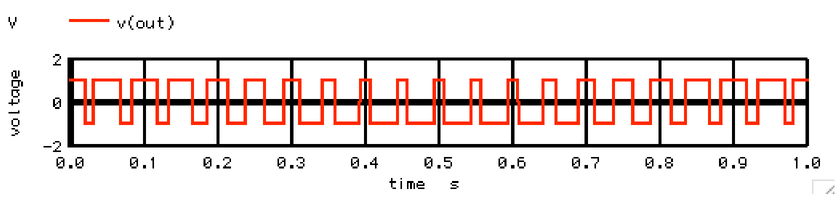
```
*=============PWM_WithOut_PM_Jitter============================
```

Can Pulse Width Modulation be done without jitter?
Good time for doing a jitter plot.

```
PWM_Without_PM_Jitter
* dsauersanjose@aol.com  5/07/08
* www.idea2ic.com
*
*
*     Vtime        ^  OUT
*           ____  / \
*  VT     _|_   |  _|_
*        /_  \  |  /_  \
*       // \  \ |_// \  \
*       \   \//   \   \//
*        \___/     \___/ B1
*          |         |
*          |         |
*         _|_       _|_
*         ///       ///
*
VP          VP        0     DC      3.141592653589793
VT          VT        0     PWL     ( 0 0 1 1 )
B1          TRI       0     V =     acos(cos(2*v(VP)*20*v(VT)))/v(VP)
B2          OUT       0     V =     2*u(1-v(TRI)*2  + .5*cos(1*v(VP)*2*v(VT)) )-1
*===========Run_Simulation=========================
.tran      .1m       1   0      .1m
.control
run
set         pensize = 2
plot        v(out)
*=================================================================
```



Find the edge transistions

```
*=========Create_AnySize_Arrays============================================
compose    anysize      start = 0 stop = 199 step =1
let        num =        length(out)-5
let i  =   0
let t  =   0
let n  =   0
*=========Find_Edge_Timing===========================================
repeat     $&num
if         ( out[i] < 0 & out[i+1] > 0)
let t =    time[i]
let        anysize[n]= t
echo       n=   $&n    out_rise= $&t
let        n =    n +1
endif
if         ( out[i] > 0 & out[i+1] < 0)
let t =    time[i]
let        anysize[n]= t
echo       n=   $&n    out_fall= $&t
let        n =    n +1
endif
let i =    i +1
endrepeat
*=================================================================
```

The MacSpice print out is such...

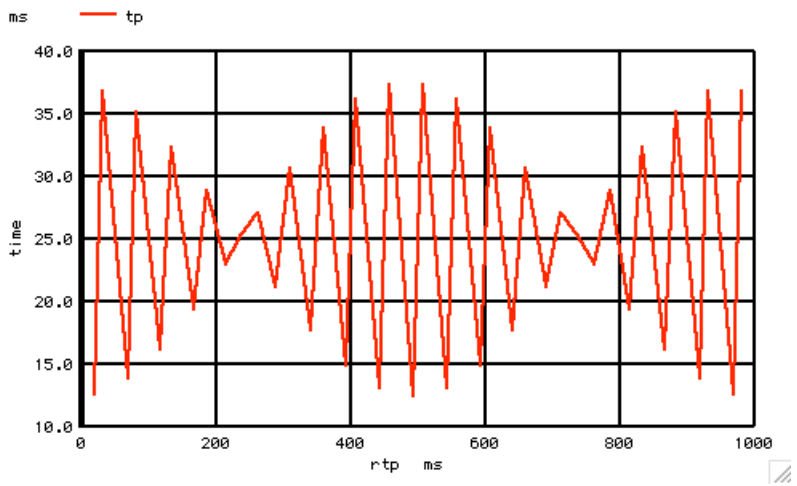
```
Circuit: PWM_Without_PM_Jitter

n = 0  out_fall = 0.0187024
n = 1  out_rise = 0.0313024
n = 2  out_fall = 0.0681024
n = 3  out_rise = 0.0820024
n = 4  out_fall = 0.117102
n = 5  out_rise = 0.133302
n = 6  out_fall = 0.165602
n = 7  out_rise = 0.185002
n = 8  out_fall = 0.213902
n = 9  out_rise = 0.236902
```

```
n = 10 out_fall = 0.262002
n = 11 out_rise = 0.289002
n = 12 out_fall = 0.310102
n = 13 out_rise = 0.340802
n = 14 out_fall = 0.358502
n = 15 out_rise = 0.392302
n = 16 out_fall = 0.407202
n = 17 out_rise = 0.443302
n = 18 out_fall = 0.456402
n = 19 out_rise = 0.493702
n = 20 out_fall = 0.506202
n = 21 out_rise = 0.543502
n = 22 out_fall = 0.556602
n = 23 out_rise = 0.592702
n = 24 out_fall = 0.607602
n = 25 out_rise = 0.641402
n = 26 out_fall = 0.659102
n = 27 out_rise = 0.689802
n = 28 out_fall = 0.710902
n = 29 out_rise = 0.737902
n = 30 out_fall = 0.763002
n = 31 out_rise = 0.786002
n = 32 out_fall = 0.814902
n = 33 out_rise = 0.834302
n = 34 out_fall = 0.866602
n = 35 out_rise = 0.882802
n = 36 out_fall = 0.917902
n = 37 out_rise = 0.931802
n = 38 out_fall = 0.968602
n = 39 out_rise = 0.981202
```

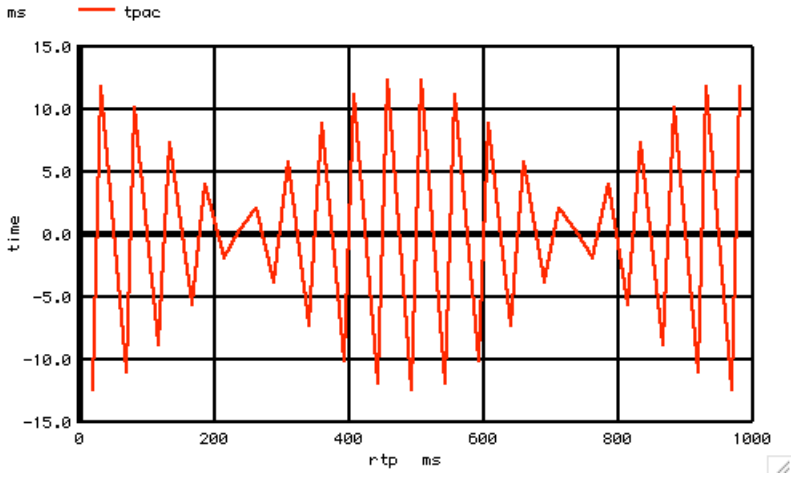
Transfer and plot the timing periods

```
*=========Create_Edge_Time_Arrays============================================
let       n1 =  n -1
let       n2 =  n -2
let       n3 =  n -3
compose   tp            start = 0 stop = $&n1 step =1
compose   tpac          start = 0 stop = $&n1 step =1
compose   rtp           start = 0 stop = $&n1 step =1
compose   td            start = 0 stop = $&n1 step =1
compose   tdac          start = 0 stop = $&n1 step =1
*=========Transfer_Arrays===================================================
let i  =  0
repeat    $&n
let       rtp[i] = anysize[i]
let i =   i +1
endrepeat
let i  =  0
let n2 =  n -1
repeat    $&n2
let       tp[i] = rtp[i+1] -rtp[i]
let i =   i +1
endrepeat
let       tp[n1] = tp[n1-2]
plot      tp  vs rtp
*============================================================================
```



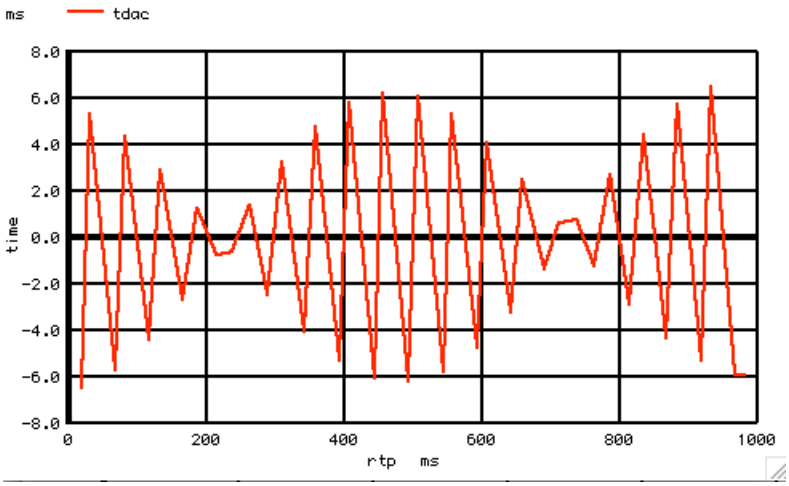
Remove average time period.

```
*=========Remove_Average_Time_Period============================
let       tpave = mean(tp)
let       tpac =  tp -tpave
plot      tpac vs rtp
*===============================================================
```



Convert FM to PM, remove average and plot

```
*=========Convert_FM_to_PM=====================================
let i  =  1
let n2 =  n -1
repeat    $&n2
let       td[i] = td[i-1] +tpac[i]
let i =   i +1
endrepeat
*=========Remove_Average_Phase=================================================
let       tdave = mean(td)
let       tdac  =  td -tdave
let       tdac[n1] = tdac[n1-1]
plot      tdac  vs rtp
*===============================================================
```




```
*===============================================================
```

The plot of the delays shows that for every delay for a rising edge there is an equal an opposite delay for the falling edge. The edges are experiencing a balanced type of jitter. And in doing so, there is no net phase modulation.

This says that the square wave is being modulated in terms of its asymmetry, but not in terms of its frequency or phase.

So as long as rising and falling edges always balance each other out, there actually appears to be no net phase shift.

So it appears there is a way to have jitter without phase modulation.

```
===========Full_Netlist_For_Copy_Paste=======================
PWM_Without_PM_Jitter
* dsauersanjose@aol.com  2.15.10_6.02PM
* www.idea2ic.com
*
*
*     Vtime       ^  OUT
*        ____    /_\
*  VT  _|_   |   _|_
*     / _ \  |  / _ \
*    // \  \ |_// \  \
*    \   \//   \   \//
*     \___/     \___/ B1
*       |         |
*      _|_       _|_
*      ///       ///
*
VP          VP        0     DC     3.141592653589793
VT          VT        0     PWL    ( 0 0 1 1 )
B1          TRI       0     V =    acos(cos(2*v(VP)*20*v(VT)))/v(VP)
B2          OUT       0     V =    2*u(1-v(TRI)*2  + .5*cos(1*v(VP)*2*v(VT)) )-1
*===========Run_Simulation=========================
.tran      .1m       1  0     .1m
.control
run
set         pensize = 2
plot        v(out)
*=========Create_AnySize_Arrays============================================
compose   anysize     start = 0 stop = 199 step =1
let       num =       length(out)-5
let i  =  0
let t  =  0
let n  =  0
*=========Find_Edge_Timing============================================
repeat    $&num
if        ( out[i] < 0 & out[i+1] > 0)
let t =   time[i]
let       anysize[n]= t
echo      n=   $&n   out_rise= $&t
let       n =    n +1
endif
if        ( out[i] > 0 & out[i+1] < 0)
let t =   time[i]
let       anysize[n]= t
echo      n=   $&n   out_fall= $&t
let       n =    n +1
endif
let i =   i +1
endrepeat
*=========Create_Edge_Time_Arrays============================================
let       n1 =  n -1
let       n2 =  n -2
let       n3 =  n -3
compose   tp           start = 0 stop = $&n1 step =1
compose   tpac         start = 0 stop = $&n1 step =1
compose   rtp          start = 0 stop = $&n1 step =1
compose   td           start = 0 stop = $&n1 step =1
compose   tdac         start = 0 stop = $&n1 step =1
*=========Transfer_Arrays============================================
let i  =  0
repeat    $&n
let       rtp[i] = anysize[i]
let i =   i +1
endrepeat
let i  =  0
let n2 =  n -1
repeat    $&n2
let       tp[i] = rtp[i+1] -rtp[i]
let i =   i +1
endrepeat
let       tp[n1] = tp[n1-2]
plot      tp  vs rtp
*=========Remove_Average_Time_Period============================================
```

```
let       tpave = mean(tp)
let       tpac =  tp -tpave
plot      tpac vs rtp
*=========Convert_FM_to_PM=============================================
let i  =  1
let n2 =  n -1
repeat    $&n2
let       td[i] = td[i-1] +tpac[i]
let i =   i +1
endrepeat
*=========Remove_Average_Phase=========================================
let       tdave = mean(td)
let       tdac  =  td -tdave
let       tdac[n1] = tdac[n1-1]
plot      tdac  vs rtp
.endc
.end
```

2.12.10_3.03PM
dsauersanjose@aol.com
Don Sauer
http://www.idea2ic.com/