Logic CAD of VLSI Design – 046880 Final Exam Term-A
Length: 3 hours
This exam contains 5 questions.
Answer only 4 questions. Each question is 25%.
If you have answered the 5 questions – please clearly state which 4 should be checked.
You can use any printed or hand written material.
You shall not use any computer or communication equipment.

Good luck.

<table>
<thead>
<tr>
<th>Question Number / שאלה</th>
<th>Grade / ציון</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
1. A general requirement of many MOS circuits is that all gates of all transistors are never left un-driven. So every gate of every transistor is driven by a node that either has:
   a. A path through MOS device channels (source to drain paths) to VDD and VSS
   b. Is a primary input port of the design
   c. Is a constant VDD or VSS

Correct and incorrect circuits are given, for example, in Figure 1.

Suggest a way for applying ALL the 4 principles of Design Automation to enable checking the design meet the above rules in efficient way for very large designs. Please be clear and describe how each of the principles is used. (4 x 7.5%)

NOTE: Do not write the algorithm of the tool

Figure 1: An example “good” and “bad” circuits (node “a” is internal to the circuit).

On the “bad” circuit input “a” of the nand2 is left floating, violating the above rule!
2. Given a ROBDD forest and a circuit in figure 2
   a. Write down the 2 functions at the top of this forest. (12.5%)
   b. Insert the logic trees of the circuit outputs y, z, w into the forest.
      Use more edges and vertex as needed. (12.5%)

Figure 2: A BDD and a Circuit
   (b) A Circuit (a) BDD
3. SAT based False Path Check
   a. Annotate on the circuit what are the logic values required on the “side inputs” of the gates on the path such that the wide path is “controlling”? (7%)
   b. Write a POS expression that can be used by SAT to check if the given wide line path in the circuit in figure 3 is not statically sensitize-able (a false path) for a rise of signal “b".
      In other words the SAT system should be satisfiable if and only if there is an assignment of the inputs that allows a rise of “b” to propagate along the wide path. (7%)
   c. Explain the algorithm that given a circuit (flat) and a requested path write a POS expression that will check if the given path is statically sensitize-able. The code should handle the check of any given path for any given rise or fall of the given input. You should write a well-documented pseudo code. (11%)

Figure 3: An example circuit with a timing path

מעגל לדוגמה עם מסלול תזמון בדיקה של התזמון עם מסלול בדיקה של מלitized את האפס הצמוד
4. Multi-Level Synthesis Kernel Extraction
   a. Given the following functions $f$, $g$ and $h$ find their Kernels using the kernels search algorithm based on weak_div algorithm learned in class. (9%)
   
   b. For each function pair find the maximal kernel intersection (if exists) (8%)
   
   c. Provide the kernels (divisors) and co-kernels. Draw the obtained gates implementing these functions. (8%)

   **Note:** the weak_div is applying simple “division” like for signed literals. But the cross of sub divisors is computed via Boolean algebra.
5. Read carefully the code in the next pages.
   For your convenience we left out space for you to fill in the comments …
   a. Describe the functionality of the functions CPD and FPI in the code (10%)
   b. Provide the errors expected to be printed (contain the text -E-) by running the program on the circuit provided as hierarchical folded model shown in Figure 4.
   Pay attention to port directions. (5%)
   c. Note that this code miss one of these errors in this circuit.
   What are these missed problems and why are they missed? (5%)
   d. How can the code be improved to find them all? (5%)

NOTE: The drawn circuit contains 4 folded cells: TOP,A,B,C (in addition to stdcells), but drawn in a flat manner.

شימו לב: המונח שבطرف בני-4 המופנים (mospins) A,C,B,TOP: (מעבב חלון פנימי-כ atas, חלון חיצוני-כ atas)
שימור בשיטור בסיסי-כ atas (flat model)

_figure 4: An example circuit for question 5.
Port directions (IN/OUT/INOUT) marked on the A, B and C cells and are IN/OUT on standard cells.

懂得 למגדלת מעבר שאול 5 (if I- 1 IN המופנים על מנת-כ atas, חלון חיצוני-כ atas (IN/OUT/INOUT)
שימור בשיטור בסיסי-כ atas (flat model)
int CPD(vector<const hcmInstPort*> &drvs)
{
    const hcmInstance *i0 = drvs[0]->getInst();
    const hcmCell *c0 = i0->masterCell();
    // ...
    for (size_t j = 1; j < drvs.size(); j++) {
        const hcmInstance *i = drvs[j]->getInst();
        if (c0 != i->masterCell()) {
            return 1;
        }
        const map<string, hcmInstPort*> &instPorts = i->getInstPorts();
        map<string, hcmInstPort*>::const_iterator pI;
        // ...
        for (pI = instPorts.begin(); pI != instPorts.end(); pI++) {
            const hcmInstPort *ip = (*pI).second;
            string ipN = i0->getName() + string("%") + ip->getPort()->getName();
            const hcmInstPort *ip0 = i0->getInstPort(ipN);
            if (!ip0) {
                cerr << ".E- 1 ... " << endl;
                return 1;
            }
            if (ip0->getNode() != ip->getNode()) {
                cerr << ".E- 2 ... : " << endl;
                return 1;
            }
            if (ip0->getPort() != ip->getPort()) {
                cerr << ".E- 3 ... : " << endl;
                return 1;
            }
        }
    }
    return 0;
}
```cpp
int FPI(hcmCell *cell, set<hcmCell*> &visited)
{
    int res = 0;
    cout << "-I- Checking cell: " << cell->getName() << endl;
    visited.insert(cell);

    map<string, hcmInstance*> insts = cell->getInstance();
    if (insts.size() == 0)
        return(0);
    // ...
    map<string, hcmInstance*>::iterator iI;
    for (iI = insts.begin(); iI != insts.end(); iI++) {
        hcmCell *ic = (*iI).second->masterCell();
        if (visited.find(ic) == visited.end()) {
            FPI(ic, visited);
        }
    }
    // ...
    map<string, hcmNode*> nI;
    for (nI = cell->getNodes().begin(); nI != cell->getNodes().end(); nI++) {
        hcmNode *node = (*nI).second;
        const map<string, hcmInstPort*> &instPorts = node->getInstPorts();
        map<string, hcmInstPort*>::const_iterator pI;
        // ...
        vector<const hcmInstPort*> dips;
        for (pI = instPorts.begin(); pI != instPorts.end(); pI++) {
            const hcmInstPort *ip = (*pI).second;
            // ...
            if (ip->getPort()->getDirection() == OUT) {
                dips.push_back(ip);
            }
        }
        if (dips.size() > 1) {
            if (CPD(dips)) {
                cerr << "-E- 4 Found XXX on cell: " << cell->getName()
                << " node: " << node->getName() << endl;
                res++;
            }
        }
    }
    return res;
}
```
```cpp
int main(int argc, char **argv) {
    int argIdx = 1;
    int anyErr = 0;
    unsigned int i;
    vector<string> vlgFiles;

    if (argc < 3) {
        anyErr++;
    } else {
        if (!strcmp(argv[argIdx], "-v")) {
            argIdx++;
            verbose = true;
        }
        for (;argIdx < argc; argIdx++) {
            vlgFiles.push_back(argv[argIdx]);
        }

        if (vlgFiles.size() < 2) {
            cerr << "-E- At least top-level and single verilog file required" << endl;
            anyErr++;
        }
    }

    if (anyErr) {
        cerr << "Usage: " << argv[0] << " [-v] top-cell file1.v [file2.v] ... \n";
        exit(1);
    }

    hcmDesign* design = new hcmDesign("design");
    string cellName = vlgFiles[0];
    for (i = 1; i < vlgFiles.size(); i++) {
        printf("-I- Parsing verilog %s ...\n", vlgFiles[i].c_str());
        if (!design->parseStructuralVerilog(vlgFiles[i].c_str())) {
            cerr << "-E- Could not parse: " << vlgFiles[i] << " aborting." << endl;
            exit(1);
        }
    }

    hcmCell *topCell = design->getCell(cellName);
    if (!topCell) {
        printf("-E- could not find cell %s\n", cellName.c_str());
        exit(1);
    }

    set<hcmCell*> visited;
    FPI(topCell, visited);
    return(0);
}
```