

Re: Phase frequency detector

Source: <http://sci.tech-archive.net/Archive/sci.electronics.design/2004-06/1088.html>

From: Mike (mike_at_nospam.com)

Date: 06/09/04

Date: Tue, 8 Jun 2004 22:14:34 -0700

On Tue, 8 Jun 2004 07:17:15 -0700, Mike wrote:

> On 7 Jun 2004 03:36:53 -0700, Mike Monett wrote:

>> Mike, I'm having difficulty following your description. Do you have a
>> schematic and timing diagrams, perhaps a SPICE model?

Here's a Python program that does the calculations and reports the results.
You'll have to undo the word wrapping on some lines. If you aren't familiar
with Python, the main thing you need to know is that indentation is
critical: don't mess with it.

```
#
# Python script to simulate UP and DN PFD outputs into bipolar diff pairs.
# The UP and DN signals are RC exponentials with RC time constant tau.
#

from math import *

#
# Units. These are a convenient fiction: it's easier to read 150*mV than
150e-3.
#
V = 1
mV = 1.0e-3

mA = 1.0e-3
uA = 1.0e-6

ns = 1.0e-9
ps = 1.0e-12
fs = 1.0e-15

pC = 1.0e-12
fC = 1.0e-15

Vt = 25.0*mV # Thermal voltage
Vo = 150.0*mV # ECL single ended output level (output swings between
```

```

+/-Vo)
Iee = 100*uA # Diff pair tail currents

tstep = 10.0*fs # simulation time step
tstop = 10.0*ns # simulation stop time

tdly = 0.0*ns # time delay before start of phase error
twid = 0.3*ns # width of UP and DOWN if the phase error is zero
tau = 1.0*ns # ECL output time constant – the PFD outputs rise and
fall
                # with this time constant
#
# Phase error (expressed as a time error)
#
terr = [ 10*fs, 30*fs, 100*fs, 300*fs, 1*ps, 3*ps, 10*ps, 30*ps, 100*ps,
300*ps, 1*ns ]

Qtot = [] # Total charge transferred
Kmax = floor(tstop/tstep)+1; # Number of steps in simulation

for tphi in terr:
    Q = 0.0
    time = 0.0
    k = 0
    print tphi,
    while k < Kmax:
        #
        # Leading edge of the PFD output – total pulse width is twid + tphi
        #
        if time < tdly:
            xp = 0.0
        elif time < tdly + twid + tphi:
            xp = 1.0 – exp(-(time–tdly)/tau)
        else:
            xp = (1.0 – exp(-(twid+tphi)/tau))*exp(-(time–tdly–twid–tphi)/tau)

        Vp = Vo*(-1.0 + 2.0*xp)
        #
        # Trailing edge fo the PFD output – total pulse width is twid
        #
        if time < tdly + tphi:
            xn = 0.0
        elif time < tdly + twid + tphi:
            xn = 1.0 – exp(-(time–tdly–tphi)/tau)
        else:
            xn = (1.0 – exp(-twid/tau))*exp(-(time–tdly–twid–tphi)/tau)

        Vn = Vo*(-1.0 + 2.0*xn)
        #
        # Calculate the pump currents from the diff pairs
        #

```

```
Ip = Iee/(1.0 + exp(-Vp/Vt)) # Up current
In = Iee/(1.0 + exp(-Vn/Vt)) # Down current
#
# Integrate the net current over time to find the total charge
transferred
# to the loop filter. The integration algorithm used is simple, but
the time
# step is small, so a more complex algorithm isn't necessary.
#
Q = Q + (Ip - In)*tstep

# Increment the iteration variables

k = k + 1
time = k * tstep

Qtot.append(Q) # Add the charge to the list
print Q
```