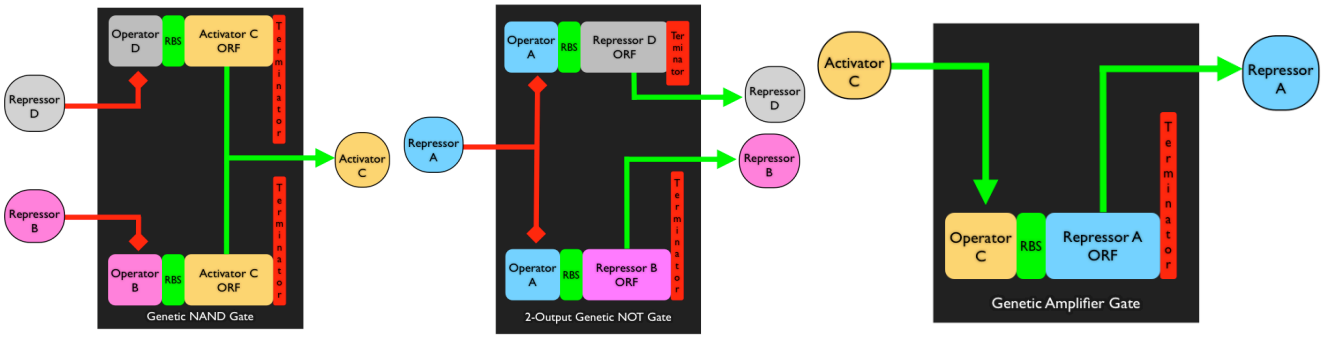
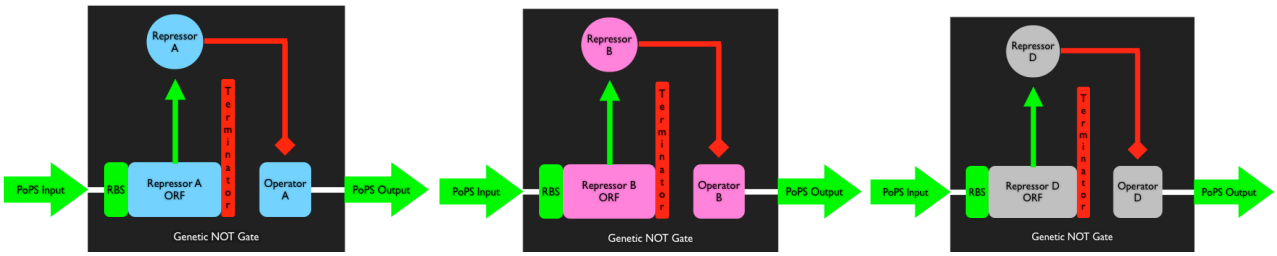
20.180 Exam 1 Name: Answer Key
23 March 2006
50 minutes, closed book

Question 1 (40 points):
BioBricks & Mortar, Inc. has been having some staffing problems in their Department of Abstraction; they've produced the following set of bizarre genetic devices:

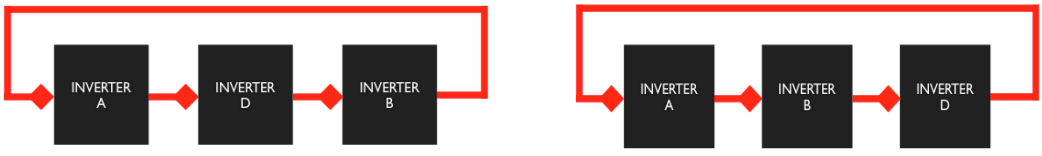


Using only the parts from these devices, remake three functionally independent and reusable inverters – include a sketch for how each inverter device works, including labels on device input and output signals and internal device logic. Then, sketch (at the device level) two different ring oscillator systems that can be built from your inverters. Finally, note that a simpler oscillator can be made connecting the output of an amplifier device to its own input, while simultaneously regulating the input to the amplifier via an inverter that is also connected to the amplifier's output. Define a reusable amplifier device and sketch (at the device-level) a two-device oscillator.

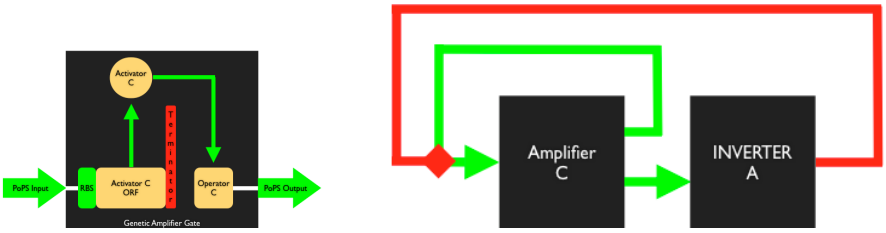
Three functionally independent and reusable inverter devices made from the above parts:



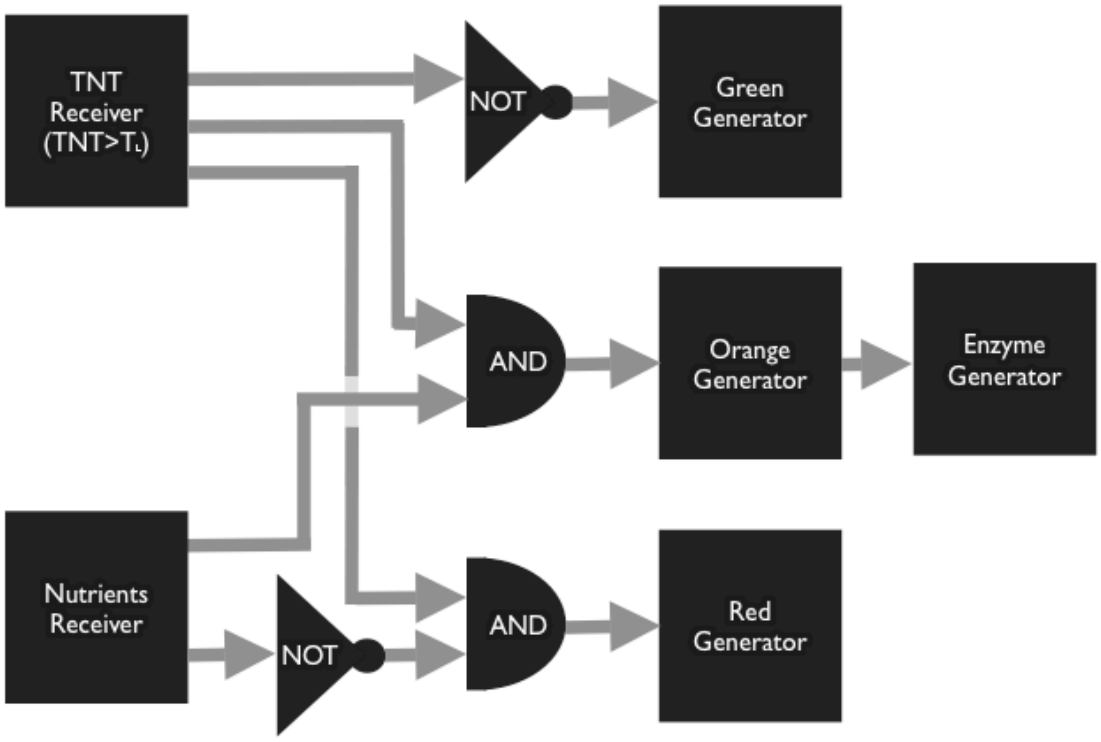
Two different ring oscillators built from the above inverters:



A PoPS-based amplifier device and a two-device oscillator:



Question 2 (30 points):
General Biologics, Inc. has asked you to design a microorganism that can detect and respond to levels of TNT in the environment. Above some threshold level of TNT (T_L), and only if nutrients are present in the local environment, the microorganism should turn orange and express enzymes that degrade TNT. However, if the TNT level is above T_L but nutrients are not present in the environment, the microorganism should do nothing except turn red. In the absence of TNT (i.e., for TNT < T_L) the organism should be green. Using PoPS-based devices, sketch out the device-level block diagram for the design of such an engineered biological system. Do not define the individual parts that make up the devices.



20.180 Exam 1 Name: Answer Key

Question 3 (30 points):

Pfizer, Inc. has given you the sequence of a genome containing many open reading frames (ORFs). Each of the genome's ORFs starts with "ATG" and ends with "UAGUGA." They want you to find each ORF and append the DNA sequence "TAGTGCTA" to the end (you're told not to worry about minimum ORF length). Meanwhile, you and your friends have designed another genome that contains 50 new synthetic DNA binding proteins. Each of these DNA binding protein ORFs starts with "ATG" and ends with "UAAUAA." Your friends want you to find each DNA binding protein ORF and add the DNA sequence "GTGTAGATCA" to the end. Write down a single Python program that can perform both of these tasks. Please note any instances of reusable code.

```
#
# Declare two reusable functions, one for finding ORFs and a second for suffixing.
#
# ORF finding function
#
def find_ORFs(seq, start_seq, stop_seq):
   found_ORFs = [] # start a new list
   tmp = seq       # create temporary variable to store sequence
   while True:     # create loop
      start = tmp.find(start_seq) # look for a start, set start to first occurence
      if start == -1: break       # if no start sequence found break loop
      tmp   = tmp[start:len(seq)] # redefine tmp sequence
      stop  = tmp.find(stop_seq)  # look for a stop, set stop to first occurence
      if stop  == -1: break       # if no stop sequence found break loop
      tmpORF= tmp[0:stop+len(stop_seq)] # create tmpORF from start & stop points (keep stop)
      if len(tmpORF)%3==0:        # check is stop sequence is in frame
         found_ORFs.append(tmpORF)# if OK then add to found orf list
      tmp   = tmp [3:] # set new starting boundary of sequence to downstream of last found start
   return found_ORFs   # send the ORFs back to main
#
# ORF suffixing function
#
def suffix_ORFs(ORF_list, suffix_seq):
   suffixed_ORFs = [] # start a new list
   for i in range(len(ORF_list)): # for all the found ORFs
      suffixed_ORFs.append(ORF_list[i]+suffix_seq) # add the suffix
   return suffixed_ORFs # send the modified ORFs back to main

#
# Main Program
#
# Read in genome sequences; define start, stop, and suffix sequences
#
friends        = open('friends.txt').read()
friends_start  = 'atg'
friends_stop   = 'uaauaa'
friends_suffix = 'gtgtagatca'

# call ORF finding function
friends_ORFs     = find_ORFs(friends, friends_start, friends_stop)
new_friends_ORFs = suffix_ORFs(friends_ORFs, friends_suffix)

#
# Repeat the above for other like tasks (i.e., Pfizer)
#
```