You may work in pairs for this assignment. If you choose to work with a partner, make sure only one of you makes a submission, and that the submitted Logisim file lists names and PIDs for both of you as described in the assignment below.

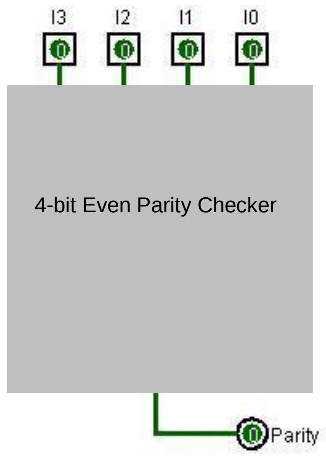
Follow the instructions given below to prepare your solutions to the questions. Your submission will be a single LogiSim file containing circuits solving the problems posed below. Do not tar or zip the file. Submit your file to the Curator system by the posted deadline for this assignment. No late submissions will be accepted.

You will submit your answers to the Curator System (www.cs.vt.edu/curator) under the heading `DL02`.

For each question, design and use Logisim to implement and test the specified logic circuit. A shell Logisim file with three circuits is available from the course website. You will use that to prepare your solutions. If you work in pairs, be sure to edit the text boxes in the Parity Checker circuit and list names and PIDs for both partners. The Text tool is in the Base library in Logisim.

Obey any restrictions that are imposed in the individual questions. Input and output components should be given descriptive labels. Input and output pins should display the bits in the order shown in the notes; that is, data bits should be shown in the usual left-to-right order with the high-order bit at the left and parity bits should be shown in left-to-right order as well.

1. [30 points] Design and implement an even 1's parity checker circuit for a 4-bit input:



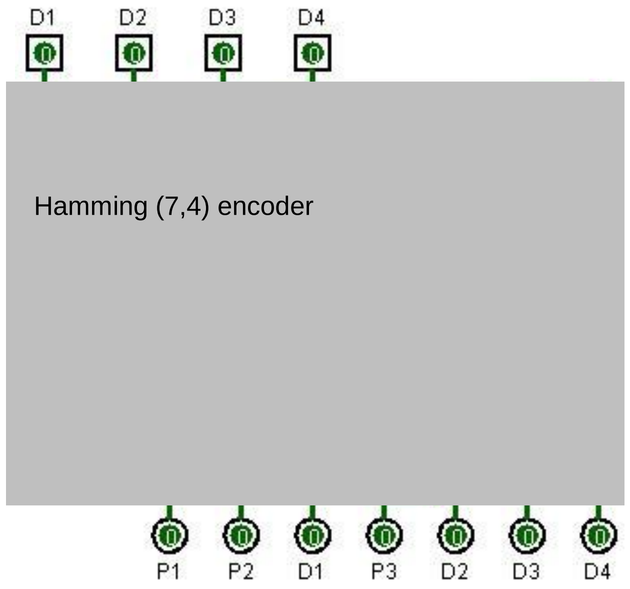
The circuit should output 1 if and only if the inputs contain an odd number of 1's. Your solution must conform to the interface shown above; you can replace the "black box" labeled "4-bit Even Parity Checker" with your implementation of the circuit internals.

Restrictions: You may use any logic gates (except for the even and odd parity components), wiring components, and input/output pins you like, but you may not use any components from the other Logisim libraries.

Be sure to keep your input/output pins in the same general orientation shown above; we will evaluate this by using Logisim to generate a truth table for your circuit, and then checking the table, and it's vital that your columns are in the same order as ours. That's determined by the layout you use.

Hint: If you look at this the right way, you can implement it using exactly three 2-input logic gates. The best way to analyze this is NOT to form a truth table for all 16 input cases, but to consider logically how you would do this for smaller input sizes.

2. [30 points] Design and implement a circuit, `Hamming(7,4)`, that takes an 4-bit data input and computes the Hamming(7,4) encoding of the input. You must conform to the following interface:

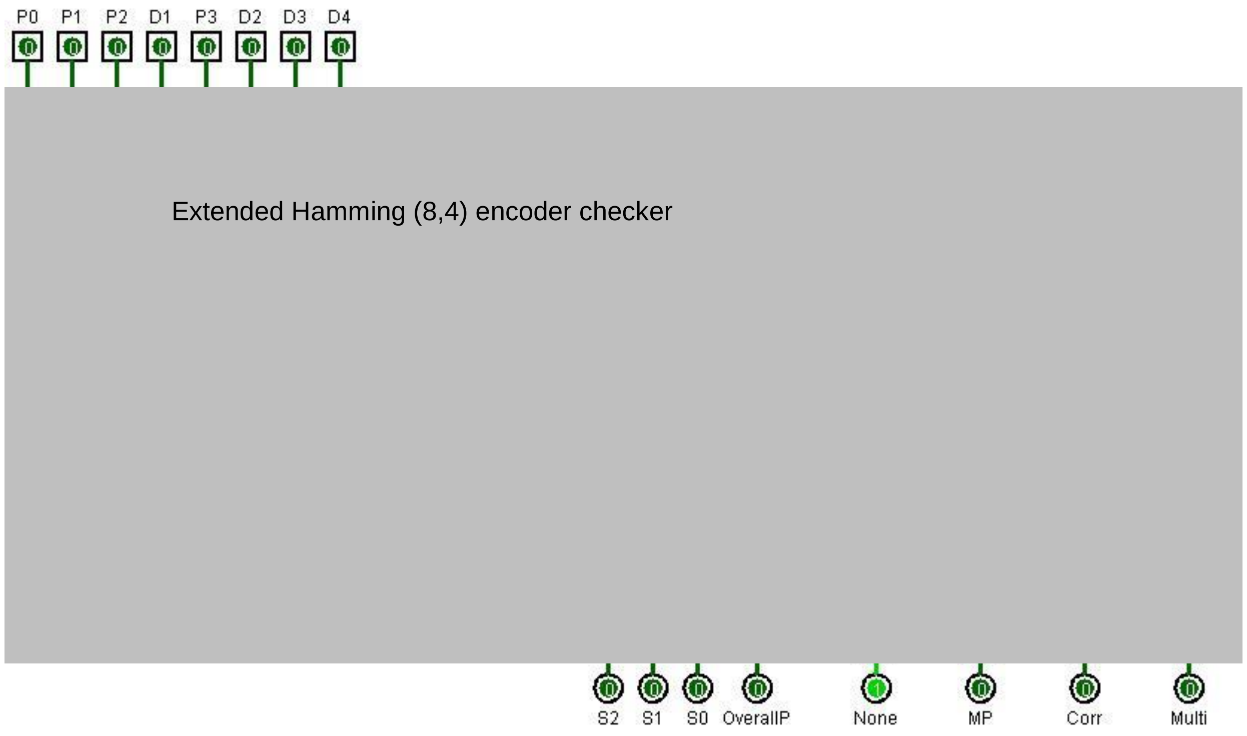


The circuit has an 4-bit input (D1, D2, D3, D4) and a 7-bit output (P1, P2, D1, P3, D2, D3, D4), that conform to the description of the Hamming (7,4) encoding scheme described in class.

As with question 1, you must keep the layout of the input/output pins that is shown above. If you use Logisim to generate the truth table for your solution, the input columns should be in the order D1-D4 and the output columns should be in the order P1-D4 as shown.

Restrictions: You may use any logic gates, wiring components, and input/output pins you like, but you may not use any components from the other Logisim libraries. Your solution to the first question could also be useful here, but you may use the Logisim parity components if you like.

3. [40 points] Design and implement an Extended Hamming (8,4) circuit, `EHamming(8,4)`, that extends your logic from question 2 by adding a master parity bit. This circuit will take an 8-bit input (presumably a proper Hamming (8,4) encoding, and determine whether or not there is a detectable error. You must conform to the following interface:



The circuit has an 4-bit input (P0-D4) and a 8 1-bit outputs:

- S2-S0 represent the computed syndrome for the input; be careful of the order here, since S2 must be the "high" bit of the syndrome.
- OverallP gives the parity of the entire bit sequence (including P0).
- None is 1 if, and only if, no errors can be detected.
- MP is 1 if, and only if, there is an error in the master parity bit.
- Corr is 1 if, and only if, a correctable 1-bit error is detected (somewhere other than the master parity bit).
- Multi is 1 if, and only if, an uncorrectable error is detected.

The logic for all this is covered in the lectures and the specification of the Hamming programming project.

As with questions 1 and 2, you must keep the layout of the input/output pins that is shown above. If you use Logisim to generate the truth table for your solution, the input columns should be in the order P0-D4 and the output columns should be in the order S2-Multi as shown.

Restrictions: You may use any logic gates, wiring components, and input/output pins you like, but you may not use any components from the other Logisim libraries. Your solution to the first question could also be useful here, but you may use the Logisim parity components if you like.