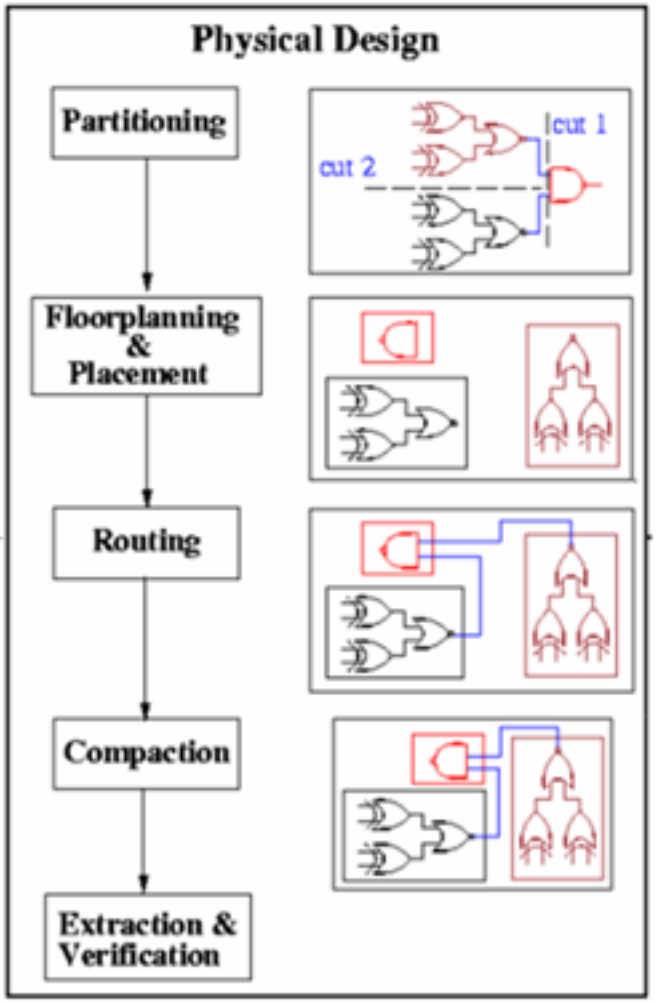
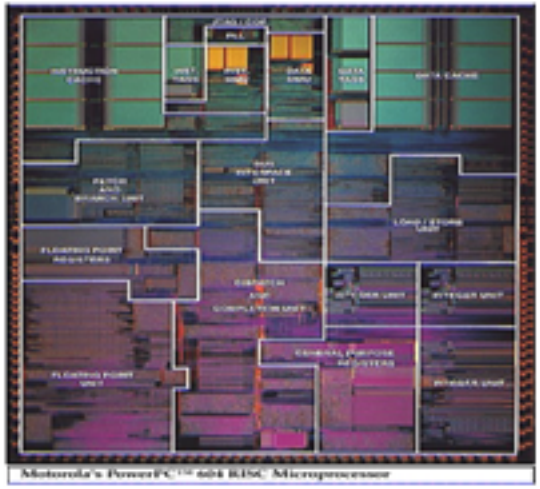
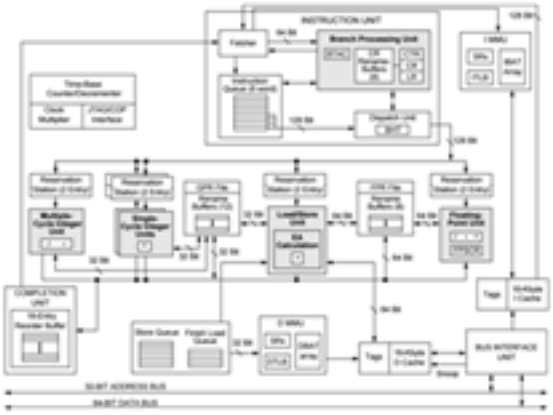
Programming Assignment 1
VLSI Floorplanner

Design Flow of Integrated Circuits (IC)

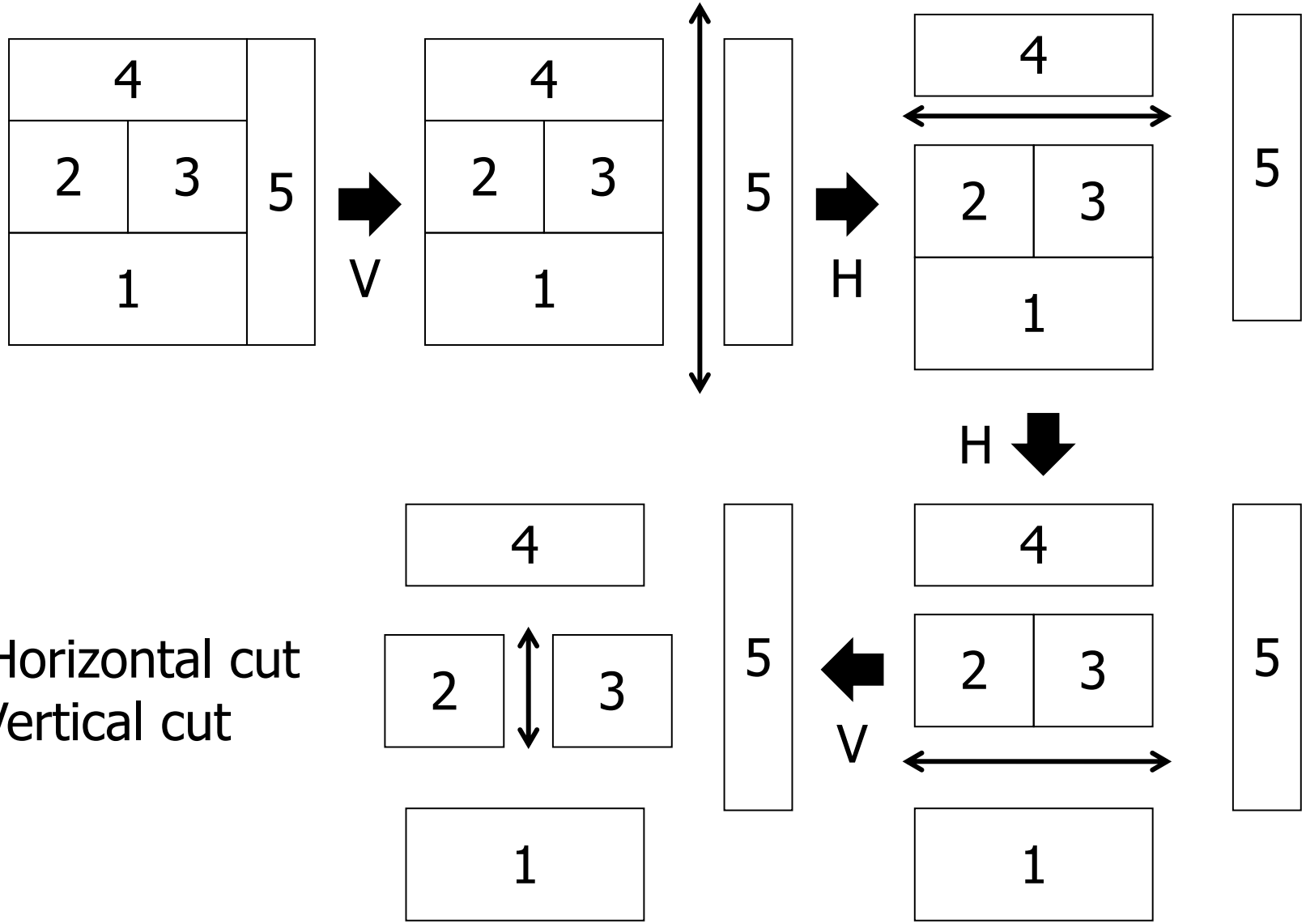


Floorplan example: PowerPC 604

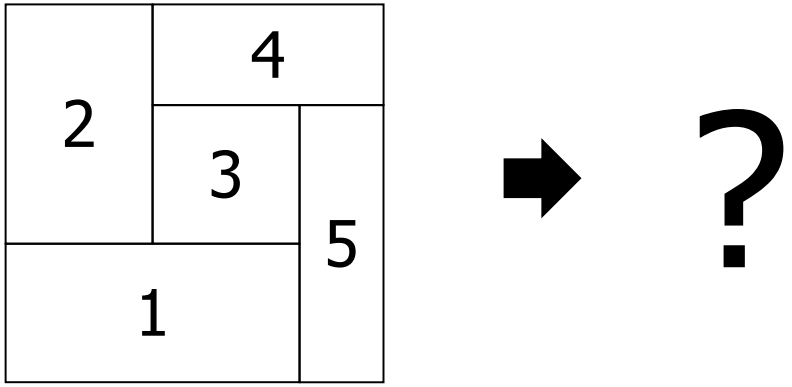




Floorplan Model – Slicing Floorplan



Floorplan Model – Non-slicing Floorplan

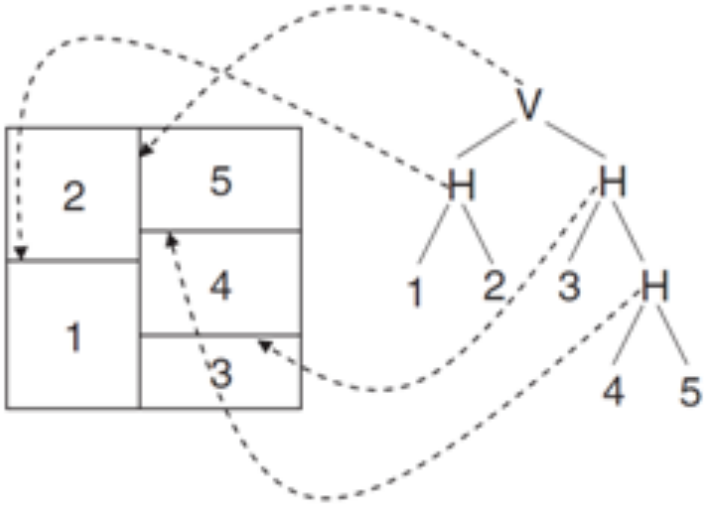


A non-slicing floorplan.

Slicing Tree Representation of Slicing Floorplan

.Properties

- A binary tree (complete)
- Modules on leave nodes & Cutlines on internal nodes
- 1D expression by postfix traversal

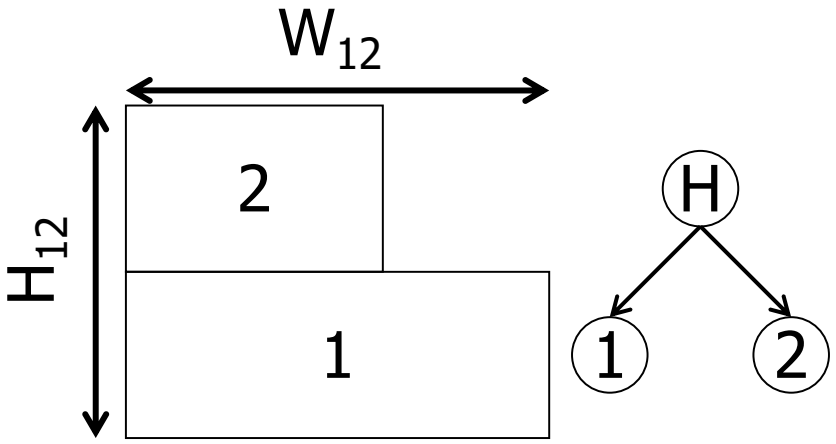


Postfix expression: *12H345HHV*

Packing from a Postfix Expression

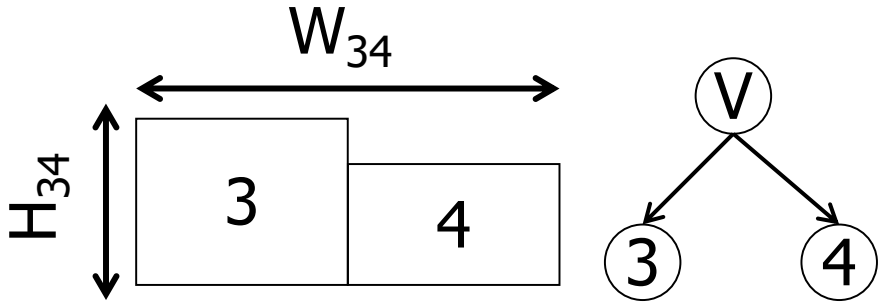
.Binary operator

- H: maximum on width and summation on height
- V: maximum on height and summation on width



$W_{12} = \max(W_1 , W_2)$
$H_{12} = H_1 + H_2$

(a) Postfix expression: 12H



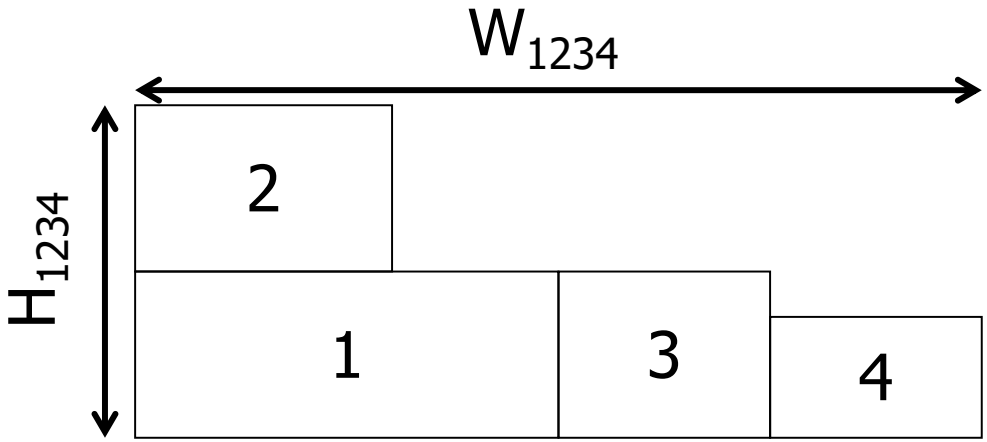
$W_{34} = W_3 + W_4$
$H_{34} = \max(H_3 , H_4)$

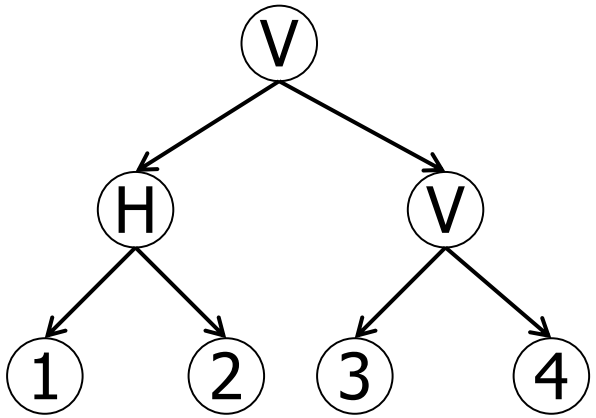
(b) Postfix expression: 34V

Packing Two Sub-floorplans Recursively (I)

.Binary operator

- H: maximum on width and summation on height
- V: maximum on height and summation on width





$W_{1234} = W_{12} + W_{34}$

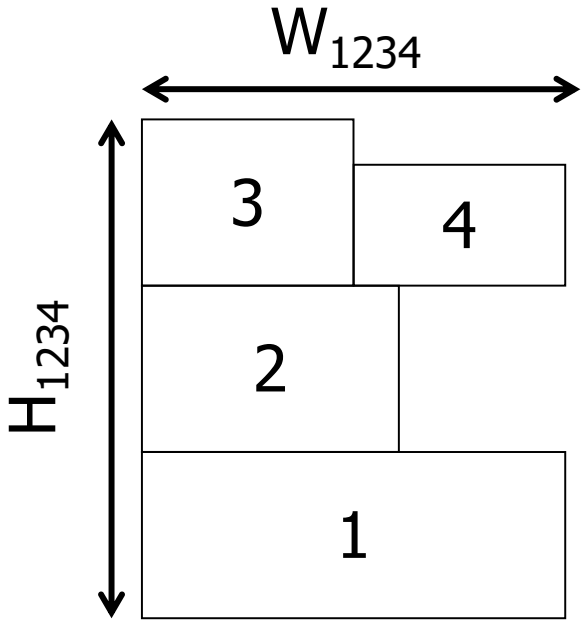
$H_{1234} = \max(H_{12}, H_{34})$

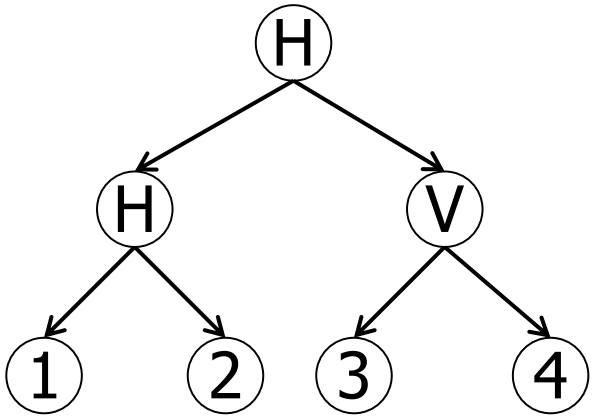
(c) Postfix expression: 12H34VV

Packing Two Sub-floorplans Recursively (II)

.Binary operator

- H: maximum on width and summation on height
- V: maximum on height and summation on width





(d) Postfix expression: 12H34VH

$W_{1234} = \max(W_{12} + W_{34})$

$H_{1234} = H_{12} + H_{34}$

Floorplan Optimization

.Area minimization is the top priority!

.Simulated Annealing (SA)

- Randomly modify the slicing tree and select the one with the minimum floorplan area

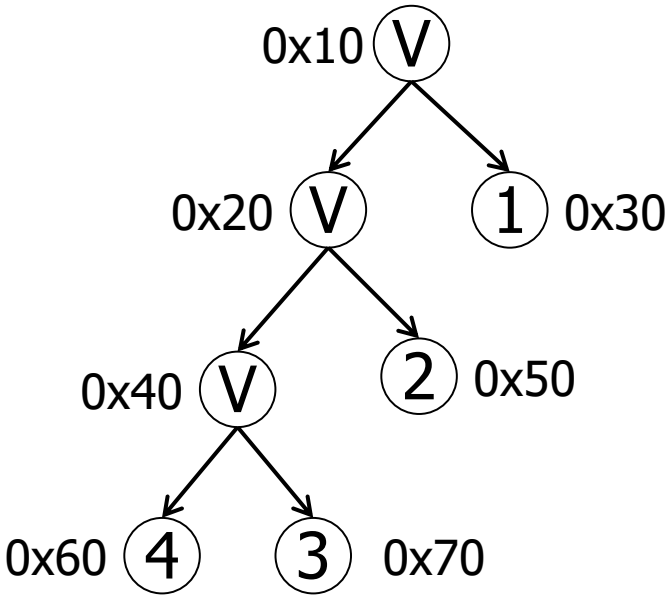
1. Generate an initial slicing tree *T*
2. Calculate the area of the slicing tree *T*
3. Generate a random neighboring solution by changing the tree
4. Calculate the cost of the new neighboring solution
5. Compare them:
   if new_area < old_area, then move to the new solution
   else accept the new solution with a user-defined probability
6. Repeat steps 3-5 above until an acceptable solution is found

- We have provided you this optimization engine

TODO Task 1: Generating an Initial Slicing Tree

.init_slicing_tree

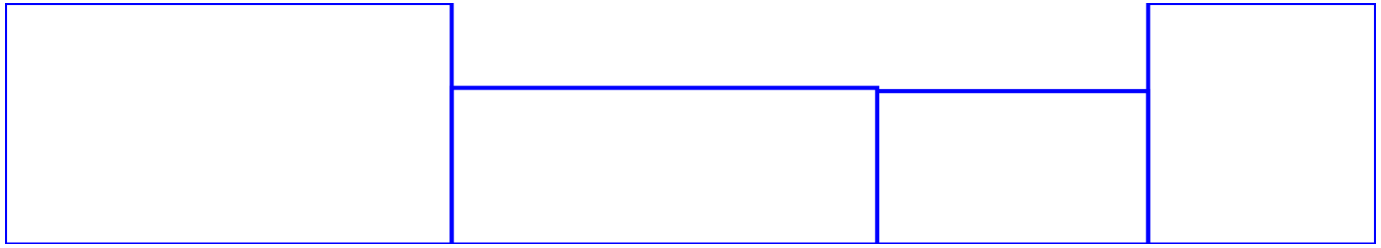
- Initialize a left-skewed slicing tree



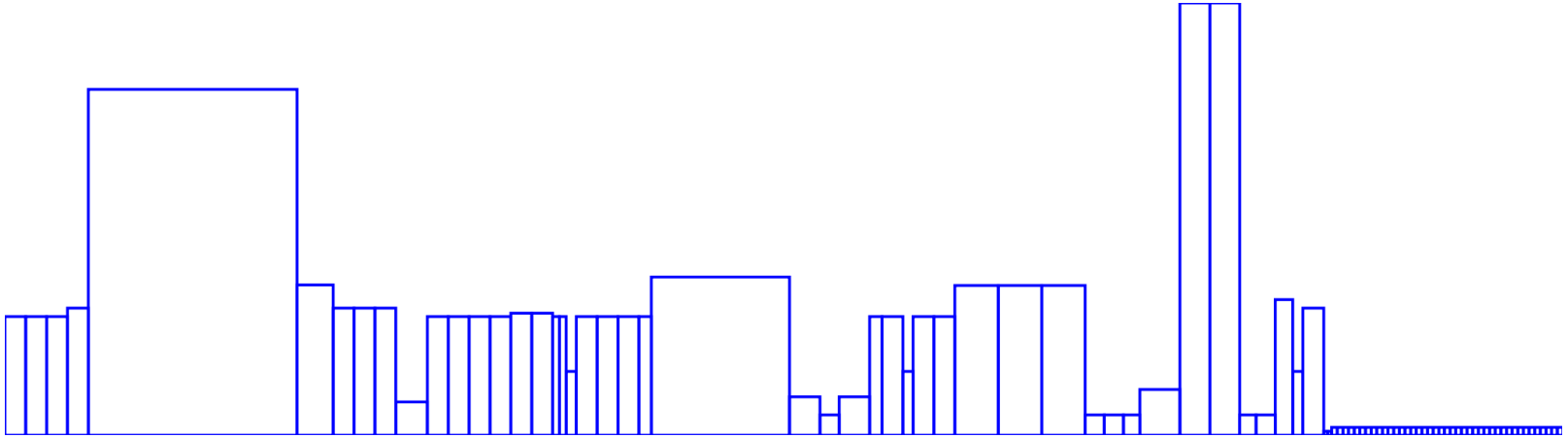

```
typedef struct NODE {
  module_t* module;
  cutline_t cutline;
  struct NODE* parent;
  struct NODE* left;
  struct NODE* right;
}node_t;
```

Node address	left	right	parent	module	cutline
0x10	0x20	0x30	NULL	NULL	V
0x20	0x40	0x50	0x10	NULL	V
0x30	NULL	NULL	0x10	1	UNDEFINED_CUTLINE
0x40	0x60	0x70	0x20	NULL	V
0x50	NULL	NULL	0x20	2	UNDEFINED_CUTLINE

Initial Floorplan



(a) circuit1.txt



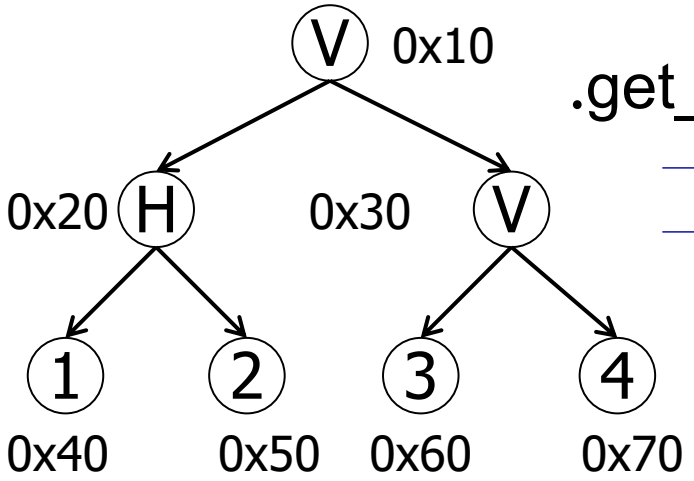
(b) circuit4.txt

TODO Task 2: Postfix Traversal Algorithm

Postfix traversal algorithm

1. Traverse the left subtree by recursively calling the Postfix function.
2. Traverse the right subtree by recursively calling the Postfix function.
3. Process the data part of root element (or current element).

nth	0	1	2	3	4	5	6
Expression unit	**1**	**2**	**H**	**3**	**4**	**V**	**V**
Node address	0x40	0x50	0x20	0x60	0x70	0x30	0x10



.get_expression

- Perform the postfix traversal
- Get the postfix expression of the slicing tree

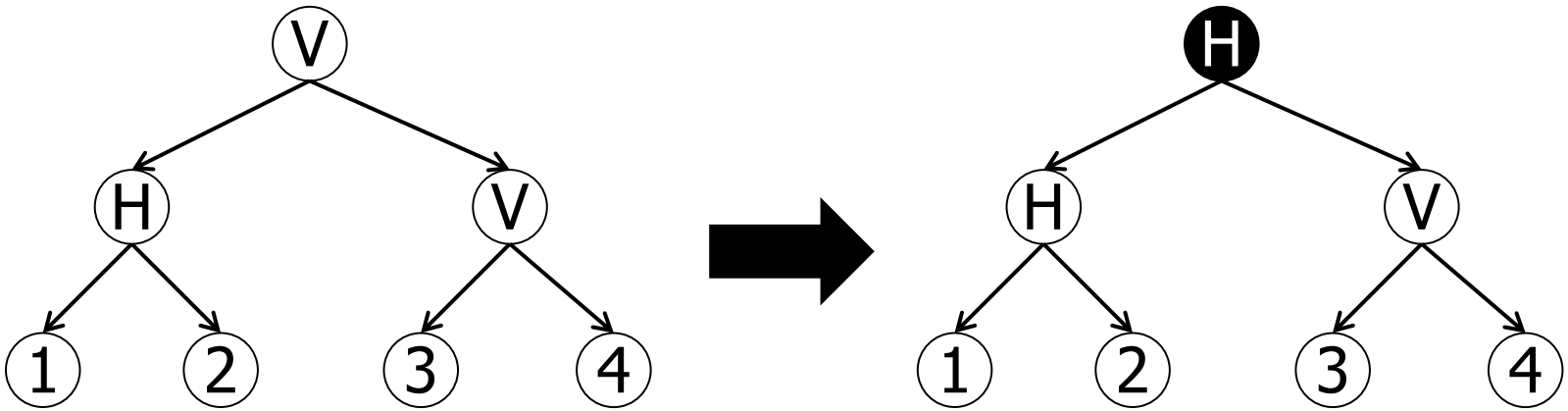
TODO Task 3: Tree Modifier – rotate and recut

.rotate

- Swap the height and the width of a module from a leave node

.recut

- Change the cutline of an internal node

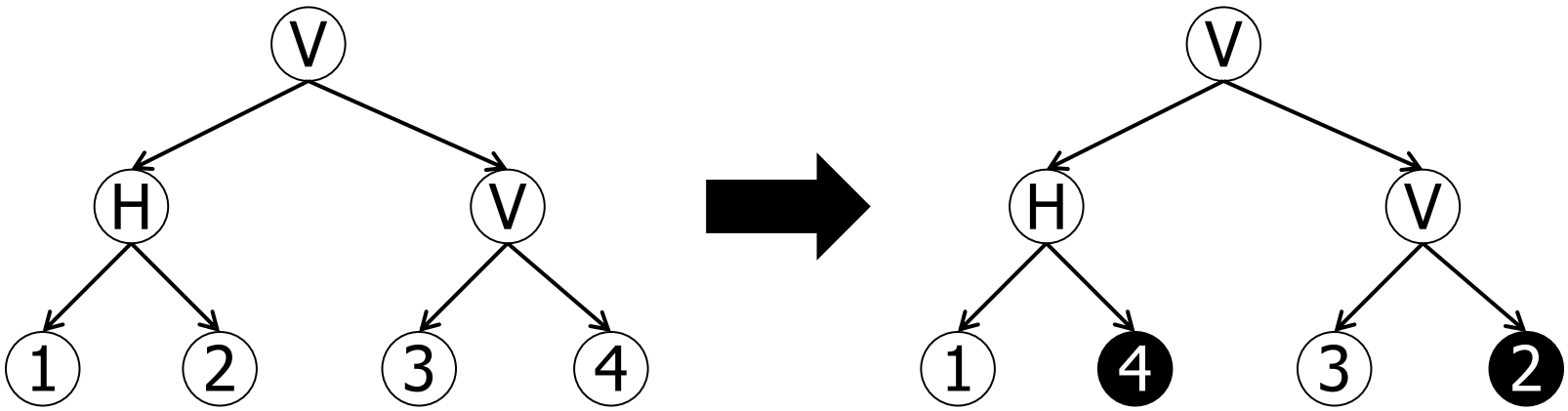


Operation: recut

TODO Task 3: Tree Modifier – swap_module

.swap_module

- Swap two modules from two leave nodes
- Simply swap the pointer value
- Do not modify the node links

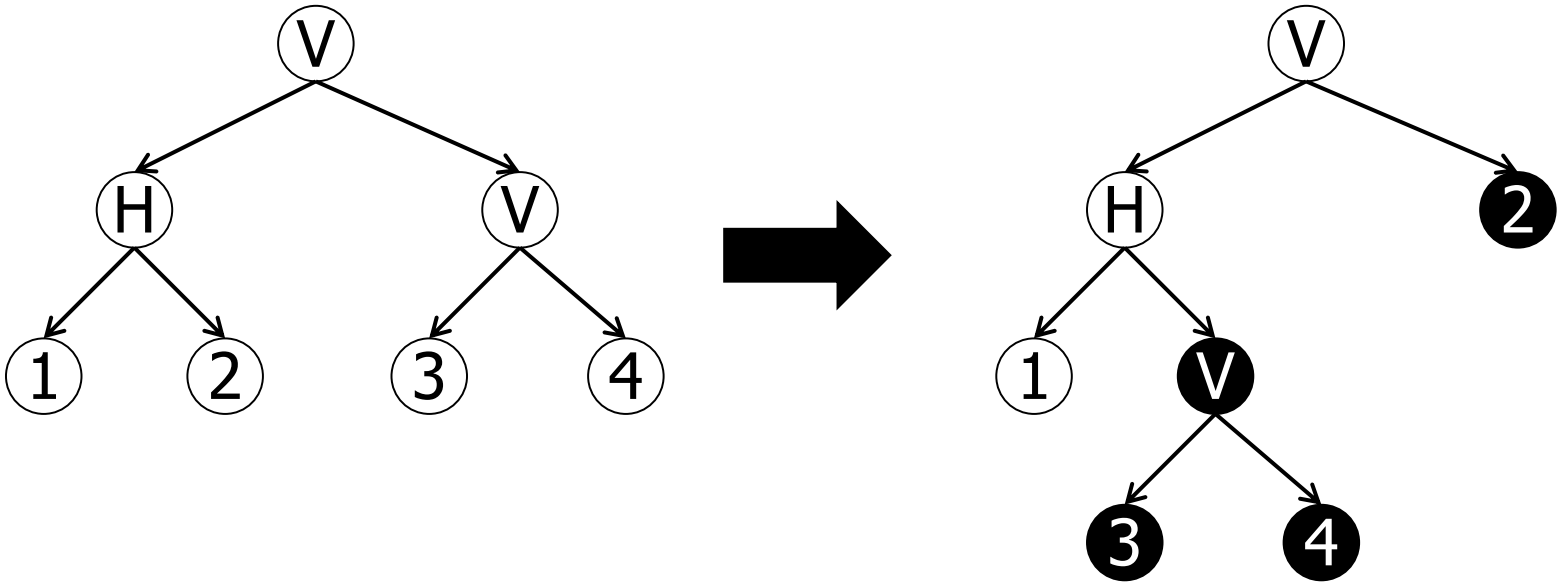


Operation: swap_module

TODO Task 4: Tree Modifier – swap_topology

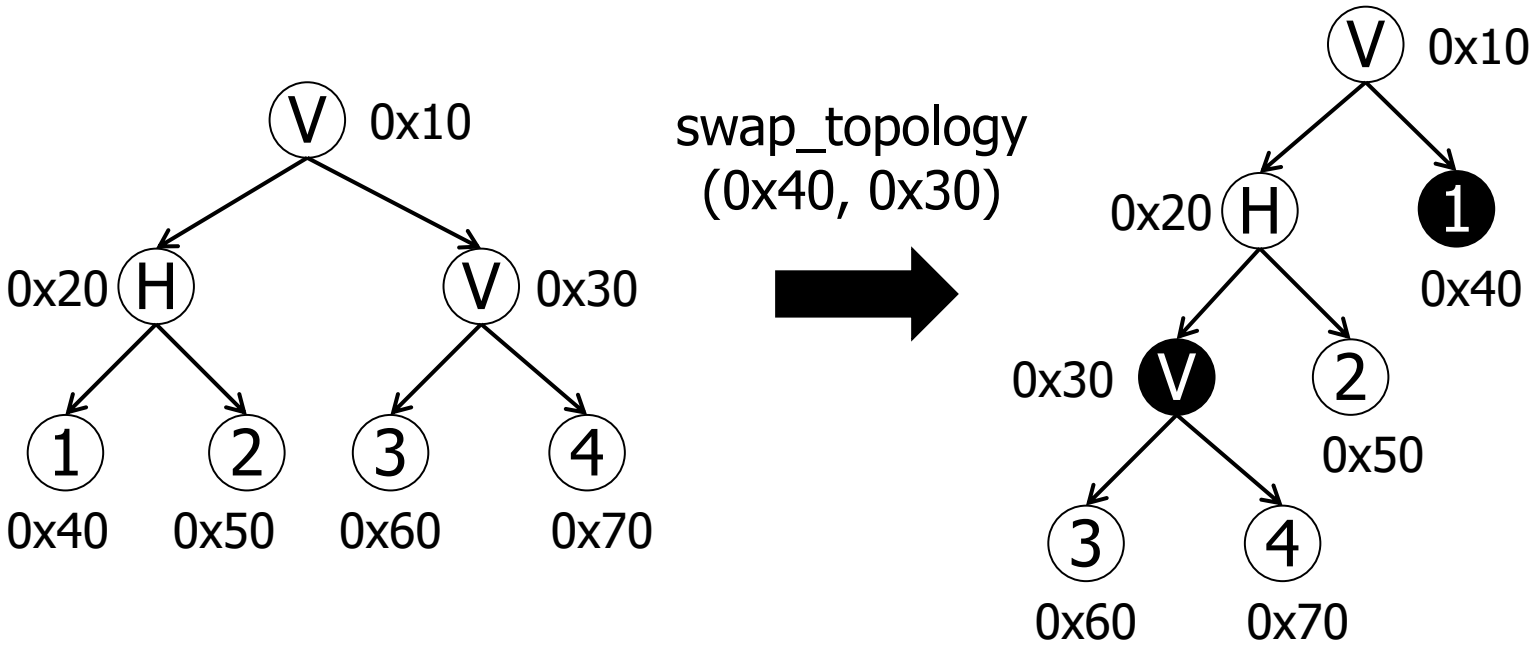
.swap_topology

- Swap two subtrees rooted at two given node pointers
- Modify the node links appropriately



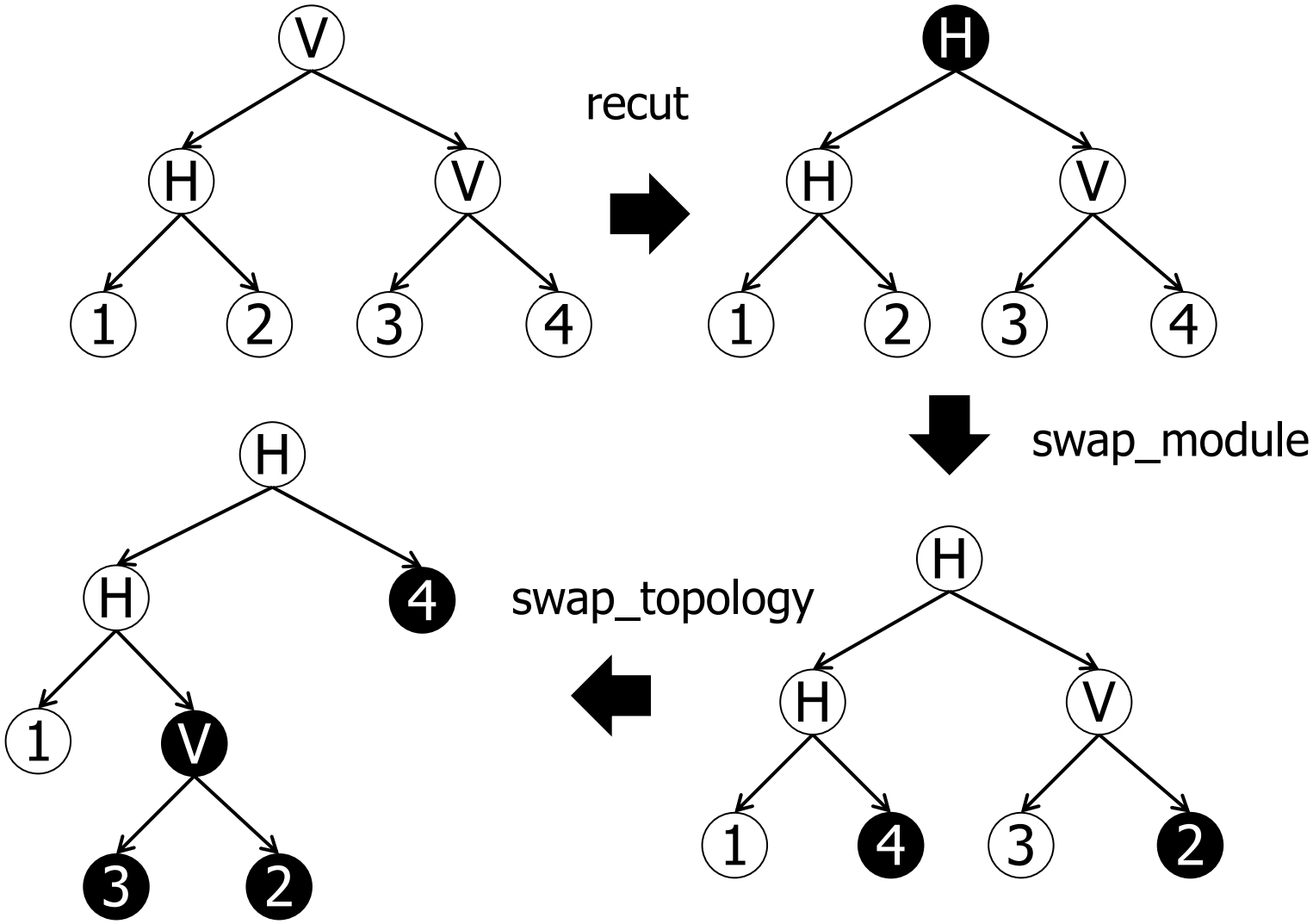
Operation: swap_topology

Example of swap_topology



Node address	left	right	parent	module	cutline
0x10	0x20	**0x40**	NULL	NULL	V
0x20	**0x30**	0x50	0x10	NULL	H
0x30	0x60	0x70	**0x20**	NULL	V
0x40	NULL	NULL	**0x10**	1	UNDEFINED_CUTLINE
0x60	NULL	NULL	0x30	3	UNDEFINED_CUTLINE

Example of a Sequence of Tree Modifiers



TODO Items

1. log in the CADE server: **lab2-20.eng.utah.edu**
   - CADE server has all required files installed already
   - You may use NoMachine or SSH
   - If you want to work on your own "Linux" machine, install cairo library following instructions at: https://www.cairographics.org/download/
2. Clone the class github
   - git clone https://github.com/tsung-wei-huang/ece5960
3. Enter the folder **ece5960/hw/hw1/**
4. Hit "**make**" to compile all sources
5. An executable "**floorplan**" will be present in the folder
6. Usage: ./floorplan circuits/circuit1.txt circuit1.png
   - Replace the number "1" with others to run different circuits
7. A default scoring output will be printed in the console
   - Maximum score is 90
   - 10 points are saved for documentation
8. Finish all TODO sections in floorplan.c; this is the only file you need to work on
9. Email me your **floorplan.c** together with your **uid** and **name** by 3:30 PM 2/27 (before class)

Demo

```
~$ git clone https://github.com/tsung-wei-huang/ece5960.git
~$ cd ece5960/hw/hw1
~$ make
~$ ./floorplan circuits/circuit1.txt circuit.png
```

```
********************************** HW1 **********************************
Initial slicing tree: Root=0xa7d0d0, num_nodes=7, num_modules=4
Initial expression: 32V1V0V
Initial area: 498760.000000
Perform optimization...
Module 0 is placed at (0, 0) with height=280 and width=296
Module 1 is placed at (523, 296) with height=188 and width=333
Module 2 is placed at (0, 296) with height=192 and width=523
Module 3 is placed at (296, 0) with height=296 and width=549
Packing area = 417728.000000 (has overlapped? 0 (1:yes, 0:no))
Draw floorplan to circuit1.png
********************************** END **********************************

****************************** VERIFICATION ******************************
Circuit: 4 golden_modules, slicing tree size = 4 leaves and 3 internals
(1) Function  'init_slicing_tree': correct!  +25
(2) Function  'is_leave'     : correct!  +5
(3) Function  'is_internal'    : correct!  +5
(4) Function  'is_in_subtree'   : correct!  +10
(5) Procedure 'rotate'      : correct!  +5
(6) Procedure 'recut'       : correct!  +5
(7) Procedure 'swap_module'    : correct!  +5
(8) Procedure 'swap_topology'   : correct!  +10
(9) Procedure 'get_expression'  : correct!  +20
Your final score for this MP   : 90
************************** END VERIFICATION **************************
```