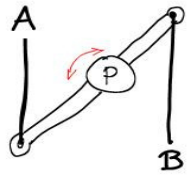
Lec-6-HW-1-devices

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

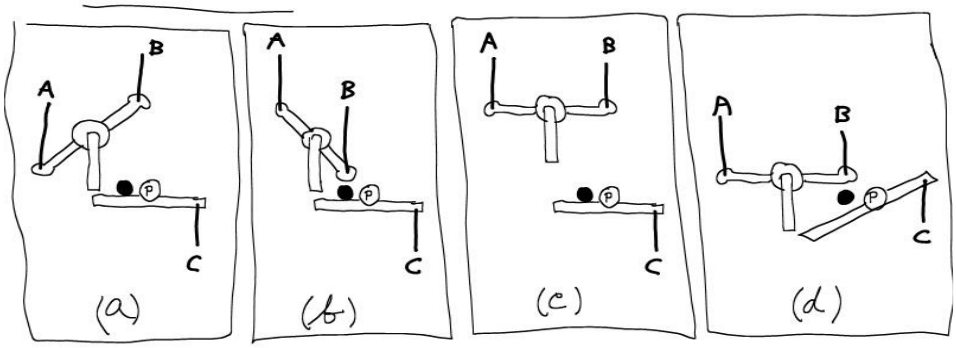
Problems, PP, Chp 3:
3.1 (n and p transistors),
3.2 (cmos inverter),
3.3 (how many 2-input functions?),
3.5 (trans. ckt. => truth table),
3.6 (as 3.5, but tricky ckt.),
3.7 (fix broken trans. NAND ckt.),
3.8 (label ckt. to match func.),
3.9 (expression to truth-table),
3.10 (NOR truth-table),
3.11a (trans. 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])

Device 1

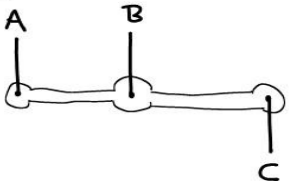


P is a fixed pivot.

Device 2



Device 3



(Problem)----------
Device 1 has a rigid arm attached to a fixed pivot at its center. The pivot cannot move up-down, nor left-right. If you pull up on rod A, rod B will move down. If we assume up is logical 1 and down is logical 0, what type of logical device might this be, supposing A is the input and B is the output? Explain briefly.

(Problem)----------
Device 2 has two inputs, A and B, and a single output C. These are again rods with pivots; however, only the pivot "P" is fixed. The black dot is a fixed peg that stops the clockwise rotation of the arm C is attached to. Assume A and B have nominal values (eg., "up one mm" and "down one mm" are logical 1 and logical 0). The possible inputs and their effects are shown. C is always down, except in case (d): if both A and B go down, C goes up.Give a truth table for this device. Does the function it implements have a name?

(Problem)----------
Device 3 has no fixed pivots. If A and B move up or down together, the entire arm moves up or down accordingly, remaining horizontal, and C moves likewise. Otherwise, the arm rotates, eg., if A goes up and B goes down, the arm rotates clockwise, moving C down. Show its transfer function, assuming the B input is fixed (either up or down, you pick). A transfer function shows how the output changes for small increments of input change. So, show C's position as A changes little by little from down to up. Display as a graph with A's position on the x-axis and C's position on the y-axis. Is this a linear or non-linear device? Assume any convenient sizes for the arm and the distance between begin down and up.

(Problem)----------
Suppose we were to build a machine using many copies of Device 1 and Device 2. Comment on error propagation and power requirements and the feasibility of scaling such a machine to very large capacity (ie., scaling up to lots of bits). Recall that moving a mass requires work (ie., energy = force X distance) to accelerate and decelerate the machine parts. Is Device 3 susceptible to error amplification? Imagine many of these connected together in a long chain.

(Problem)----------
Basic CMOS gates are NOR, NAND, NOT. Electric comes with three basic gates:

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

Invert the outputs to get (NAND, NOR, NOT):

^{output crosshair}
Edit.TechnologySpecific.TogglePortNegation

(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 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 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;
```