# Logic Synthesis Homework:
# Boolean optimization & Technology mapping

Adnan Aziz

1. **Node Simplification – I**

   Consider the Boolean network given by the following equations:

```
u1 = ! ( u2 * u4 * u5 );
u2 = ! ( u3 * u4 );
u3 = ! ( c * u4 );
u4 = ! ( a + d );
u5 = ! ( u4 * b );
```

   The single primary output of this circuit is *out*.

   For each node in the network, specify its SDC and ODC in terms of its immediate inputs, as well as the primary inputs. For the node whose output is *u2*, specify its SDC and ODC when the node is viewed as a function of *u3, u4* and *u5*.

   **20 marks**

2. **Node Simplification – II**

   (a) Describe the set of functions $f : B^4 \rightarrow B^3$ which can be "safely" substituted for the adder element in the design shown in figure 1. (By safely, we mean the overall function from inputs $a, b$ to the output remains unchanged). What is simplest circuit that can safely replace the adder block?

   (b) *Directly* write a `.blif` level description of this design, i.e., do not go via verilog/vl2mv/VIS. Describe the adder using 3 tables (since blif tables are single output, this is the minimum required). Optimize this in SIS using **only** the *full_simplify* command (which incorporates the ODC and the SDC to simplify nodes).

   Report your results, and describe why SIS could not achieve the level of optimization you were (hopefully) able to in the previous section.

   **25 marks**

3. **Node Simplification – III**

   Take the 8-bit ALU you performed algebraic optimizations on in the previous homework, and peform additional optimizations using the *simplify, full_simplify,* and *red_removal* commands; report the best area (measured using literal in factored form) you achieved using these optimizations. Also run the design with the prepackaged set of optimizations in *script.rugged*, and report results. Compare your results with pure algebraic optimization.

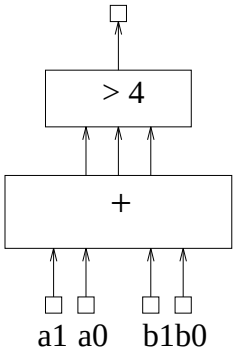


Figure 1: Cascaded designs

The *script.rugged* script is in */home/projects/logic_synthesis/sis/sis/sis_lib/*; invoke it by typing *source script.rugged* at the SIS command line.

**10 marks**

4. **Technology Mapping**

   (a) Decompose a 4-to-1 mux gate into 2-input NOR and NOT gates. Next, map this network (using the tree mapping approach described in class), using a library which has only 2-input NORs, 2-to-1 muxes, and NOT gates available. (Assume that the 2-input NOR has area 5, the 2-to-1 mux has area 2, and the NOT has area 1.)

   (b) Dynamic programming is supposed to return an optimum answer. Does this mean that the answer returned by technology mapping has minimum area? If not, where is optimality lost?

   (c) Use SIS to produce a minimum area circuit with no consideration for load limits; interpret the results. This will entail creating a technology library; see

   - `/home/projects/logic_synthesis/sis-1.2/sis-1.2/sis/sis_lib/22-1.genlib`

   for an example of what a library looks like. Also, read the man pages for *read_library* and *map*.

**35 marks**

5. **Technology Mapping – II**

   (a) Decompose a 2-input XOR gate (using the standard decomposition) into 2-input NAND and NOT gates (boolean functions). Draw all decompositions which have fewer than 10 gates in all. Next, map this network, using a library which had only NOR and NOT gates available.

   (b) Notice, that dynamic programming is an algorithm that returns a minimum (area) answer. Does this mean that the answer returned by technology mapping has minimum area? If not, where is optimality lost?

**30 marks**

**⋆–credit (Technology mapping)**

Derive a formula that yields the number of distinct decompositions of a function $f$ implementing the conjunction of $n$ variables, into two-input AND gates. Tabulate the number of distinct decompositions for $n = 2, 3, \ldots, 10$.