Logic CAD of VLSI Design - 046880

Topic: Hierarchical Data Model - HCM

Introduction

Almost all CAD tools require an efficient and clear data model to represent the design being analyzed, synthesized, or verified by the tool. In this course, we provide a reference design of such a hierarchical “folded” data model that aims to clarify the concepts learned in the class and provide services like Gate Level Verilog parser which should ease the development of algorithms without the need for recoding the platform.

NOTE: that future homework assignments will rely on knowledge of this data model and algorithms like the ones presented here. So, this wet assignment is important to all your next exercises either.

In this assignment

1. You will learn about the HCM model and get familiar with the model functionality.
2. You will implement a “ranking” algorithm that requires a traversal of the hierarchical connectivity model.

Detailed Tasks

1. Setup
   Follow the Virtual Machine workshop and setup your VM.

2. Write a program that allows you to answer the following set of questions (**60%**). Read the **HW1ex1.cc** file carefully and write your answers in the designated places.
   Notice to insert the answer to the relevant variable, the program will manage the printing to test your code automatically.

   NOTE: VDD and VSS are considered global nodes – do not refer to them as common nodes! They are not considered when referring to nodes.

3. answers the following questions in file **HW1ex1.cc** :
   a. How many nodes exist in the top-level cell?
   b. How many instances exist in the top-level cell?
   c. How many instances of the cell “and4” exist in cells of the folded model?
   d. How many instances of cell “and4” exist in the entire hierarchy (means the number of “and4”s that are needed for full implementation of the top cell)?
   e. How many levels of hierarchy traverse the top cell node with the deepest reach? (Reach means the count of the number of hierarchy levels that the same node connects until we reach at the lowest level a cell that does not contain any other instances).
   f. What are the **hierarchical names** of the nodes that are deepest, i.e. the hierarchical names of the nodes that are in the lowest cells levels.
      Order the node names lexicographically.
      assign your answer for section f to the given list in the code.
      (Remember VDD and VSS are globally connected)

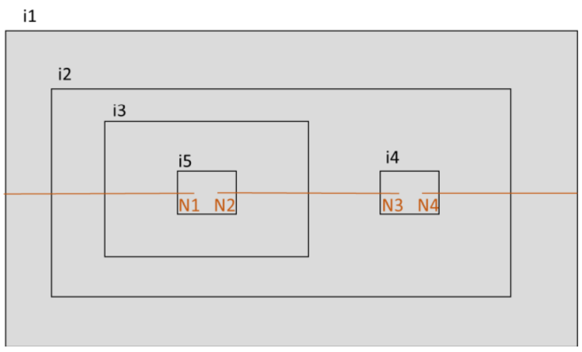


Figure 1 Example – the hirarchical names of the deepest nodes are: i1/i2/i3/i5/N1 and i1/i2/i3/i5/N2

4. Implement "max rank" algorithm (**40%**).
   Read the **HW1ex2.cc** file carefully and write your answers in the designated places.

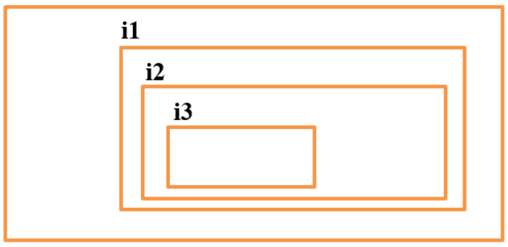
   Notice to insert the answer to the relevant variable, the program will manage the printing to test your code automatically.

   A maximal ranking algorithm provides each occurrence instance its ***maximal distance*** from any of the **input** ports of the blocks.

   NOTE: "VDD" and "VSS" are considered global nodes.

   The program starts at the inputs of the given top cell and provides one line of output for every occurrence instance found.
   The output should be inserted into the given vector in the code, such that each element in the vector is of type $\boldsymbol{pair < int, string >}$ - **the rank followed by the occurrence instance name**.
   The names of occurrence instances are a concatenation of all their parent instance name from top to bottom.

   E.g. **i1/i2/i3** is the name of an occurrence instance describe in the following drawing:



   NOTE: The order of the element in the output vector will be according to their rank and their lexicographic name - first according to the rank, and for the same rank – lexicographically. example in the next page.

The run time (and complexity) should be linear in relation to the number of instance ports.

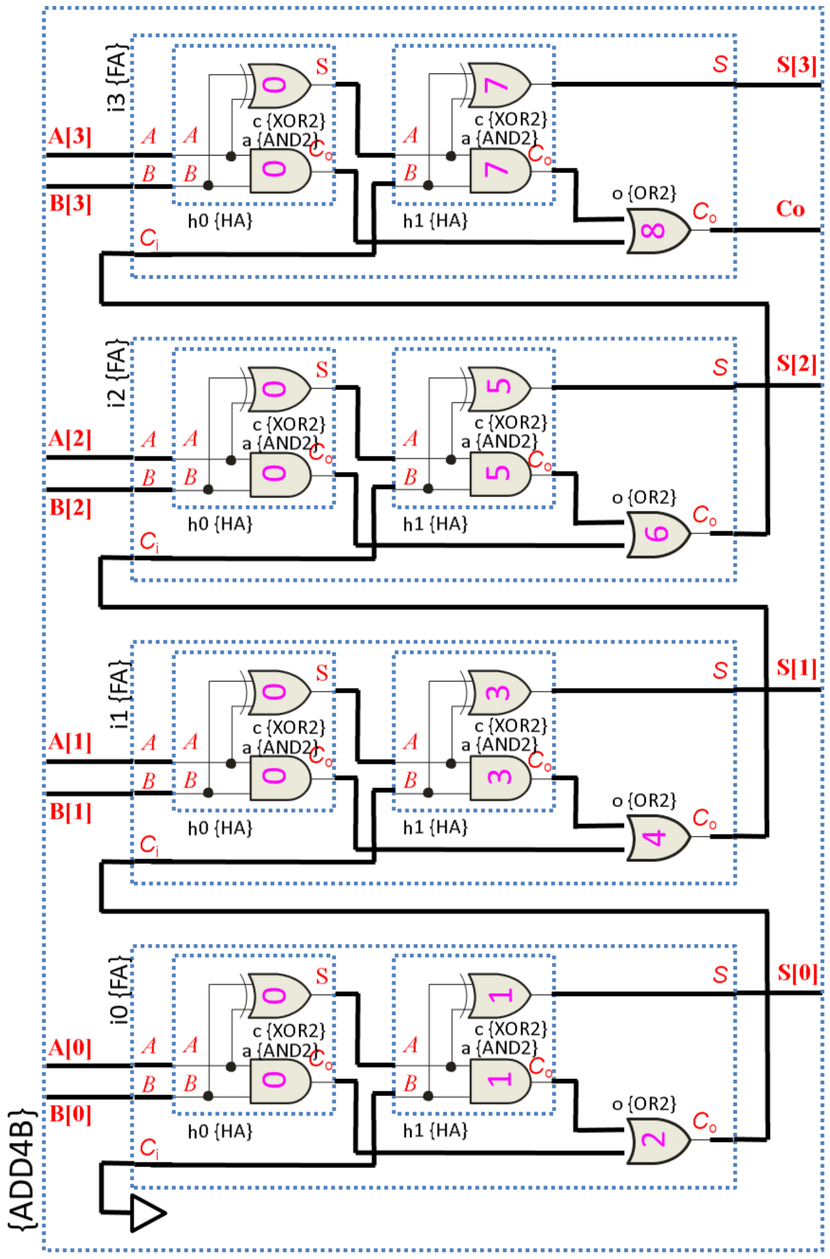
All your code must include sufficient documentation.

Implementation hint: you may want to start with the example of the flattener code provided in the flattener directory that we saw in the second tutorial. The circuits you need to handle are hierarchical.

Example Circuit and Ranks file

The circuit described below can be used for understanding the ranks files format.

The Verilog is also provided.



```
ADD4B.ranks:
0 i0/h0/a
0 i0/h0/c
0 i1/h0/a
0 i1/h0/c
0 i2/h0/a
0 i2/h0/c
0 i3/h0/a
0 i3/h0/c
1 i0/h1/a
1 i0/h1/c
2 i0/o
3 i1/h1/a
3 i1/h1/c
4 i1/o
5 i2/h1/a
5 i2/h1/c
6 i2/o
7 i3/h1/a
7 i3/h1/c
8 i3/o
```

```
adder_4b.v:
module or2 (A, B, Y );
  input A, B;
  output Y;
endmodule

module and2 (A, B, Y );
  input A, B;
  output Y;
endmodule

module xor2 (A, B, Y );
  input A, B;
  output Y;
endmodule

module HA (A, B, S, Co );
  input A, B;
  output S, Co;
  and2 a ( .A(A), .B(B), .Y(S) );
  xor2 c ( .A(A), .B(B), .Y(Co) );
endmodule // HA

module FA (A, B, Ci, S, Co );
  input A, B, Ci;
  output S, Co;
  wire   Sab, Cab, Csc;
  HA h0 ( .A(A),   .B(B),  .S(Sab), .Co(Cab) );
  HA h1 ( .A(Sab), .B(Ci), .S(S),   .Co(Csc) );
  or2 o ( .A(Cab), .B(Csc), .Y(Co) );
endmodule // FA

module ADDER4B (A, B, S, Co);
  input [3:0] A;
  input [3:0] B;
  output [3:0] S;
  output      Co;
  wire        C0, C1, C2;
  FA i0 ( .A(A[0]), .B(B[0]), .Ci(1'b0), .S(S[0]),
.Co(C0) );
  FA i1 ( .A(A[1]), .B(B[1]), .Ci(C0),   .S(S[1]),
.Co(C1) );
  FA i2 ( .A(A[2]), .B(B[2]), .Ci(C1),   .S(S[2]),
.Co(C2) );
  FA i3 ( .A(A[3]), .B(B[3]), .Ci(C2),   .S(S[3]),
.Co(Co) );
endmodule // ADDER4B
```

Stage A – Build and write your own programs with HCM

1. **cd HCM**
2. **make** – not needed if you followed the workshop
3. **cd wet01**
4. Write your code in HW1ex1.cc and HW1ex2.cc files.
   Notice to insert the answer to the relevant variable, the program will manage the printing to test your code automatically.
5. **make**
   NOTE: the tests for this exercise are automatic!
   If your submission will not compile on the virtual machine – it will be grated ZERO!

Stage B – Test your own programs with HCM

After finishing Stage A, make sure you run the following command from wet01 directory.

Now you are ready to generate the required output files for self-testing before submitting your work.

To generate **TopLevel####.stat** and **TopLevel####.rank** files run the following line in the Virtual machine, in "wet01" directory.

- **./gl_stat TopLevel#### stdcell.v c####high.v**
- **./gl_rank TopLevel#### stdcell.v c####high.v**

We provided you the files **TopLevel####.stat** and **TopLevel####.rank** to compare with your outputs.

Stage C – Create your submission

After finishing Stage B, make sure your wet01 directory include only the .cc files.
Run the following command from wet01/ directory -

1. **cd ..**
2. **tar czf wet01_<id1>_<id2>.tar.gz wet01**
   (<id1> and <id2> are the id numbers of the students)
3. Upload wet01_<id1>_<id2>.tar.gz to Moodle
   (Example of the file name: wet01_123456789_147852369.tar.gz)

- Make sure that only the .cc files are in the folder !

Note: Bold lines are execution commands for the Linux environment (LUX).

Topic	Hierarchical Data Model - HCM
Exercise owner	Mor Dahan - mordahan@campus.technion.ac.il
Given Code files	**HW1_ex1.cc** **HW1_ex2.cc** **Makefile**
Given Input Verilog files	**c1355high.v** **c2670high.v**
Given Extra Verilog files	stdcell.v
Given Output files for self-testing	**TopLevel1355.stat** **TopLevel1355.rank** **TopLevel2670.stat** **TopLevel2670.rank**
Files to submit	**HW1_ex1.cc** **HW1_ex2.cc**
Submission format	**wet01_ < id1 > _ < id2 >.tar.gz**
NOTE: the tests for this exercise are automatic! wrong submission format will be grated ZERO!	

Questions about this WET exercise should be posted on Moodle. You required to follow the answers of the course staff and write your code according to the answers.

Good luck!