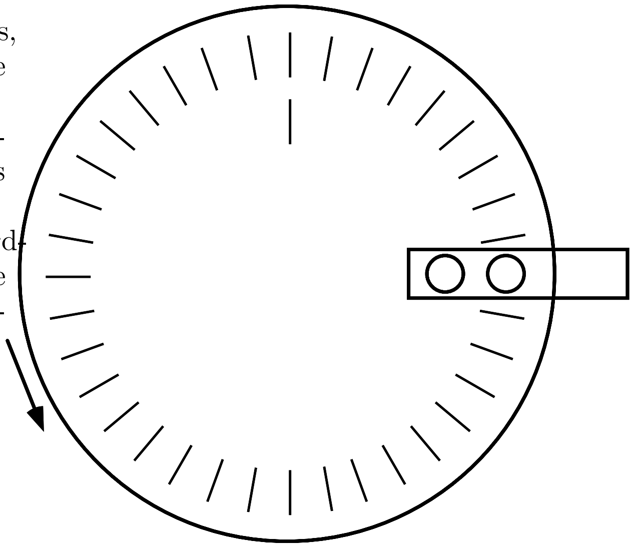
EE 4702 **Homework 2** **Due: 23 February 2000**

Homework 2 and 3 are being assigned simultaneously. Homework 3 is really just homework 2a, but calling it that would ruin the numbering scheme. Solution templates can be found in `/home/classes/ee4702/files/v` *and will be linked to the web page. Instructions for submission will be posted later.*

Problem 1: A tachometer measures rotation rate by detecting marks on a disk using photodetectors as illustrated below.



In the illustration there are two rings of marks, in this assignment only the outer ring (the one with lots of marks) will be used.

As the disk spins the number of marks passing under the disk are counted. At fixed intervals a rotation rate is updated.

Write a Verilog behavioral description for hardware that determines the rotation rate using the photodetector output. The module has the following declaration:

```
module tach1(rpx,pd,clk);
   input pd, clk;
   output rpx;
   wire   pd, clk;
   reg [9:0] rpx;

   parameter  freq = 500;    // Clock frequency.
   parameter  marks = 4;     // Number of marks on ring.
   parameter  update_interval = 0.5;  // Update every update_interval seconds.
   parameter  perwhat = 60;  // Measure in revolutions per 60 seconds.

// Solution goes here.

endmodule
```

Input `clk` is a square wave for use by the module. Input `pd` is the photodetector output. It is `1` when a mark is under the photodetector. Output `rpx` is the rotation rate. Parameter `freq` is the frequency of `clk` and `marks` is the number of marks on a disk. Parameter `update_interval` is the number of seconds between updates of `rpx`. For example, if `update_interval` were 3 then `rpx` would have to be updated every 3 seconds. Parameter `perwhat` is the time unit for measuring revolutions, in seconds. If it is `60` then `rpx` should be in revolutions per minute, if it is `1` then `rpx` should be in revolutions per second, etc.

Consider the instantiation below:

```
   tach1  #(200) s1(rpx,pd,clk);
```

This instantiates a tachometer which is to use a 200 MHz clock.

Call $\frac{\texttt{perwhat}}{\texttt{marks}\times\texttt{update\_interval}}$ the *precision*, p. Let n_m denote the number of marks that have been counted in a time interval of duration `update_interval`. Then `rpx` should be set to $n_m \times p$.

In addition to generating `rpx` the module should also check to make sure its parameters are suitable. The parameters are not suitable if the precision is not an integer or if any registers would overflow in normal operation.

Use the testbenches provided in the solution template to test your circuit. Testbench module `test_tach1_fast` tests a single instance, while `test_tach1_detailed` tests several instances (using different parameters).

Follow the following rules when writing the hardware description. (The rules do not apply to testbench code.)

- Do not use multipliers or dividers.
- Do not use delays: #`3 i=1;`. You can use event controls: `@(posedge clk)`.
- Use the initial block for parameter verification and register initialization only.

To be continued in homework 3 . . .