LSU EE 4755 Homework 4 Due: 6 November 2023

Collaboration Rules

Each student is expected to complete his or her own assignment. It is okay to work with other students and to ask questions in order to get ideas on how to solve the problems or how to overcome some obstacle (be it a question of Verilog syntax, how a part of the problem might be solved, etc.) It is also acceptable to seek out digital design resources for help on Verilog, digital design, etc. It is okay to make use of AI LLM tools such as ChatGPT to answer these questions. Just don't trust the answers. (Do not assume LLM output is correct. Treat LLM output the same way one might treat legal advice given by a lawyer character in a movie: it may sound impressive, but it can range from sage advice to utter nonsense.)

After availing oneself to these resources **each student is expected to be able to complete the assignment alone**. Test questions will be based on homework questions and **the assumed time needed to complete the question will be for a student who had solved the homework assignment on which it was based**.

Helpful Examples

See the simple model slides for material on computing cost and delay, and also for a list of some sample problems. Also see 2022 Homework 3.

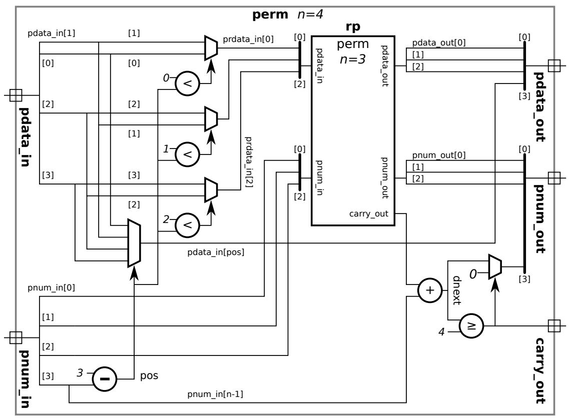
Permutation Module

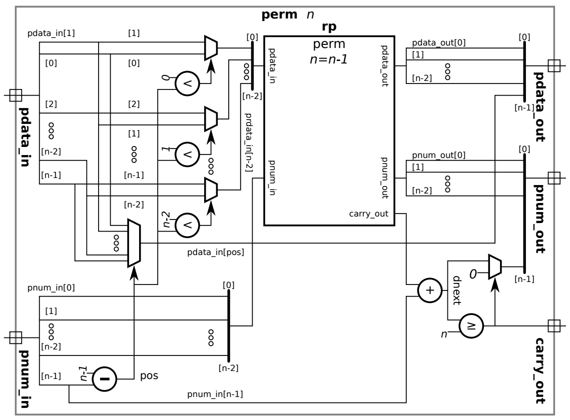
This assignment is based on the solution to Homework 3, the recursive permutation module `perm`, and the solution to Midterm Exam Problem 1, the inferred hardware for the permutation module. See Homework 3 for details on what the permutation module does. Appearing below is the Homework 3 solution with some comments removed. For the unabridged version visit `https://www.ece.lsu.edu/koppel/v/2023/hw03-sol.v.html`.

```
module perm
  #( int w = 8, n = 20, wd = $clog2(n) )
   ( output uwire [w-1:0] pdata_out[n],    output uwire [wd-1:0] pnum_out[n],
     output uwire carry_out,
     input uwire [w-1:0] pdata_in[n],       input uwire [wd-1:0] pnum_in[n] );

   if ( n == 1 ) begin

      assign pdata_out[0] = pdata_in[0];
      assign carry_out = 1;
      assign pnum_out[0] = 0;

   end else begin

      //  Set pos to the position of the element to be moved.
      uwire [wd-1:0] pos = n - 1 - pnum_in[n-1];

      //  Copy the element at position pos to position n-1 in the output.
      assign pdata_out[n-1] = pdata_in[pos];

      //  Prepare an array of n-1 elements and set to ..
      //  .. the elements of pdata_in except for the element at pos.
      uwire [w-1:0] prdata_in[n-1];
      for ( genvar i=0; i<n-1; i++ )
        assign prdata_in[i] =  i < pos  ?  pdata_in[i]  :  pdata_in[i+1];

      // Recursively instantiate perm.
      uwire co;
      perm #(w,n-1,wd) rp( pdata_out[0:n-2], pnum_out[0:n-2], co,
                           prdata_in, pnum_in[0:n-2] );

      //  Compute a tentative next value of digit n-1.
      uwire [wd-1:0] dnext = pnum_in[n-1] + co;

      //  Determine whether there is a carry.
      assign carry_out = dnext >= n;

      //  Set the next value of digit n-1 based on whether there is a carry.
      assign pnum_out[n-1] = carry_out ? 0 : dnext;

   end

endmodule
```

Permutation Module Inferred Hardware

Midterm Exam Problem 1 asked for the inferred hardware for the `perm` module instantiated with `n=4`. The solution appears below on the left. For this assignment the inferred hardware for a non-specific value of `n` will be needed, that is shown on the right.





There's no need to squint, the diagrams appear again in larger size at the end of this assignment. Also, SVG source for these modules are at `https://www.ece.lsu.edu/koppel/v/2023/mt-p1-sol.svg` and `https://www.ece.lsu.edu/koppel/v/2023/hw04-perm-gen.svg`.

Problem 1: Compute the cost and delay of the following arithmetic hardware from the `perm` module. Assume that ripple units are used for addition, subtraction, and comparison.

(*a*) Compute the cost and delay of the hardware computing `pos = n - 1 - pnum_in[n-1]` in terms of w_d, the value of parameter `wd`. Optimize for constants, including `n`.

☐ Cost of hardware in terms of w_d. ☐ Delay of hardware in terms of w_d.

☐ Optimize for constants, don't confuse elaboration-time computation with computation hardware.

(*b*) Compute the cost and delay of the hardware computing `dnext = pnum_in[n-1] + co` in terms of w_d, the value of parameter `wd`. Optimize for constants and for the size of `co`. Assume in this problem that `pnum_in` and `co` arrive at $t = 0$.

☐ Cost of hardware in terms of w_d. ☐ Delay of hardware in terms of w_d.

☐ Optimize considering the size of `co`. ☐ Optimize for constants, don't confuse elaboration-time computation with computation hardware.

(*c*) Compute the cost and delay of the hardware described by these lines:

```
uwire [wd-1:0] dnext = pnum_in[n-1] + co;
assign carry_out = dnext >= n;
```

Assume in this problem that `co` and `pnum_in` arrive at $t = 0$. The cost, of course, includes the cost of computing `dnext` in the previous part. The delay must be computed taking both lines into account.

☐ Cost of hardware in terms of w_d. ☐ Delay of `co` in terms of w_d.

☐ Optimize considering the size of `co`. ☐ Optimize for constants, don't confuse elaboration-time computation with computation hardware.

There are more problems on the next page.

Problem 2: In this problem consider the multiplexors with inputs connecting to `pdata_in`. (In the diagram they are the multiplexors on the upper-left including the 2-input muxes the n-input mux.) Call these the pdata multiplexors. In the solutions to the parts below use w for the value of parameter `w` and w_d for the value of parameter w_d.

(a) Compute the cost of the pdata multiplexors for a module instantiated at size $n = N$ including only the hardware in the `n=N` instantiation, not in the recursive instantiations. The answer should be in terms of N and w. *Hint: this is easy.*

☐ Cost of the pdata multiplexors at one level in terms of N, w, and (if needed) w_d.

(b) **This is important. Expect to expend brain energy. Don't skip.** Compute the total cost of the pdata multiplexors for an instantiation at size $n = N$ including the recursive instantiations all the way down. The answer should be in terms of N and w.

☐ Cost of the pdata multiplexors including recursive instantiations in terms of N, w, and (if needed) w_d.

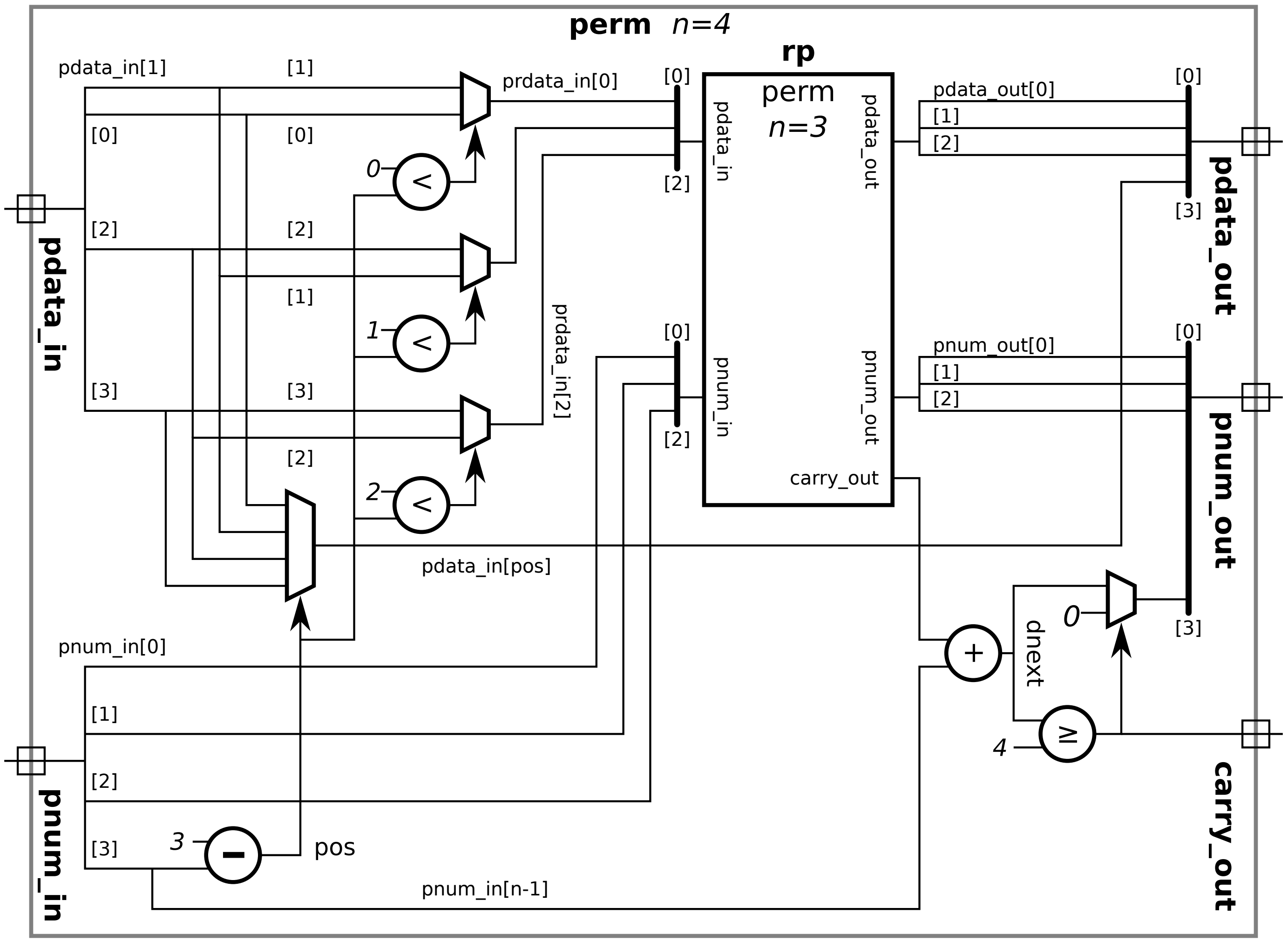
Problem 3: In this problem compute delays for `pdata_out` and `pnum_out`. In the solutions use d for the value of parameter `wd`. **This is also important and even more interesting. Expect to expend brain energy. Don't skip.**

(a) Assume that the delay of the subtractors computing `pos` is $\lg w_d$, where w_d is the value of parameter `wd`. (Note that $\lg w_d$ is not an answer to Problem 1.) Further, suppose the delay of the less-than units providing a select signal to the 2-input pdata multiplexors is zero. Using these assumptions compute the delay of the first and last elements of `pdata_out` for an instantiation at `n=N` and show the critical path. The delay should be in terms of N and w_d. To solve this problem it might be helpful to draw two instantiation levels to help find the critical path.

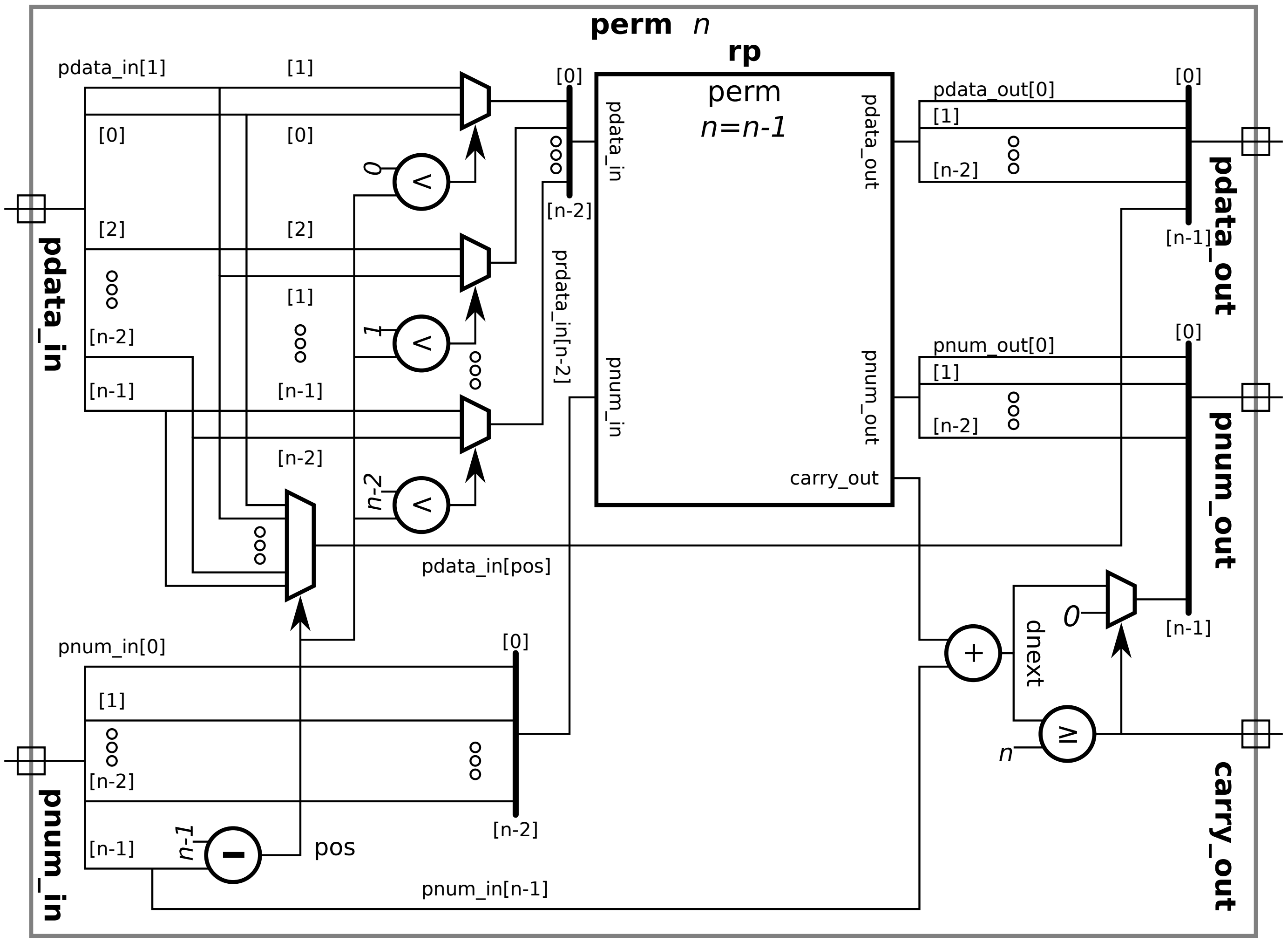
☐ Delay of `pdata_out[0]` in terms of N and w_d accounting for recursive instantiations. ☐ Show critical path.

☐ Delay of `pdata_out[N-1]` in terms of N and w_d accounting for recursive instantiations. ☐ Show critical path.

SVG source for the module below is at
`https://www.ece.lsu.edu/koppel/v/2023/mt-p1-sol.svg`.



SVG source for the module below is at
`https://www.ece.lsu.edu/koppel/v/2023/hw04-perm-gen.svg`.


perm n
rp
perm
n=n-1
pdata_in[1]
[1]
[0]
[0]
[0]
[2]
[2]
[1]
[n-2]
[n-1]
[n-1]
[n-2]
[n-2]
pdata_in
0
1
n-2
pdata_in
pdata_out
prdata_in[n-2]
pnum_in
pnum_out
carry_out
pdata_out[0]
[1]
[n-2]
[0]
[n-1]
pdata_out
pnum_out[0]
[1]
[n-2]
[0]
pnum_out
[n-1]
pdata_in[pos]
0
dnext
n
carry_out
pnum_in[0]
[0]
[1]
[n-2]
[n-2]
[n-1]
n-1
pos
pnum_in[n-1]
pnum_in