

CVSD09 Fall Midterm Examination

Binary Tree Insertion

Please write the RTL code of a binary tree module with two functions in **90 minutes**. We have provided the testbench to test the correctness of your RTL code. To confirm your code is synthesizable, you should run the synthesis tool (Design Vision) with the provide script. Finally, remember to upload the target files before the deadline.

Function Description 1

The structure of a binary tree is illustrated in Fig. 1. There are four levels in the binary tree. The first to third levels contain 7 nodes, which have labels with number 1 to 7. Each node in level 1 to level 3 contains an 8-bit data.

It is assumed that there are no data in the fourth level. The purpose of this design is to insert a new value to the correct node of the binary tree in the fourth level.

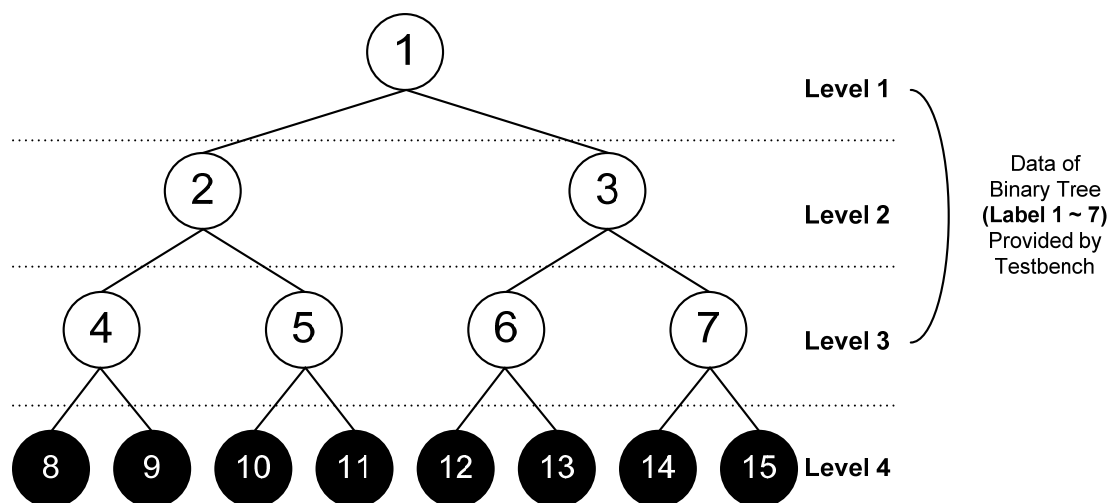


Fig. 1. Label representation of nodes in a binary tree.

An example of the binary tree is shown in Fig. 2. The 7 data of the first three levels are stored into the binary tree in the following order:

101, 37, 165, 5, 69, 133, 197.

Then, a new value is inserted to this binary tree as shown in Fig. 2. The algorithm is stated as follows:

- 1) In the first level, if the inserted data is smaller than the data in node 1, select the left child (node 2) in the second level. Otherwise, select the right child (node 3) in the second level.
- 2) In the second level, if the inserted data is smaller than the data of the current node x , select the left child (node $2x$) in the third level. Otherwise, select the right child (node $2x+1$) in the third level.
- 3) Similar to the previous step, in the third level, if the inserted data is smaller than the data of the current node x , select the left child (node $2x$) in the fourth level. Otherwise, select the right child (node $2x+1$) in the fourth level.
- 4) Output the result of the final label in the fourth level.

Following this algorithm, a new value, **105**, will obtain the final label, **12**.

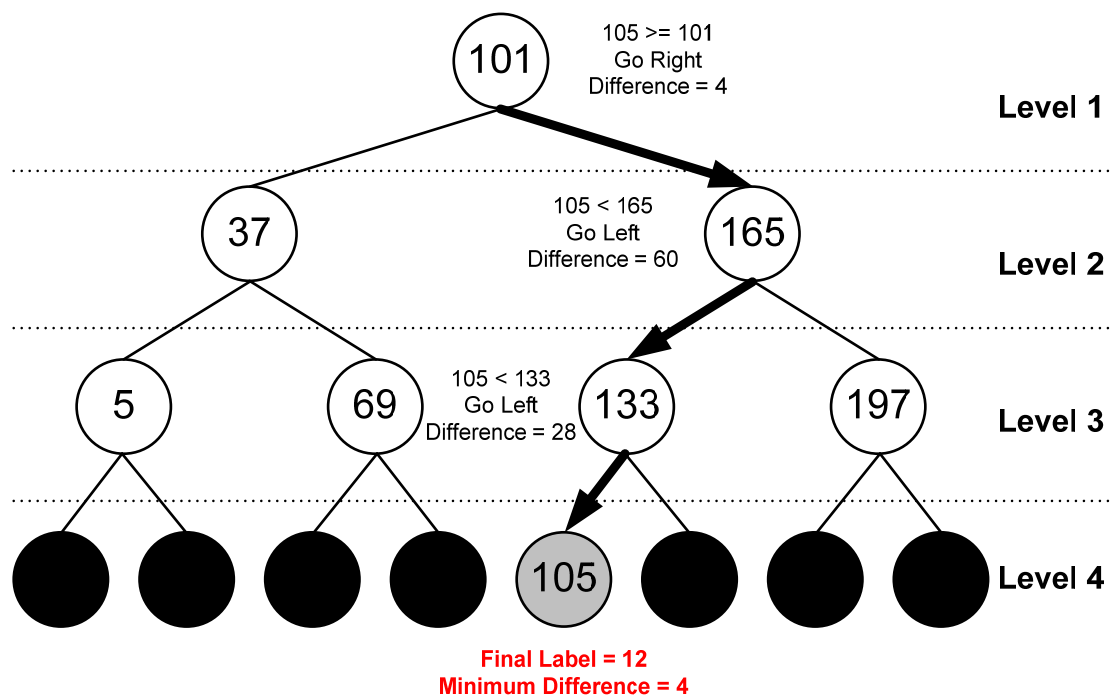


Fig. 2. Insertion algorithm for a binary tree.

Function Description 2

During the binary tree insertion process in Function Description 1, the minimum difference of the inserted data and the 7 data in level 1 to level 3 is also computed. In Fig. 2, by computing the absolute difference of the inserted data and the selected data in each level, the minimum difference can be recorded. The minimum difference is equal to **4** when the inserted data is **105**.

Binary Tree Module I/O Definition

Fig. 3 shows the I/O port connection of the binary tree module and the test bench, and Table 1 shows the I/O definition.

Signal Name	I/O	Bit Width	Description
clk_p_i	Input	1	Clock for the binary tree module. Positive edge trigger.
reset_n_i	Input	1	Reset signal. 1'b0: Reset the system. 1'b1: Working state.
valid_i	Input	1	Input data valid signal. 1'b0: Input data to binary tree module. 1'b1: No input.
data_i	Input	8	Input data to binary tree module. Positive integer. Range: 0 ~ 255
label_o	Output	4	Output label to testbench. Positive integer. Range: 8 ~ 15
min_dif_o	Output	8	Output minimum absolute difference to testbench. Positive integer. Range: 0 ~ 255
valid_o	Output	1	Output valid signal. 1'b0: No output. 1'b1: Output data to testbench.

Table 1. I/O definition.

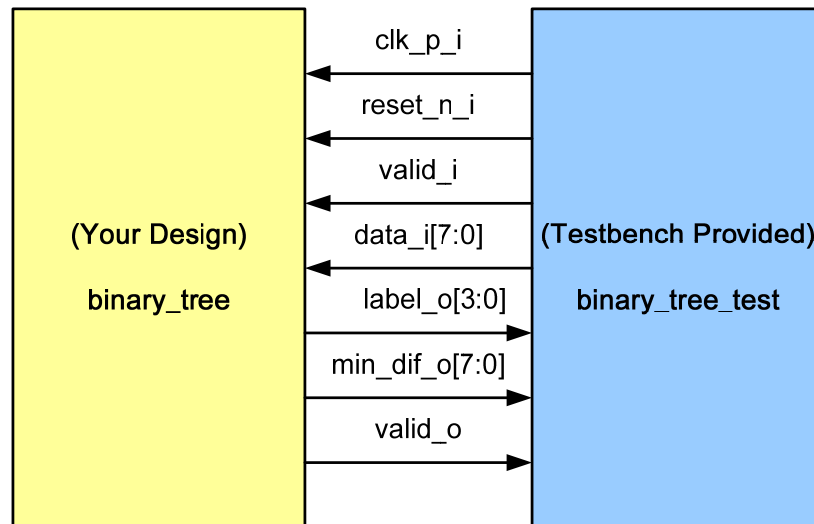


Fig. 3. I/O port connection.

The timing diagram is shown in Fig. 4. The input data, and the inserted data are sent to the binary tree module in 8 cycles sequentially. Then, after **3 cycles**, the result of label and minimum absolute difference is checked by the testbench. The output valid signal is also set to 1 in this cycle. Please confirm that your data is dumped out in the correct cycle. Otherwise, the testing will fail.

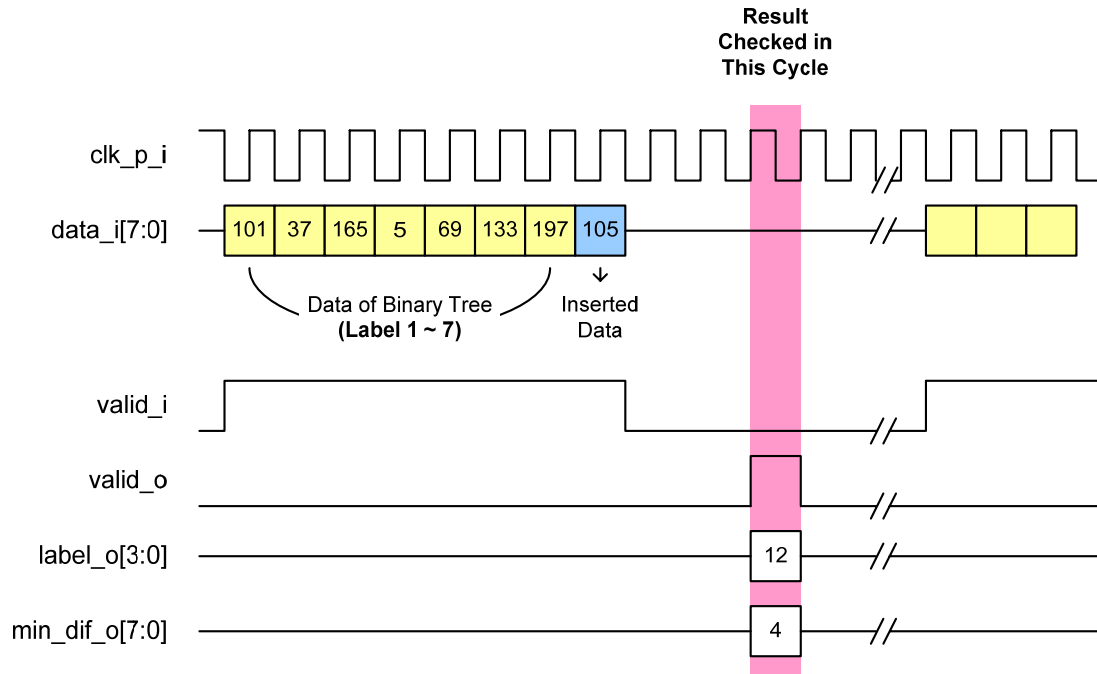


Fig. 4. Timing Diagram.

Simulation

After finishing hardware coding, you can run the provided testbench to test your circuit. We will use the ncverilog compiler to run your code. Please test your code by using the following command:

```
ncverilog +access+r binary_tree_tb.vbinary_tree.v
```

Synthesis

After finishing hardware simulation, you have to run the synthesis tool (Design Vision) with provided script.

Step 1. Open Design Vision (Remember to check the .synopsys_dc.setup file)

Step 2. Open your verilog file (Your code can't have error or warning messages)

Step 3. Run the script (type "source binary_tree.script" in the command line)

Step 4. The script will save the netlist RTL(binary_tree_syn.v), timing information (binary_tree_syn.sdf), and area/timing reports.

Netlist Simulation

Finally, run the provided testbench to test your netlist RTL. Please test your code by using the following command:

```
ncverilog +access+r binary_tree_tb.vbinary_tree.v tsmc18.v
```

Provided Files

Copy the files in Table 2 from CVSD directory. (cp -r ~cvsd/CUR/midterm .)

File Name	Description
binary_tree.v	the I/O declaration for the RTL code
clock_tb.v	testbench of biniary_tree.v
.synopsys_dc.setup	the environment settings of DV (all the paths are for cad server)
tsmc18.v	Netlist simulation file
binary_tree_tb_syn.v	testbench of your netlist RTL(binary_tree_syn.v)
binary_tree.script	the script for circuit synthesis

Table 2. Provided Files.

Upload files to the FTP server

server: media6.ee.ntu.edu.tw

port: 3999

User ID: mid09F

Password: the same as homework submission

Please open a directory with a name of your student ID:

Example: R98943000/

Then upload the following three files in Table 3 to your own directory:

binary_tree.v	RTL code of your design
binary_tree_syn.v	Netlist of your RTL code
binary_tree_syn.sdf	Timing information of your netlist RTL

***Note that the file name must be exactly the same. Otherwise you will get zero score.**

Table 3. File Name.

If you want to update your files, please use the following naming rule to open a new directory and upload the files.

Example: R98943000_v1/

Please upload your files to FTP server (the same like homework submission) before the deadline. If you are late to submit, you will get zero score on this exam.