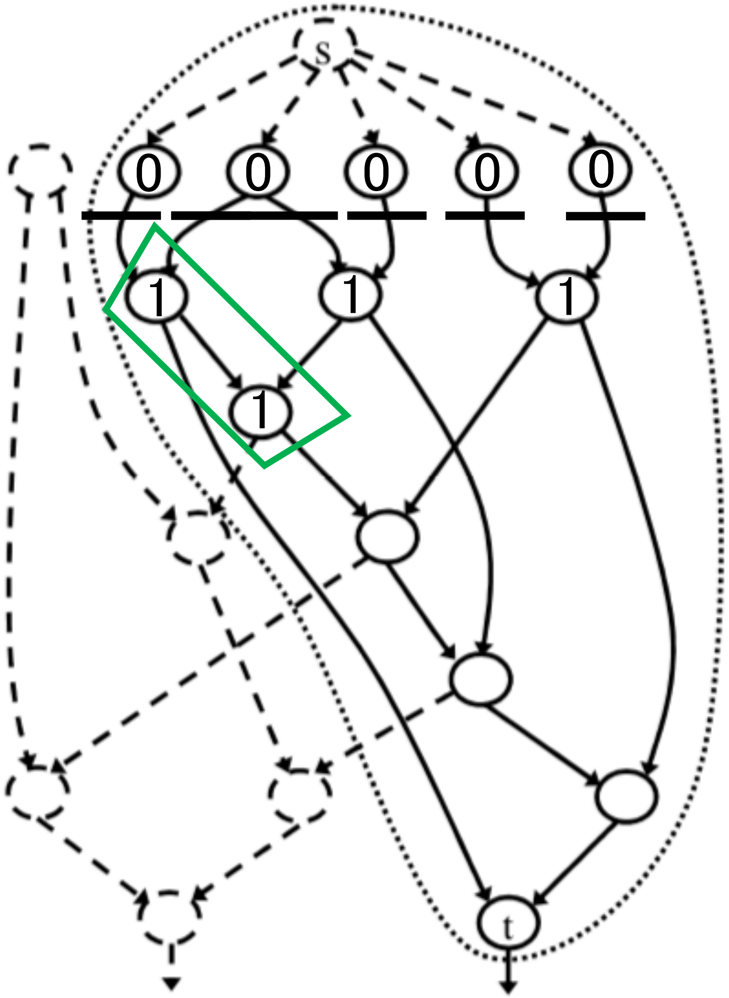
#in is less than 4,

and #out is one.

 is infeasible,

since #in=3 and #out=3.

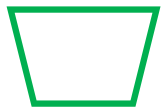
Covering by the k-Input LUT (k-LUT)



3−LUT :

#in is less than 4,

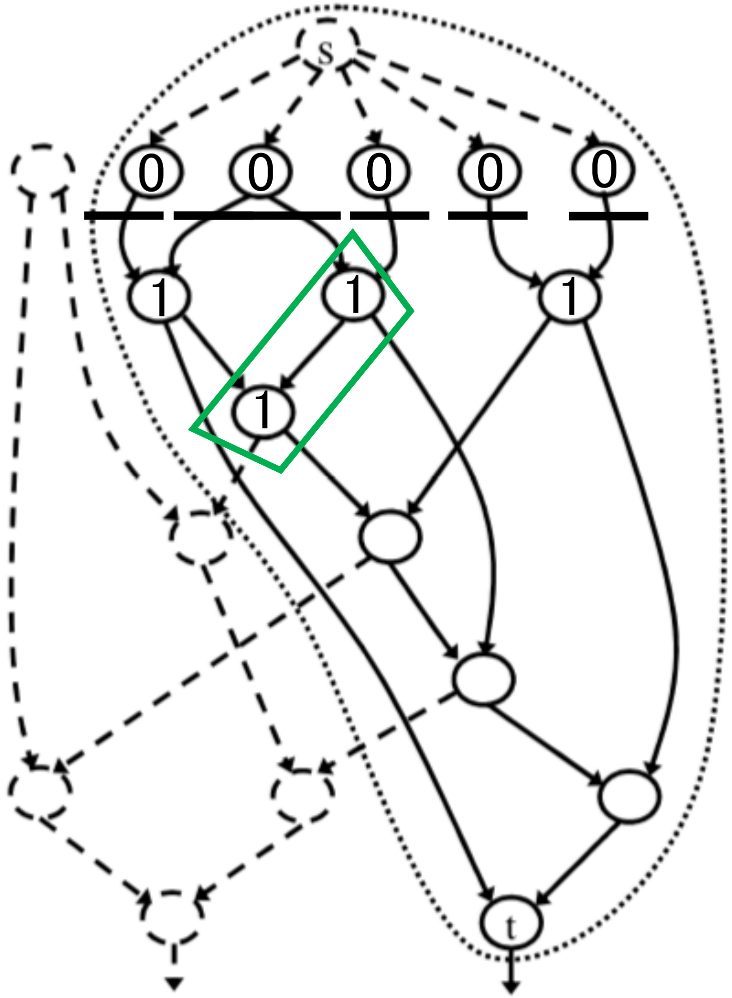
and #out is one.



is infeasible,

since #in=3 and #out=2.

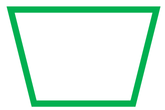
Covering by the k-Input LUT (k-LUT)



3−LUT：

#in is less than 4,

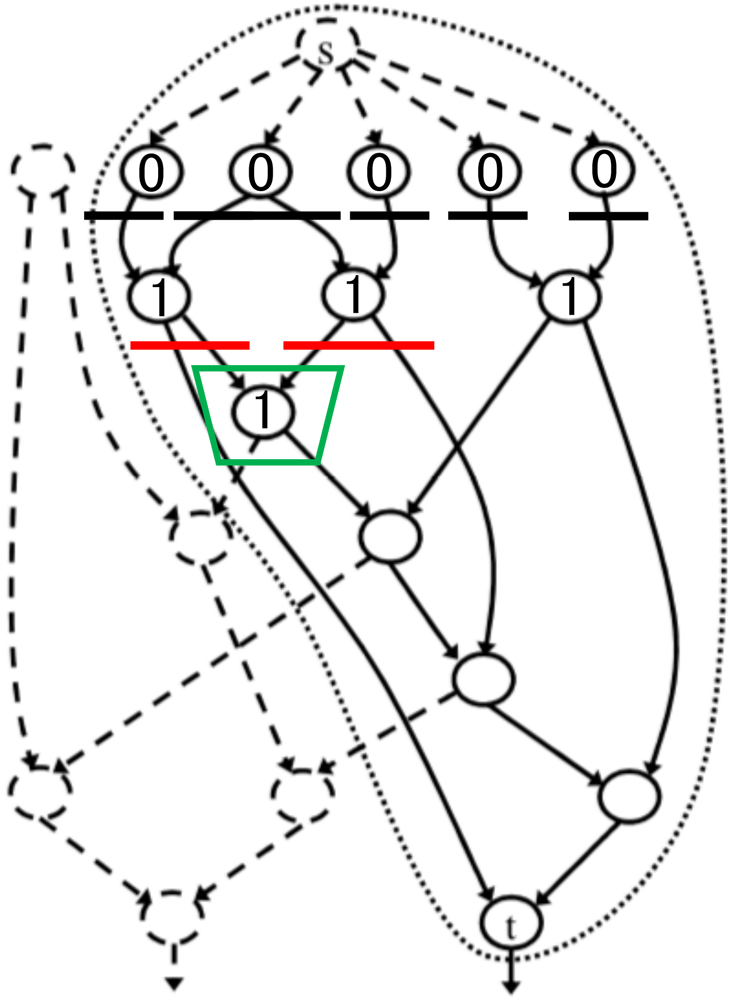
and #out is one.



is infeasible,

since #in=3 and #out=2.

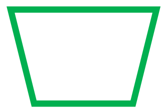
Covering by the k-Input LUT (k-LUT)



3−LUT：

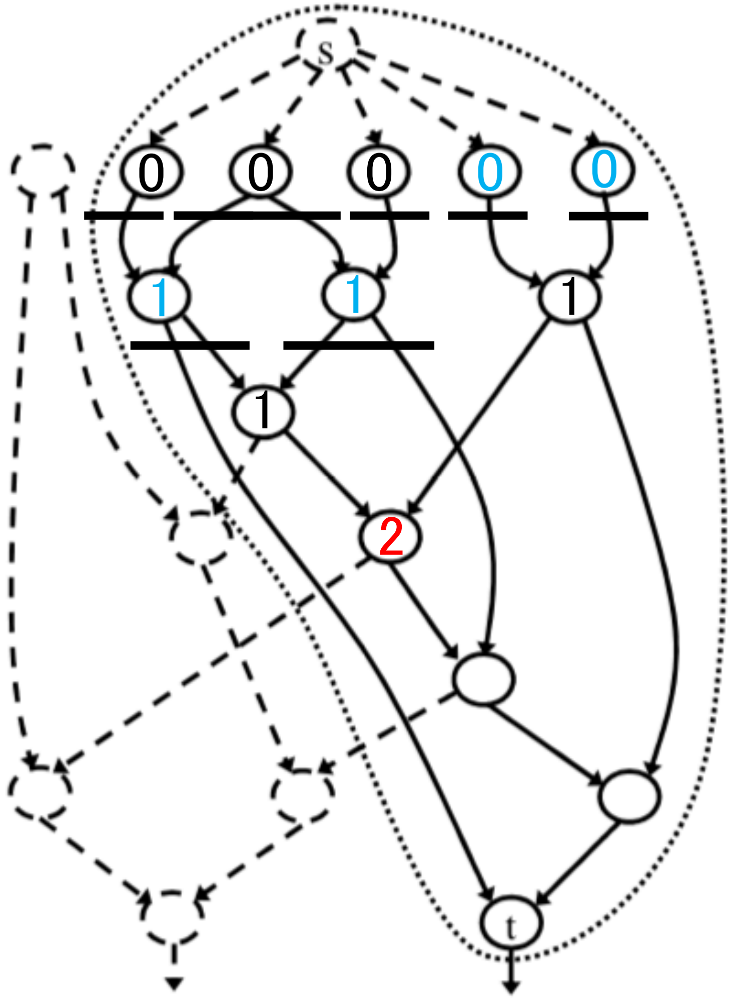
#in is less than 4,

and #out is one.

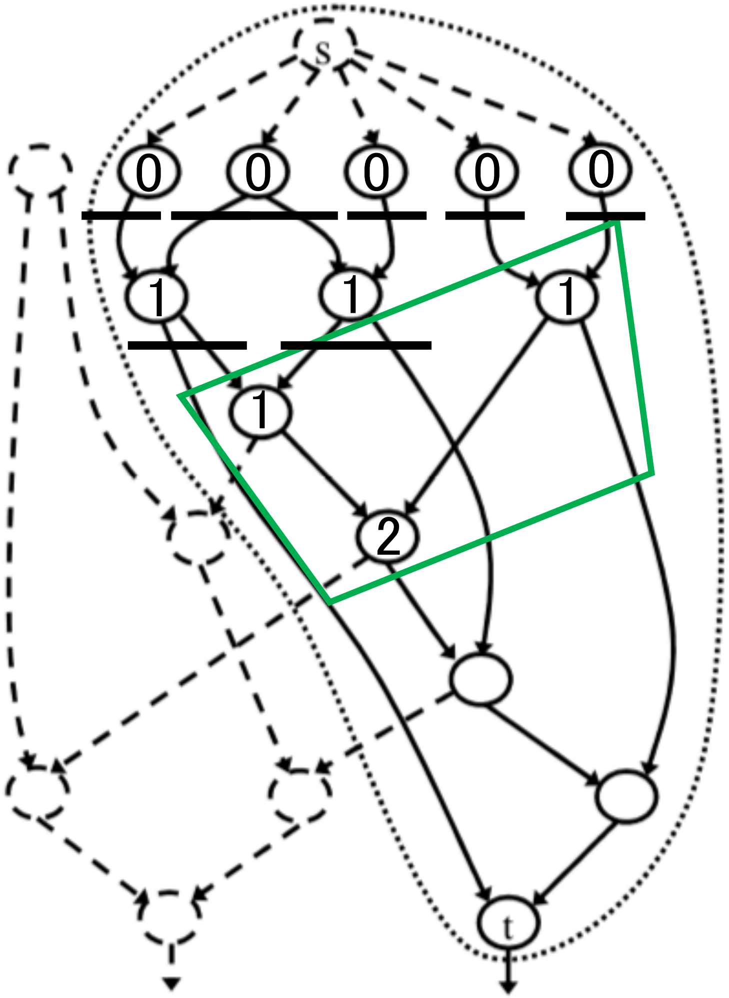
 is feasible,

since #in=2 and #out=1.

Update Labeling Index



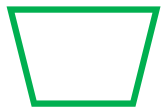
Covering by the k-Input LUT (k-LUT)



3−LUT：

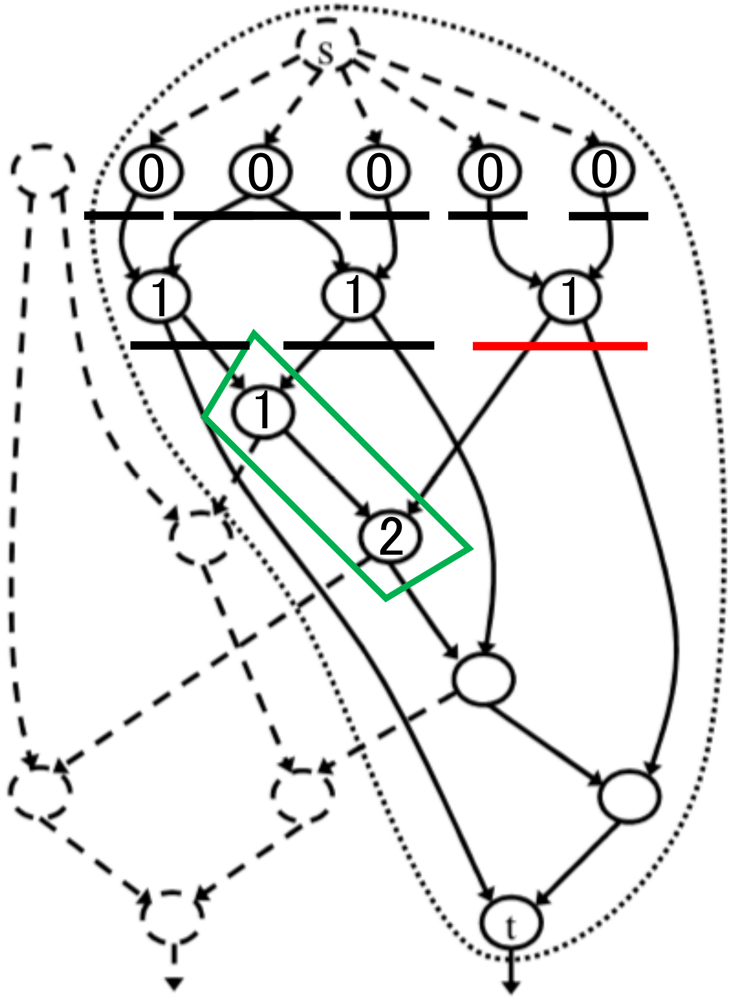
#in is less than 4,

and #out is one.

 is infeasible,

since #in=4 and #out=3.

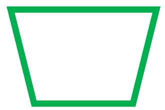
Covering by the k-Input LUT (k-LUT)



3−LUT：

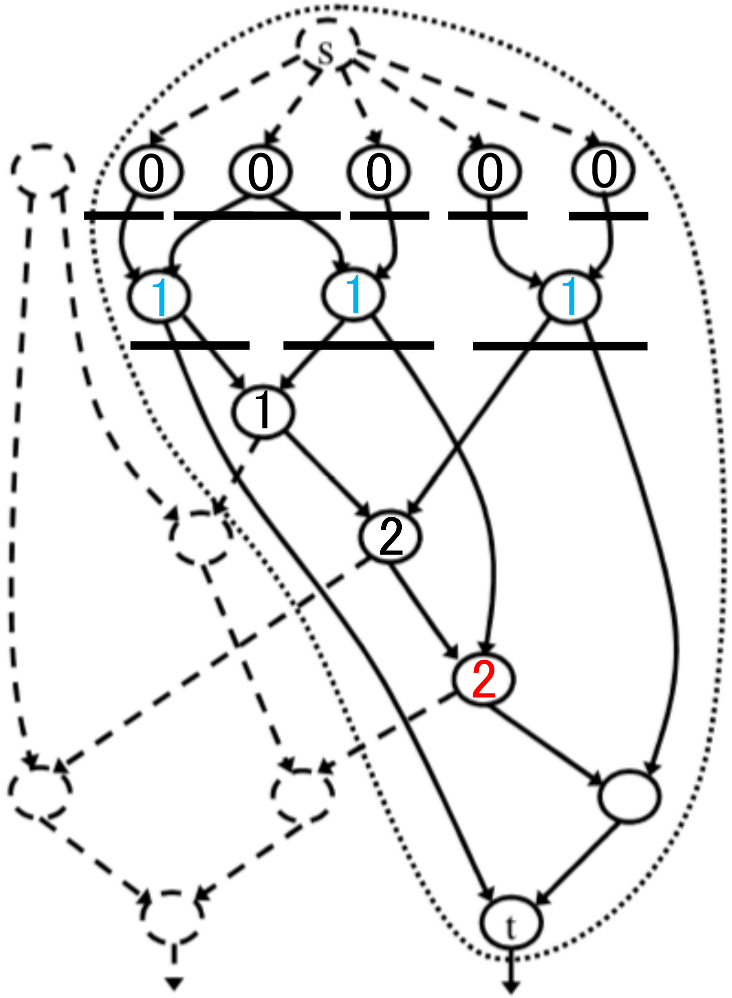
#in is less than 4,

and #out is one.

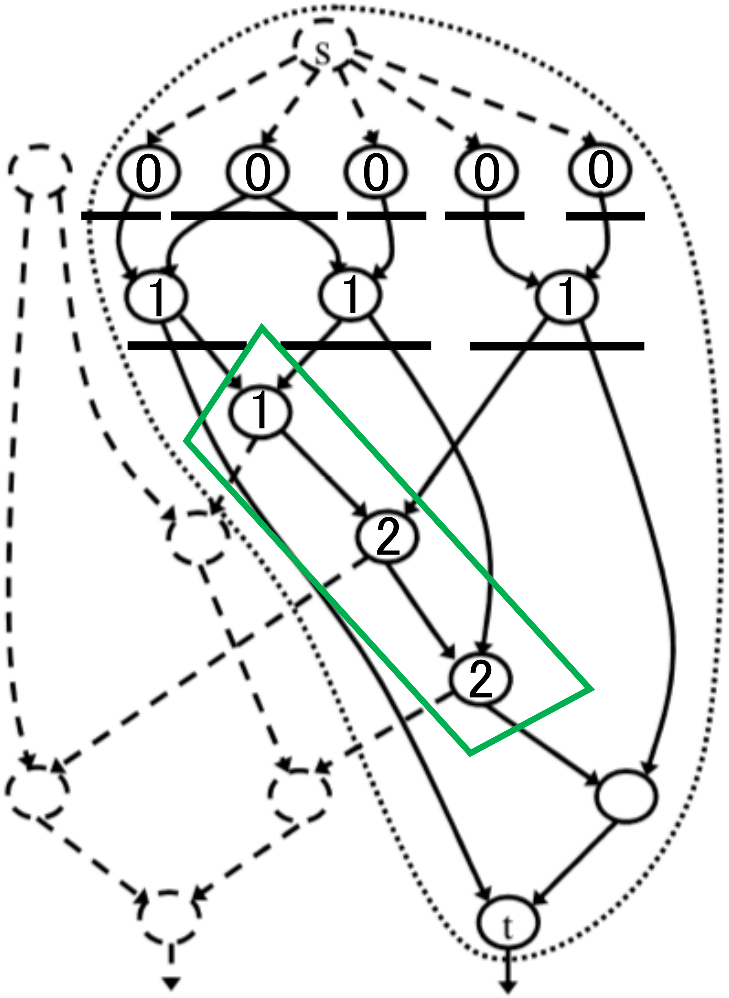
 is feasible,

since #in=3 and #out=1.

Update Labeling Index



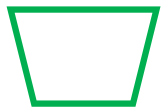
Covering by the k-Input LUT (k-LUT)



3−LUT：

#in is less than 4,

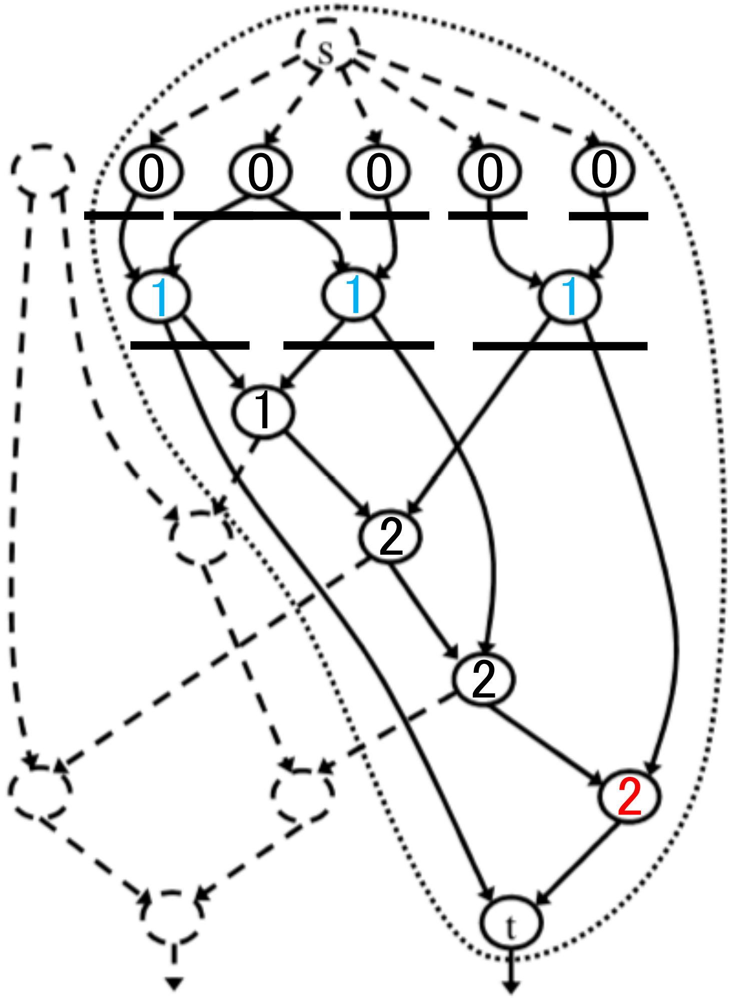
and #out is one.



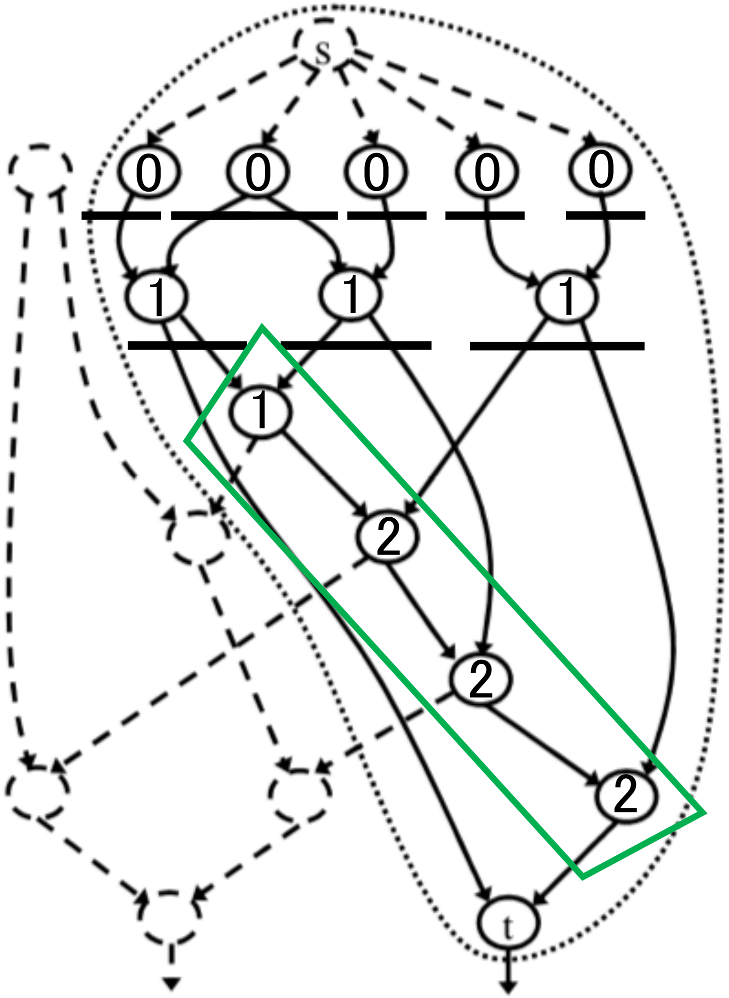
is feasible,

since #in=3 and #out=1.

Update Labeling Index



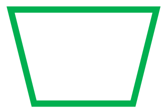
Covering by the k-Input LUT (k-LUT)



3−LUT：

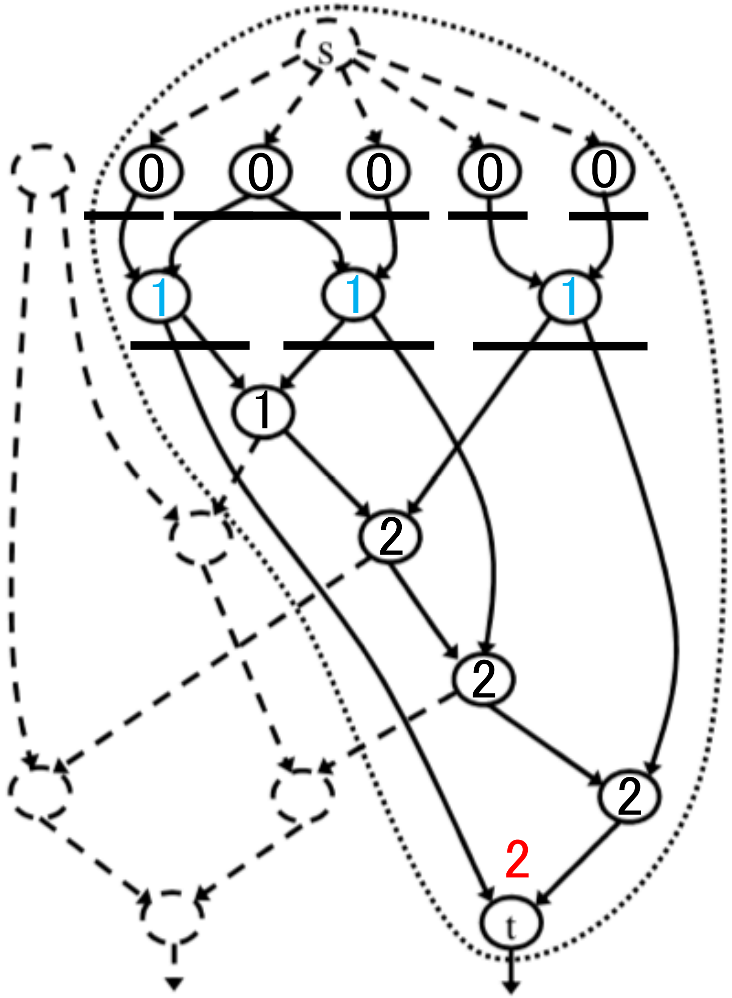
#in is less than 4,

and #out is one.

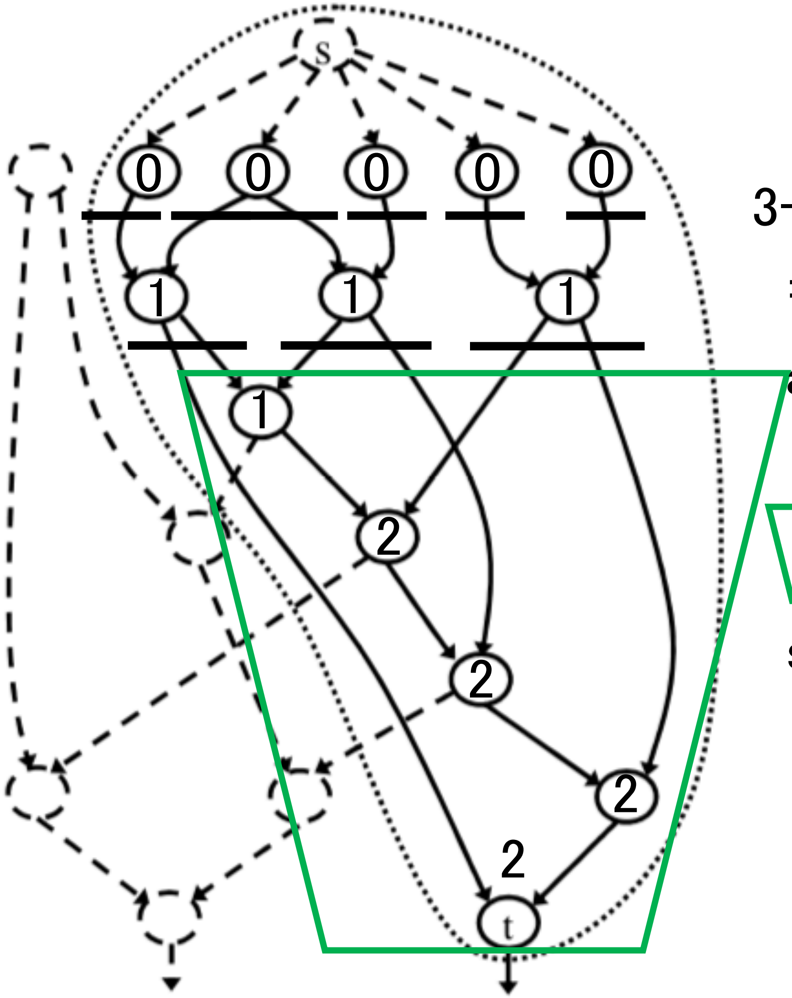
 is feasible,

since #in=3 and #out=1.

Update Labeling Index



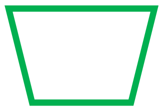
Covering by the k-Input LUT (k-LUT)



3−LUT:

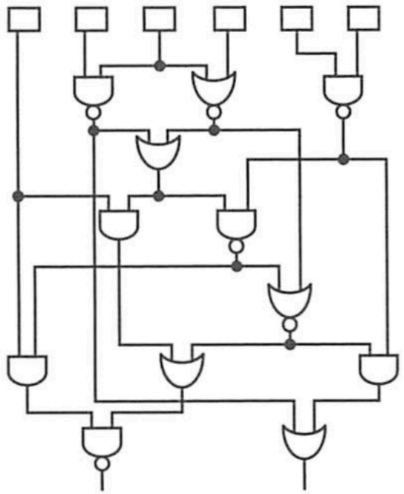
#in is less than 4,

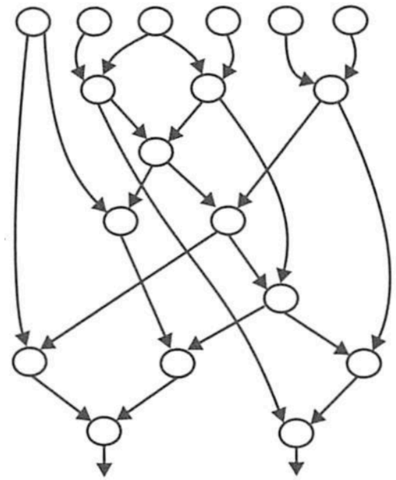
and #out is one.

 is feasible,

since #in=3 and #out=1.

FlowMap Process



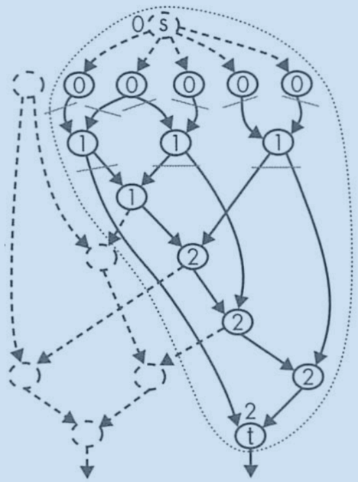


Boolean netlist

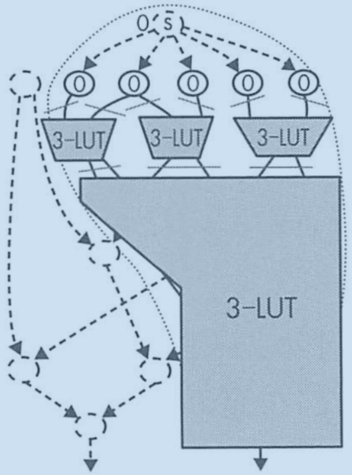
DAG

Labeling

Cut

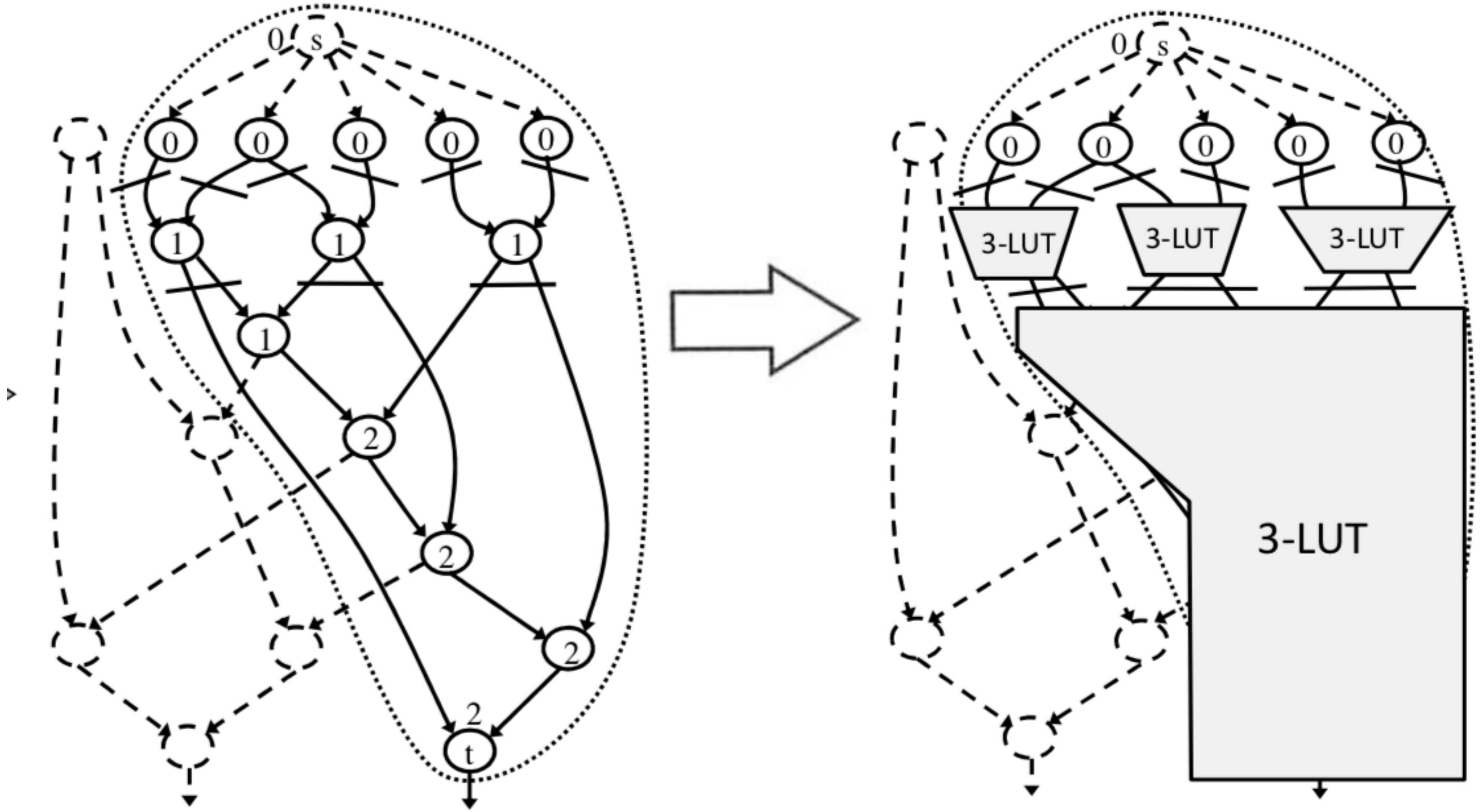


Mapping

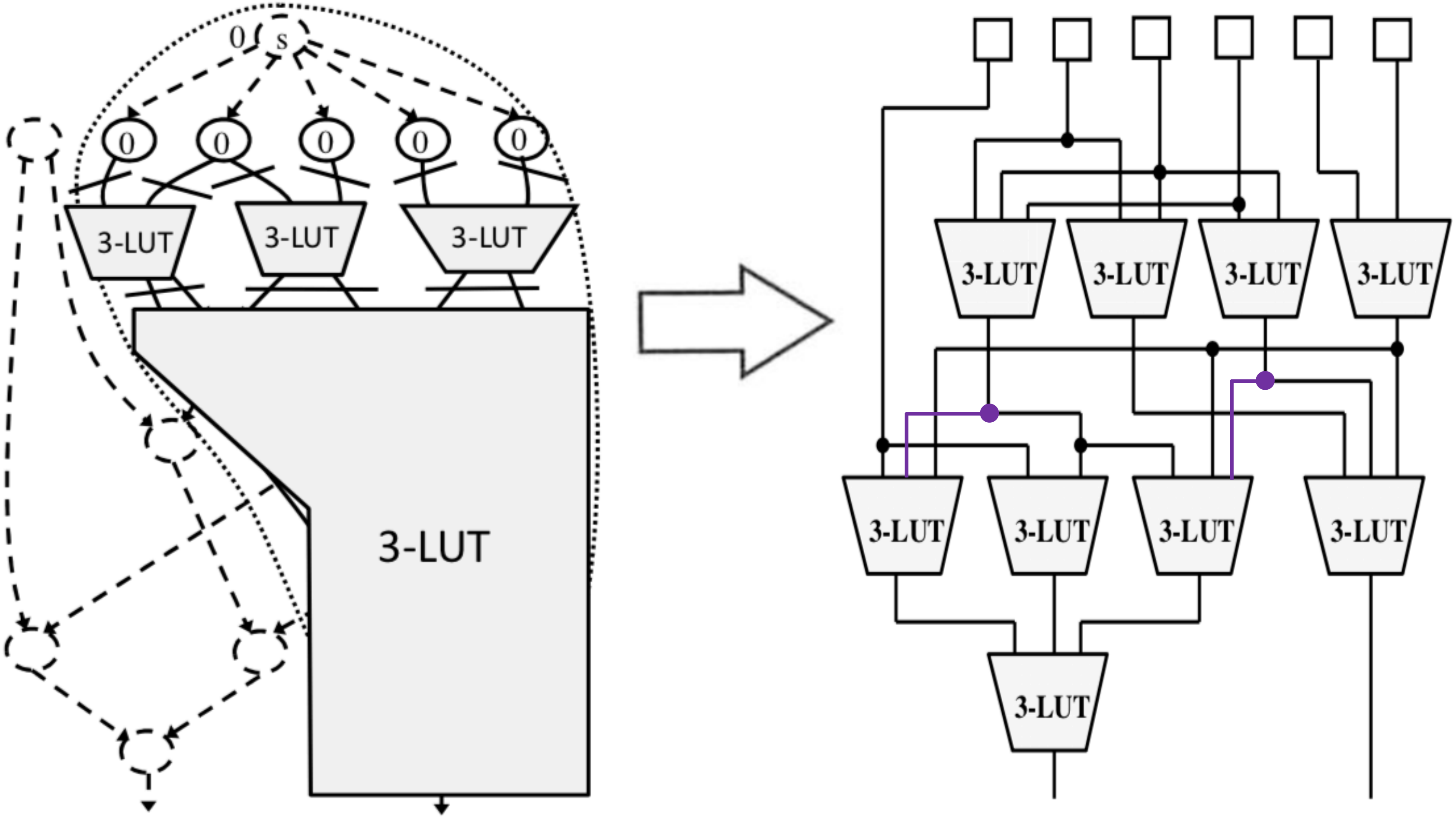


LUT netlist

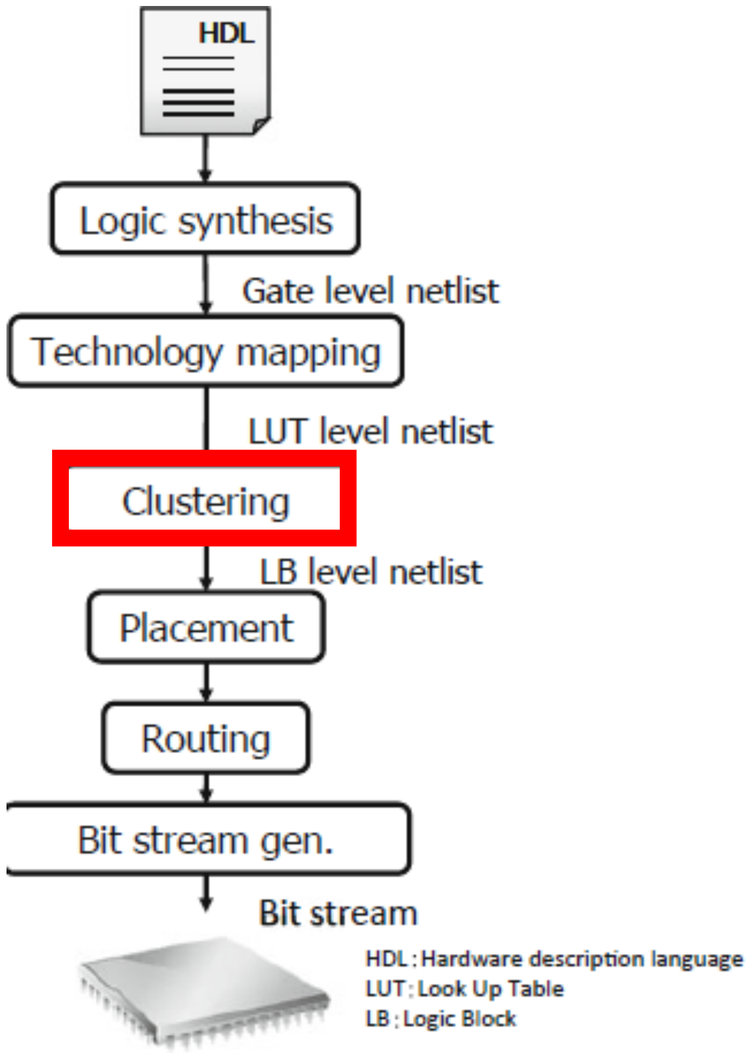
Technology Mapping



Merge k-LUT Netlists



Clustering



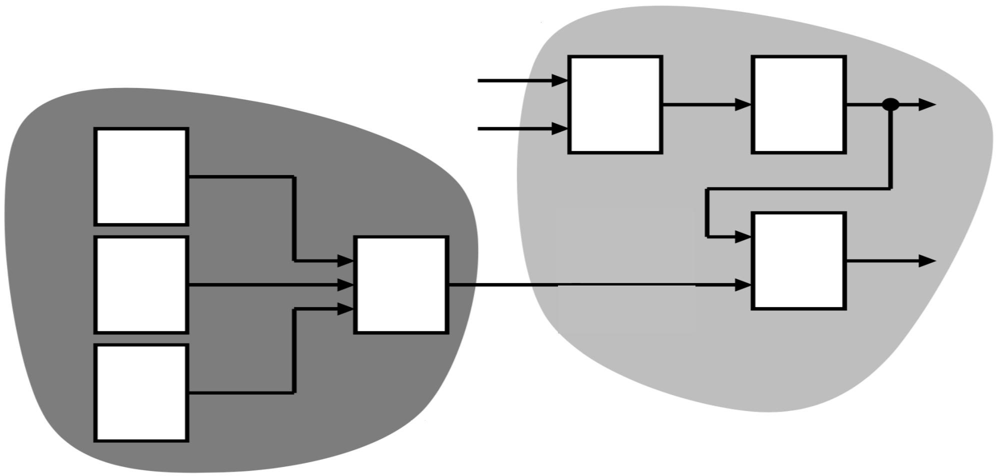
1. VPack
2. T-Vpack
   - Connection importance
   - Total route number impact
3. RPack/t-RPack/iRAC

Clustering

- Goal: Merge several LUTs into a cluster
- Considerations:

1. Routing outside the cluster has a larger delay penalty than in the cluster

2. If there is an empty in the cluster, many logic blocks must be consumed



VPack

- Goal:

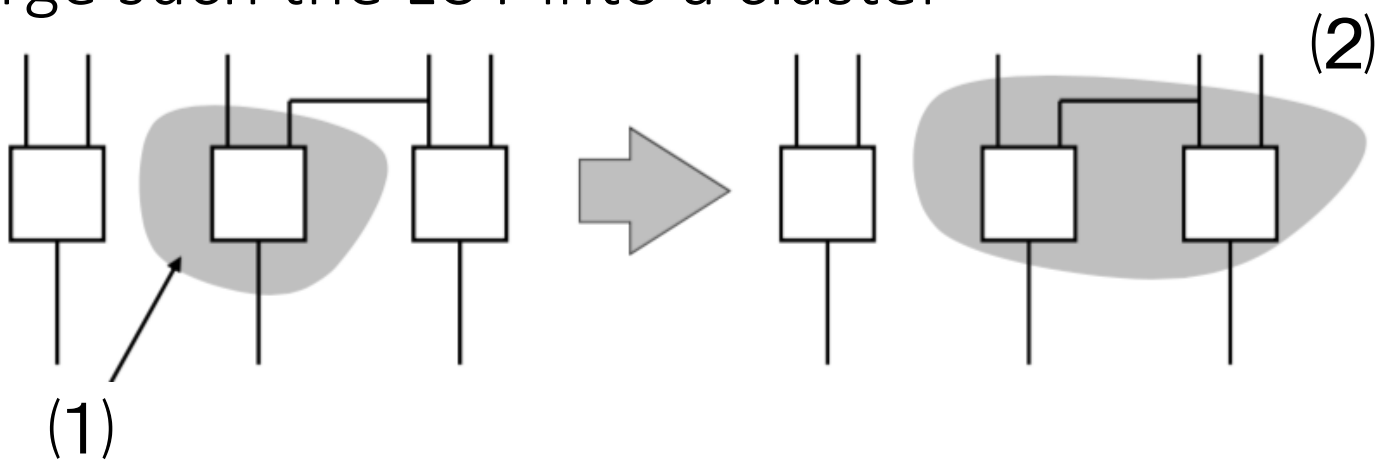
1. Minimize the number of connections between clusters

2. Minimize the number of clusters

- Strategy:

1. Select the LUT with the largest number of inputs

2. Merge such the LUT into a cluster



T-VPack

- VPack:
  - Effective for reducing the number of clusters and the number of connections between clusters

    → Not consider the delay for inside and outside the cluster

- T-VPack:

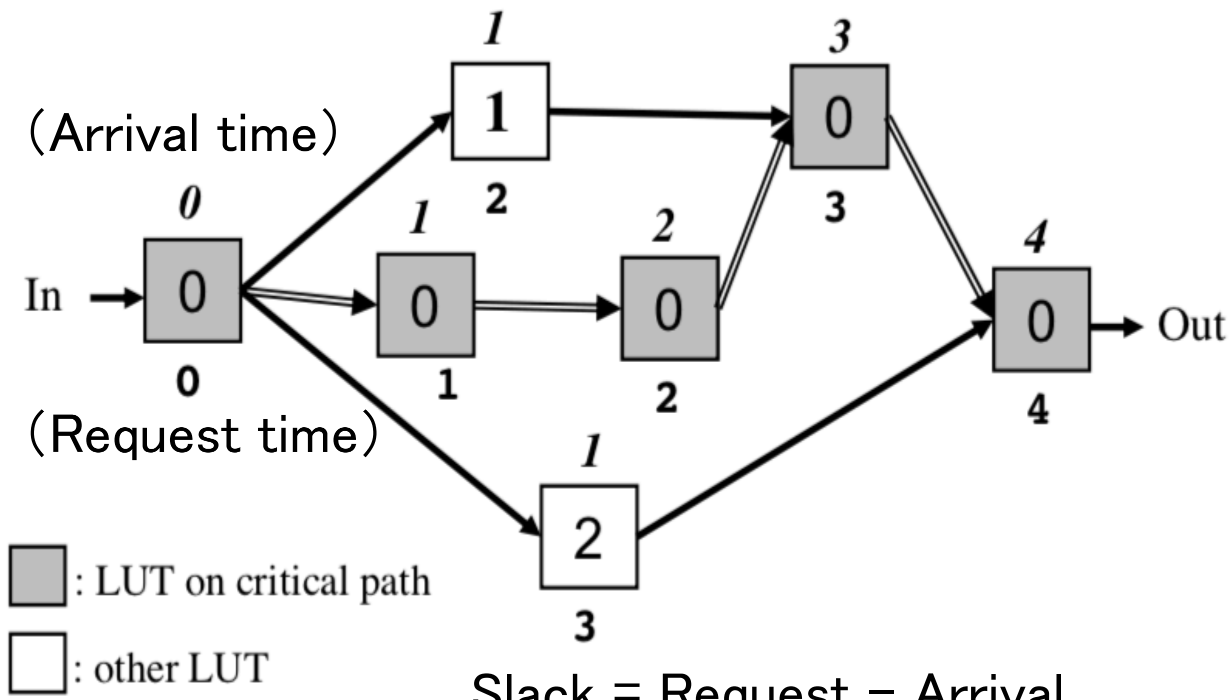
  1. Connection importance

  2. Total route number impact

  →Reduce the delay by placing the critical path in the cluster

Connection Importance

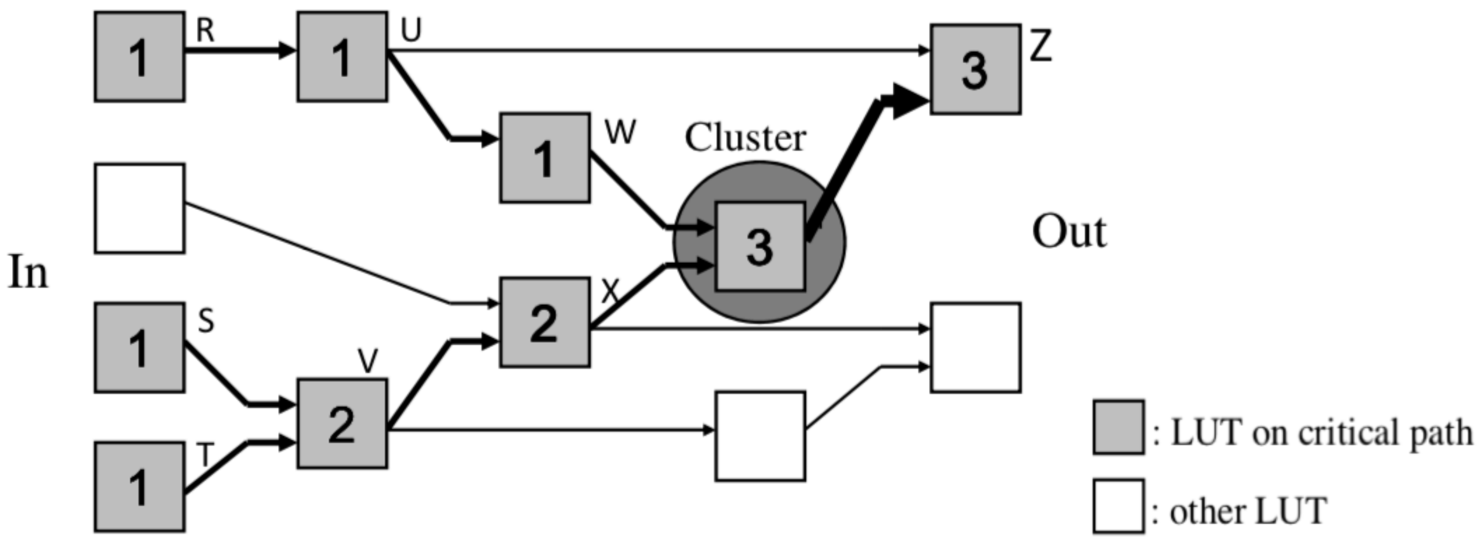
- LUTs close to the critical path (LUT with small Slack (delay margin)) are placed into the same cluster



Slack = Request − Arrival

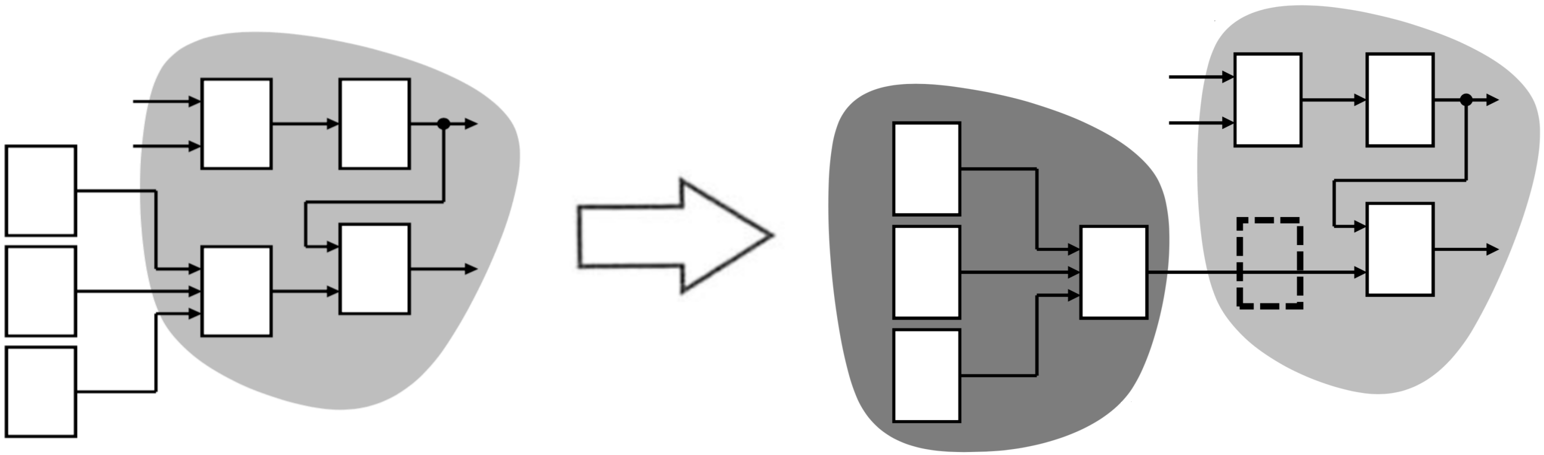
Total Route Number Impact

- Place LUTs affected by many critical paths in the same cluster



RPack/t-RPack/iRAC

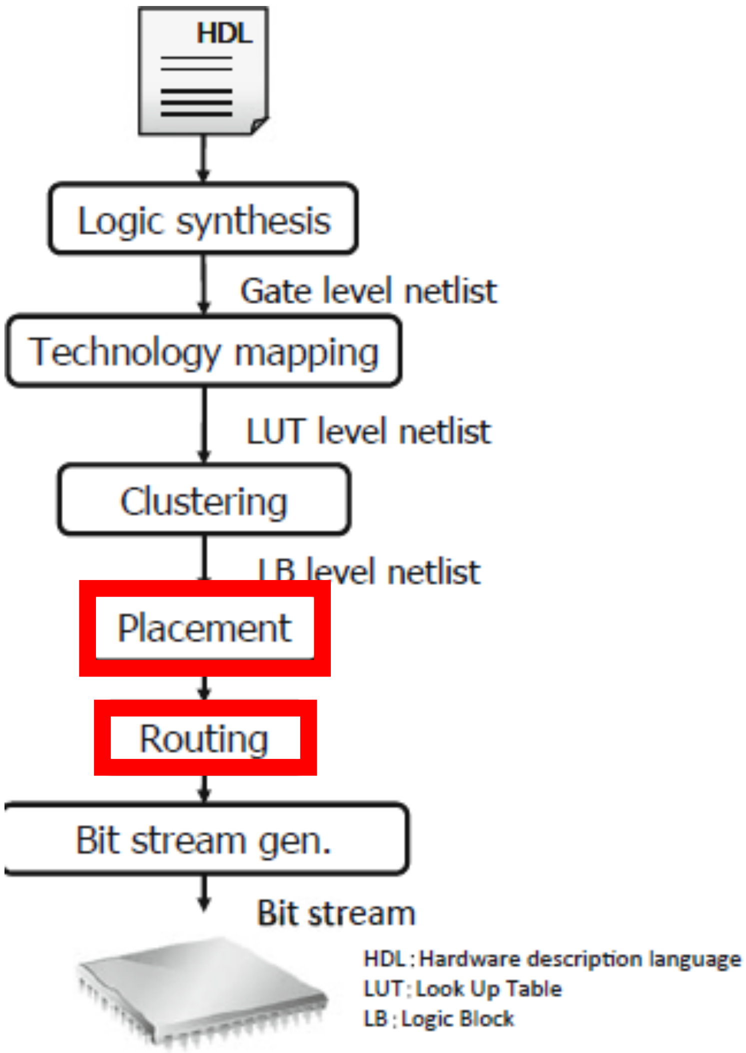
- Consider routing characteristics (degree of freedom in routing, simplicity connection)



No empty in the cluster,
however three external wires

Although an empty exist,
only a external wire

Place-and-Routing



VPR (Versatile Place and Route)

1. Placement
2. Routing
   - Detail routing

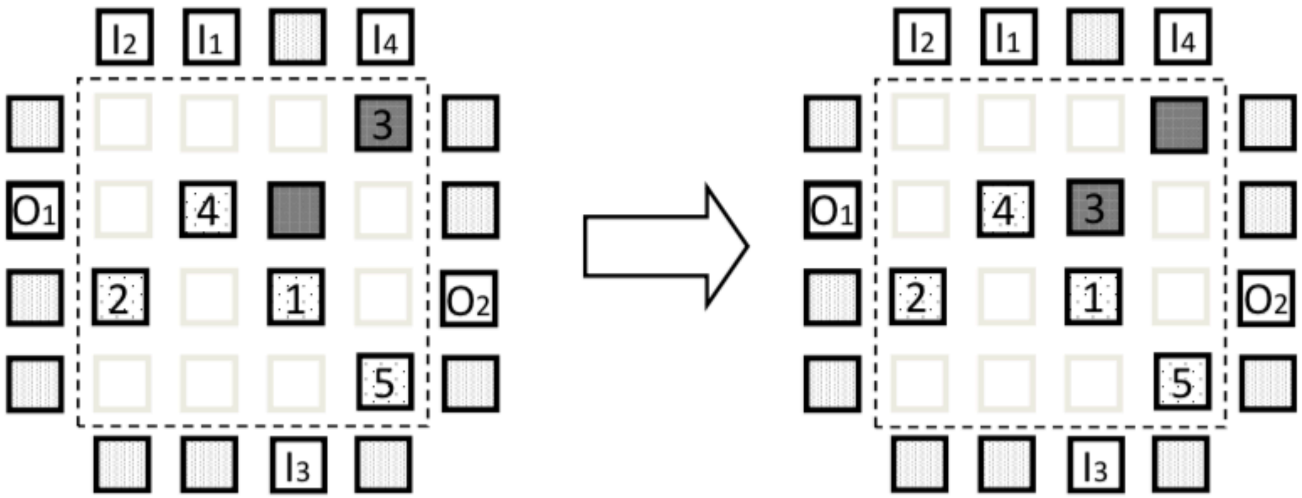
Placement

Goal: Determine the position of each block

Strategy:

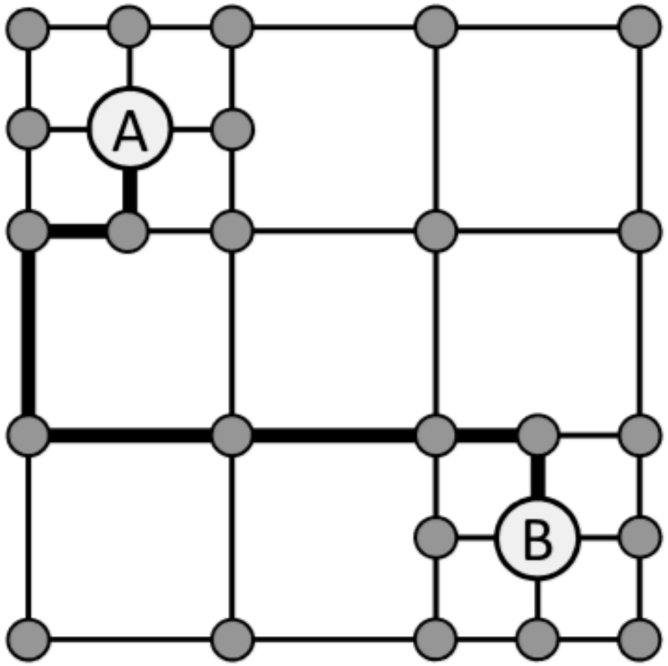
1. Place logical blocks and I/O blocks randomly

2. Exchange two blocks at random and accept cost improvement with a certain probability

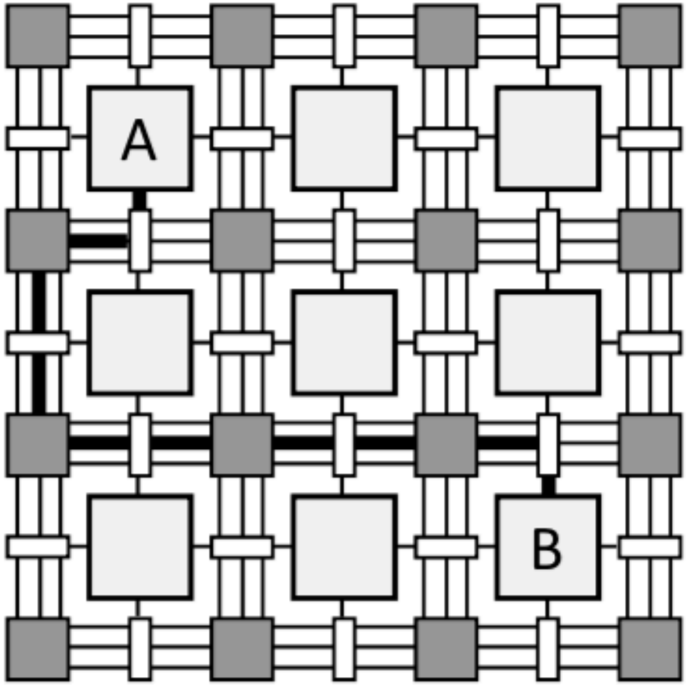


Routing

- Determining the path of the signal connection for each block



Global routing



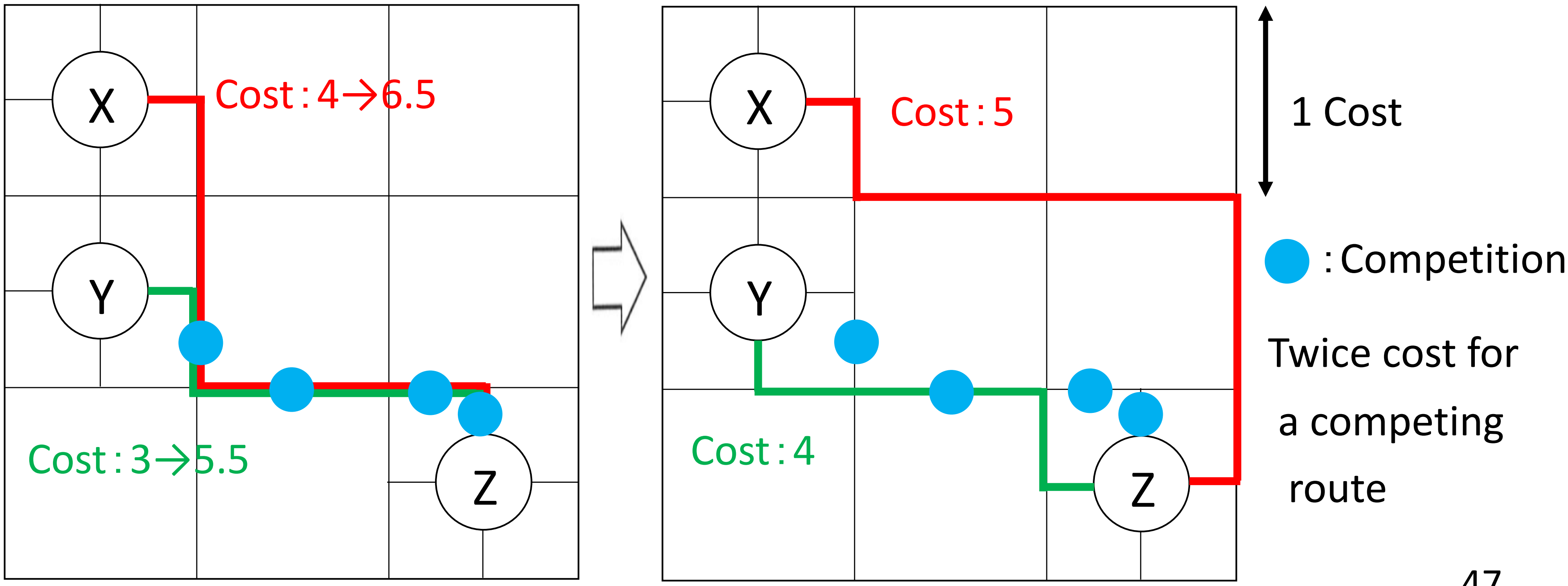
Detail routing

Detail Routing

1. Routing at minimum cost for each net

2. Add cost to competing routes, re-calculate minimum cost, then perform routing

Example: Routing to input the output of X, Y to Z



Low-Power Design Tools

1. Low-power design
2. Emap for technology mapping
3. P-T-VPack for clustering
4. P-VPR for place-and-routing
5. ACE for a measurement of activity

Low-Power Design

- **Dynamic Power Consumption**

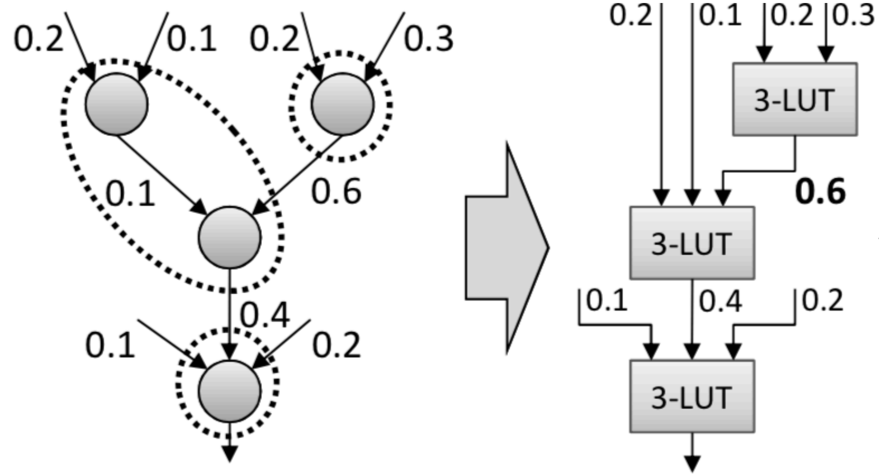
$$Power_{dynamic} = 0.5 \cdot V^2 \cdot f_{clk} \cdot \sum_{i \in nodes} Activity(i) \cdot C_i$$

- **Power Reduction**

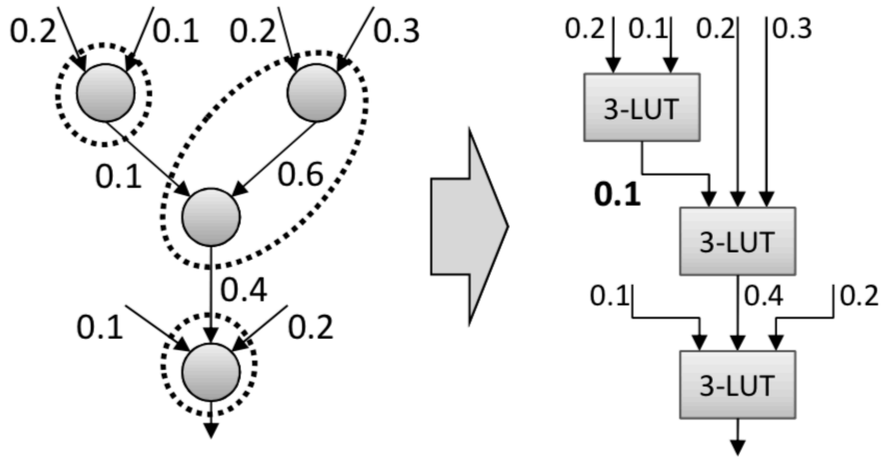
1. Low voltage for power source (V)
2. Low clock frequency (f_{clk})
3. Low switching activity ($Activity(i)$)
4. Low capacitance (C_i)

Emap: Mapping Tool

- Embed routing with the largest activity in the LUT



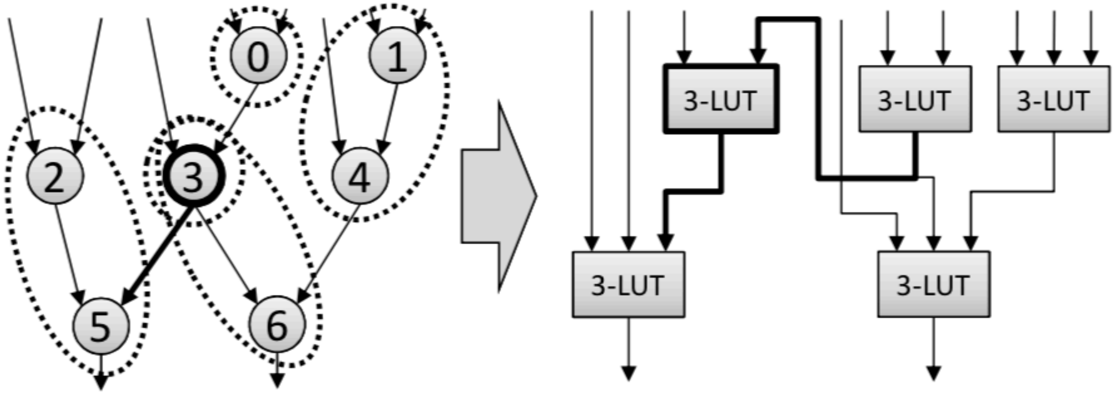
Without activity consideration



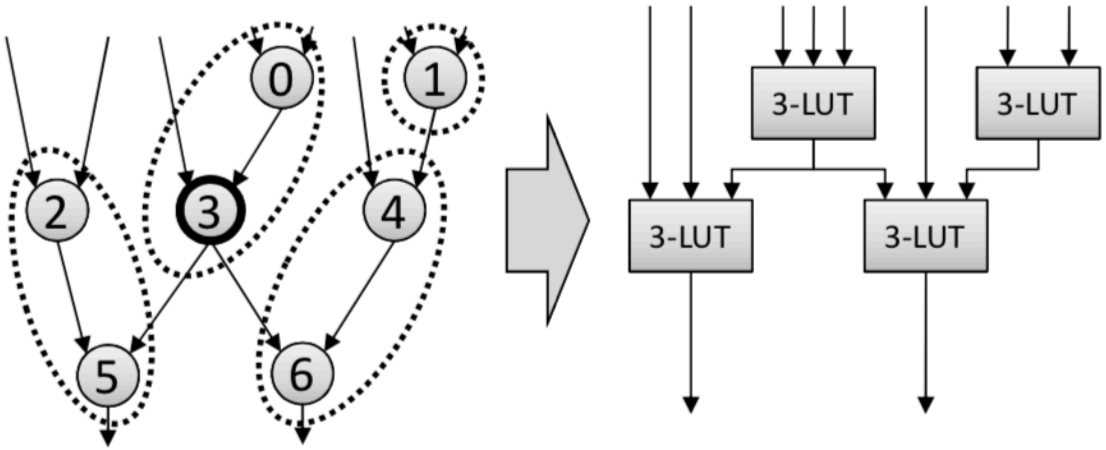
With activity consideration

Cont'd

- Consider fan-out, reduce the number of branches of wiring by reducing the number of nodes to be duplicated



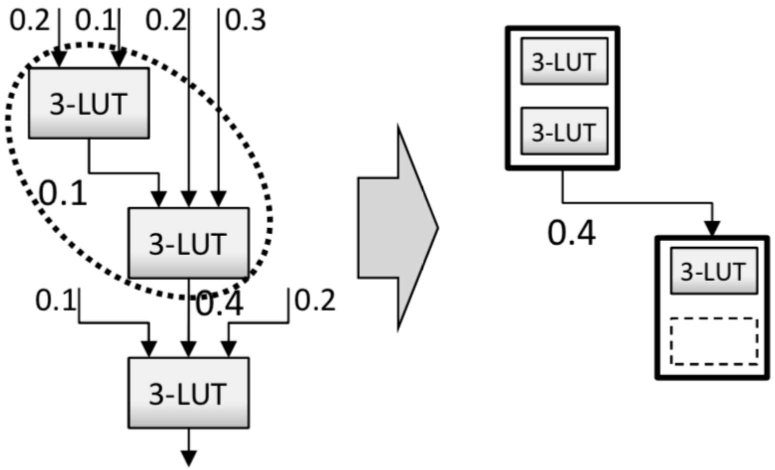
Without considering fan-out



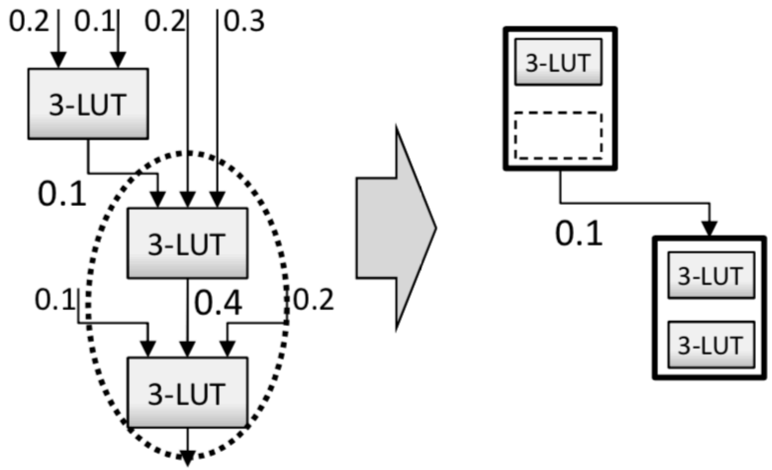
With considering fan-out

P-T-VPack: Clustering Tool

- Include routes with high activity in the cluster



Without considering activity



With considering activity

P-VPR: Place-and-Routing Tool

- Determine routes with high activity so that they are as short as possible

→ Consider making the routing with a high activity, which is not placed in the critical path

ACE: Activity Measurement Tool

1. Deterministic approach by using a simulation result

→High prediction, however long-time computation and depend on a testbench quality

2. Probabilistic approach

→Low prediction and short-time, however result is depend on an initial value

Conclusion

- In each process, aimed to optimize delay, area, power consumption

- In the future, it is expected that a method to optimize across multiple processes

Exercise

- (Mandatory) Investigate another open-source CAD tools for FPGA architecture and CAD research and report it.
- Send a report via OCW-i

Deadline is 7th, July, 2020

(At the beginning of the lecture)