CMU Fall 01 18-760 VLSI CAD
Project 2 – Addendum (Checker)

Instructor: Prof. Rob A. Rutenbar, TA: Amit Singhee

1. Checker Summary:

The checker performs the following tests for all **Transistor Netlists**:

- **Topological Sort:** Checks if you have correctly formed the signal-flow graph. If not, it tells you which node is incorrect in your order. Since each channel-graph has a unique output node, we need you to print out the channel-graph output nodes in the order in which these channel-graphs are sorted in your program.
- **Boolean equations at *each* channel-graph output node**: Checks the Boolean function at each of these nodes. This will also help you figure out which nodes you went wrong in, if any. See next section for details on printing BDDs using CUDD.

Note: For the **Gate-level Netlists**, we will not provide any checkers. You must check the final outputs yourself and tell us if they match the corresponding transistor-level netlists provided. Your write-up must however contain the details of how you made this comparison.

2. Printing using CUDD:

Use the command **PrintMinterm( )** in Cudd. This commands prints out the various paths to constant node 1 that satisfy the BDD, which correspond to cubes that cover the function. For example, if your variables are: a, b, c, d, e, the output for (b AND d') will appear as:

```
-1-0-   1
```

This implies that b appears in the non-complemented form and d appears in the complemented form, while other variables do not appear in this cube. The position of each variable in the cube corresponds to the index of the variable in the BDD manager, which in-turn corresponds to the order in which those variables were created in CUDD. In the above example, The index of a is 0, b is 1, c is 2, d is 3 and e is 4. In your output, you will have to tell us what the indices are for the various variables. Your program will have 2 variables each for every primary input variable in the circuit. You will therefore have to tell us the indices for both the inp.0 and inp.1 variables corresponding to each primary input. You will also print the node.0 and node.1 functions for each channel-graph output

The cover for (b OR d') has 2 cubes as shown below. Drawing the truth table will convince you that this is correct.

```
-0-0-   1
-1---   1
```

3. Ouput File Format:

TOPOSORT <node1> <node2> <node3> …in sorted order for channel-graph output nodes

VARS <number_of_primary_inputs >
<Input_node1> <inp1.0_variable_index> <inp1.1_variable_index>
<Input_node2> <inp2.0_variable_index> <inp2.1_variable_index>
…
… variables corresponding to each primary input
…

NODE <node1> <0: if node1.0>
```
-1---1---- 1
-1---0---- 1
... PrintMinTerm() output from cudd
```

NODE <node1> <1: if node1.1>
….
NODE <node2> <1 / 0>
…
… For all channel-graph output nodes (node.1 and node.0)

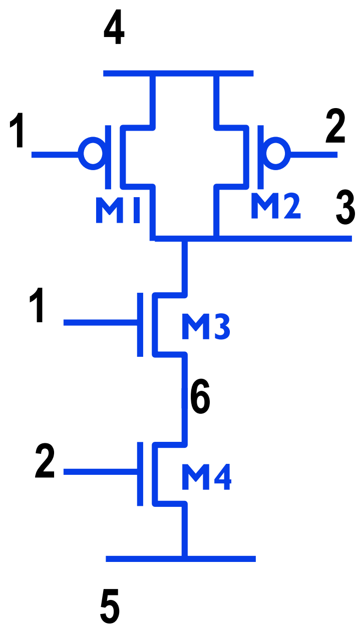
4. Example Output for c17.TRAN

```
TOPOSORT 8 9 10 11 6 7
VARS 5
1 0 1
2 2 3
3 4 5
4 6 7
5 8 9
NODE 8 0
-1---1----  1
NODE 8 1
0---1-----  1
1---------  1
NODE 9 0
-----1-1--  1
NODE 9 1
----0-1---  1
----1-----  1

.. for all channel graph output
nodes ..
```

5. Example CUDD Program for Generating the Output

Here is an example program showing output generation for a simple NAND Gate

```
main(){
      Cudd nand_mgr(0,0);

      // Create Variables for all the inputs
      BDD x_0 = nand_mgr.bddVar();
      BDD x_1 = nand_mgr.bddVar();

      BDD y_0 = nand_mgr.bddVar();
      BDD y_1 = nand_mgr.bddVar();


      // create functions for the output
      BDD out3_1 = x_0 + y_0;
      BDD out3_0 = x_1 * y_1;

      // Print out the Toposort that you generated for all the Channel-graph
      // Output Nodes Trivial in the NAND example!
      cout<<"TOPOSORT 3"<<endl;

      // Print out the Variable indices in your BDD Note you have variables
      cout<<"VARS 2"<<endl
          <<"1 0 1"<<endl
          <<"2 2 3"<<endl;

      // Print out the Cubes / Minterms for each channel Graph Output
      cout<<"NODE 3 0"<<endl;
      out3_0.PrintMinterm();

      cout<<"NODE 3 1"<<endl;
      out3_1.PrintMinterm();

}
```



nand.TRAN

```
NUMMODS 4
NUMNETS 4
NUMINPUTPADS 2
NUMOUTPUTPADS 1
VDD 4
GND 5
INPUT 1
INPUT 2
OUTPUT 3
P1 1 4 1 3
N1 1 3 1 6
P2 1 2 2 3
N2 1 6 2 5
END
```

Output

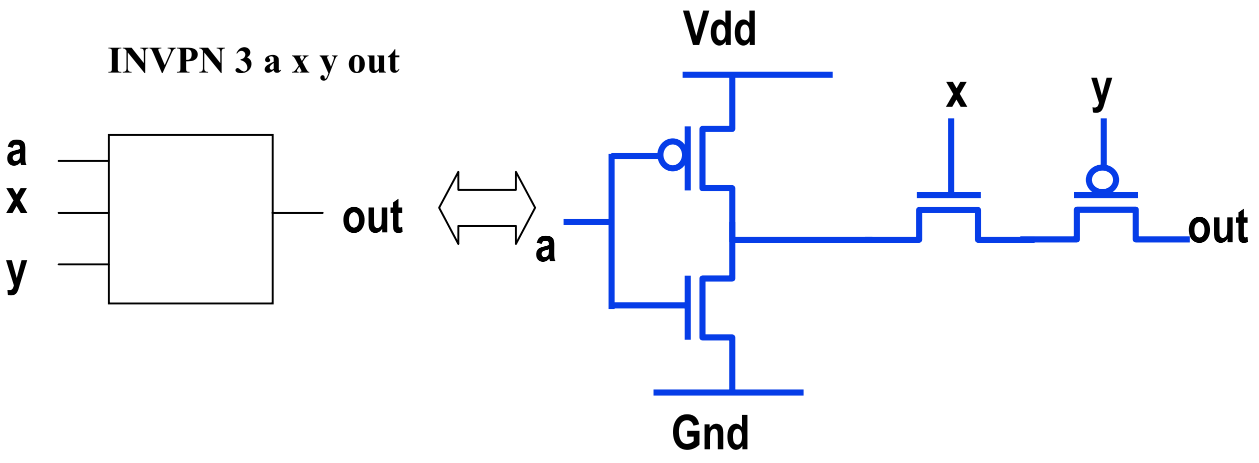
```
TOPOSORT 3
VARS 2
1 0 1
2 2 3
NODE 3 0
-1-1  1
NODE 3 1
0-1-  1
1---  1
```

6. Some new Gates in Benchmarks

INVPN (inverter with 2 Pass Transistors)

Format in the Gate-level netlist:

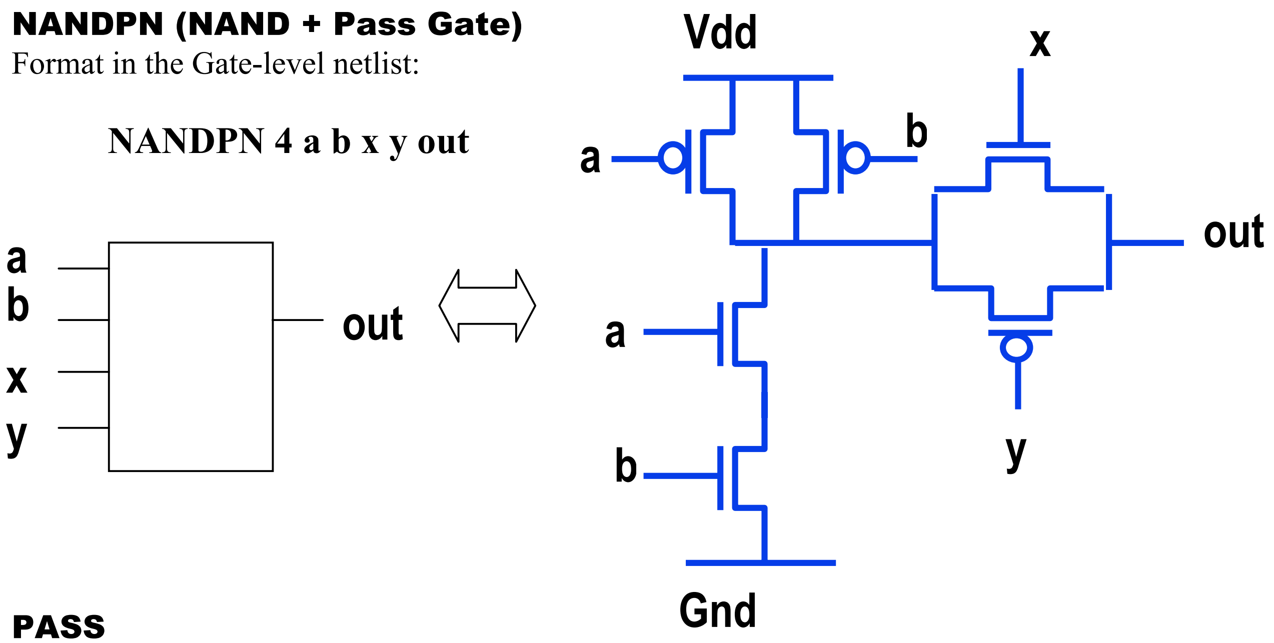
INVPN 3 a x y out



NANDPN (NAND + Pass Gate)

Format in the Gate-level netlist:

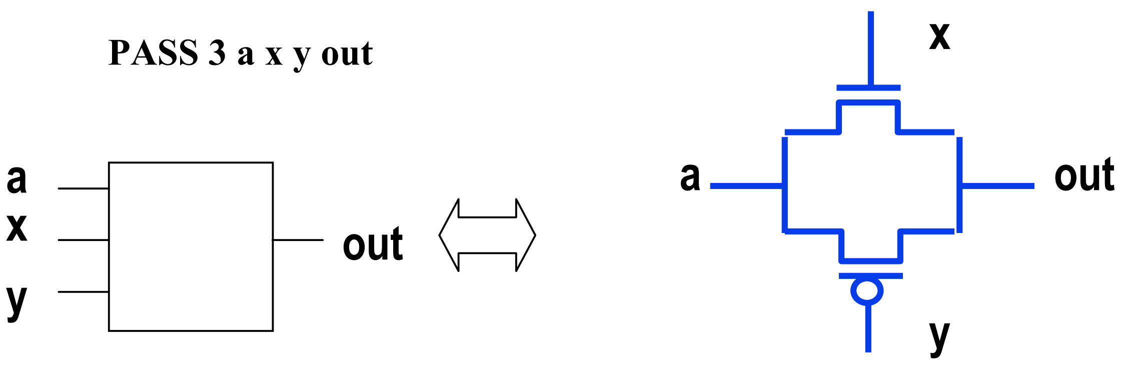
NANDPN 4 a b x y out



PASS

Format in the Gate-level netlist:

PASS 3 a x y out


a
x
y
out
x
a
out
y