Lec-6-HW-1-devices

Reading: PP, Chapter 3:
§ 3.1 (transistors),
§ 3.2 (OR, NOR, AND, NAND, and DeMorgan),
§ 3.3 (DEC, MUX, FA, PLA)

Problems, PP, Chp 3:
3.1 (n and p transistors),
3.2 (cmos inverter),
3.3 (how many 2-input functions?),
3.5 (trans. ckt. => truth table),
3.6 (as 3.5, but tricky ckt.),
3.7 (fix broken trans. NAND ckt.),
3.8 (label ckt. to match func.),
3.9 (expression to truth-table),
3.10 (NOR truth-table),
3.11a (trans. ckt. for 3-AND, 3-OR),
3.11b (conduction diagram for 3.11a using input vectors),
3.16 (truth-table to PLA [connect parts of fig. 3.17])

(Problem)----------

Basic CMOS gates are NAND, NOR, NOT. Electric comes with these three corresponding basic gates:

Components.schematic.{AND}
Components.schematic.{OR}
Components.schematic.{BUF}

These are the boxes in the upper part of the left-side display panel in Electric.
Starting with an instance of one of these, we invert its output to change it to CMOS-style:

^{output crosshair}
Edit.TechnologySpecific.TogglePortNegation

NB--There must be a wire already attached to the gate's output for this to work.

Make a testbench which sets all possible inputs for the NOR. Record the input and output results as a truth table. You will need reg types to drive the gate's inputs: in your test bench define a "reg" for each input, use "assign" to connect the reg-type to the input wire, and assign values to those regs in an "initial" statement. Do the same for the NAND. Turn in the truth tables on paper and checkin the circuits to you branch (perhaps use a separate library).

(Problem)----------

In Electric, implement a NAND-NAND latch. Test its behavior by driving your circuit with all possible sequences of inputs. For latches, the history of inputs is important, not just the current input, because latches have state. That is, they remember what happened before. Do the same for the NOR-NOR latch, and comment on the difference between the NAND-NAND latch and the NOR-NOR latch. Turn your answers in on paper and check in your testbench and its output.

Note: For the NAND-NAND latch, we are interested in 3-step input sequences that begin and end with A=1, B=1 (for NOR-NOR, A=0, B=0). For instance, here is a 3-step NAND-NAND input sequence: (1,1), (0,1), (1,1). Here is another that is particularly interesting:

```
Asrc = 1;
Bsrc = 1;
#1
Asrc = 0;
Bsrc = 0;
#1
Asrc = 1;
Bsrc = 1;
```

That might give different results from the sequence below (why? does this mean we should use random delays for signal propagation through our gates to be more physically realistic? HINT--metastability problem):

```
Asrc = 1;
Bsrc = 1;
#1
Bsrc = 0;
Asrc = 0;
#1
Asrc = 1;
Bsrc = 1;
```