

*An example of differential Voltage Controlled Oscillator (**VCO**) and Differential to Single Ended Converter (**D2S**)*

- DELAY CELL BIASING
- D2S CONVERTER
- VCO+D2S BEHAVIOUR
- COMPLETE VerilogHDL MODEL

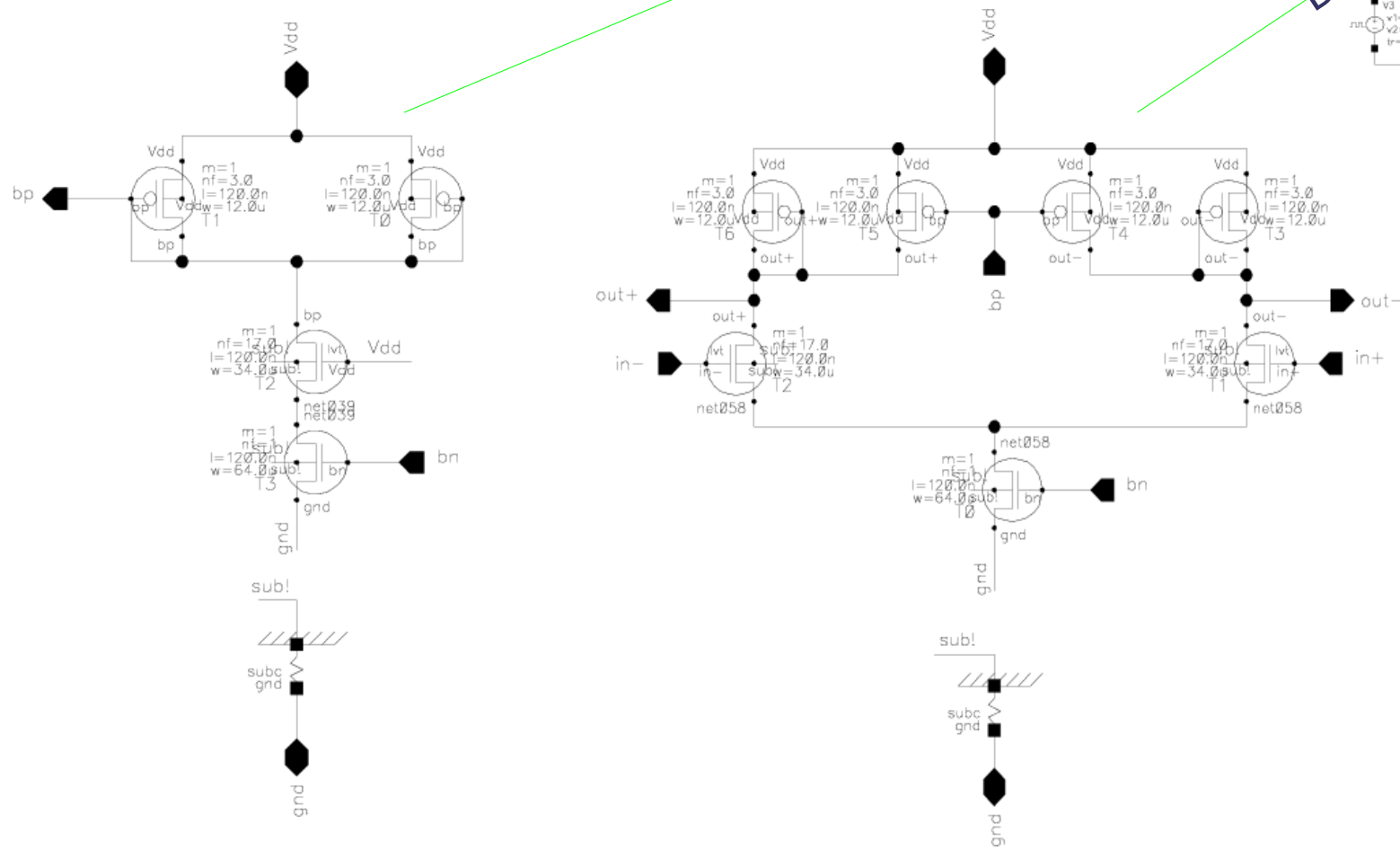
DELAY CELL BIASING

- Highest gain (and the centroids for Gaussians of ANY variation) should be @ trip point ($V_{in}=V_{out}=V_{dd}/2=600\text{mV}$)
 - Size the transistors to set the trip point to 600mV
 - Maximize gain (and centralize ALL kinds of variation) @ trip point
 - Consider process corners

TEST CIRCUIT

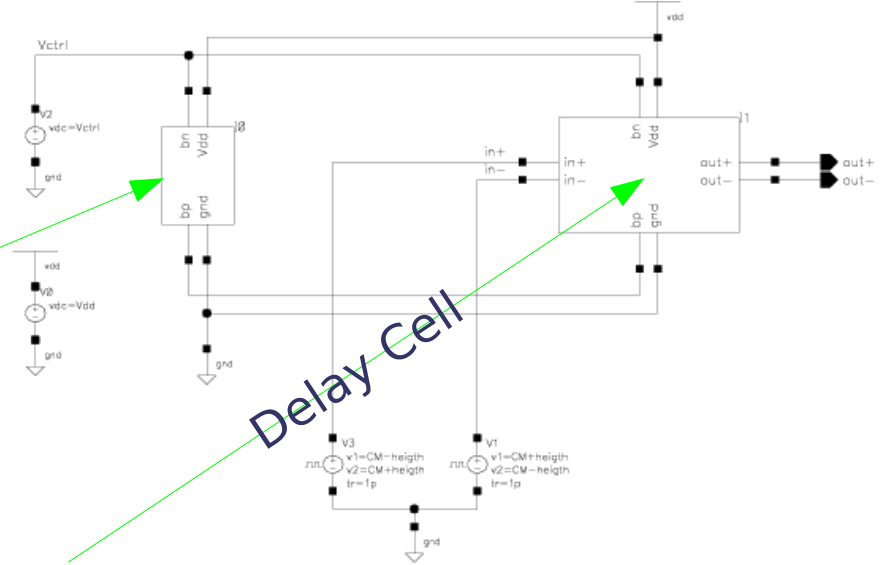
Size, Finger, Mult., Model

12u/120n, f=3, m=6, rf
34u/120n, f=17, m=1, lvt_rf
64u/1u, f=1, m=1, rf



Bias

Delay Cell



This is the circuit for the next page.

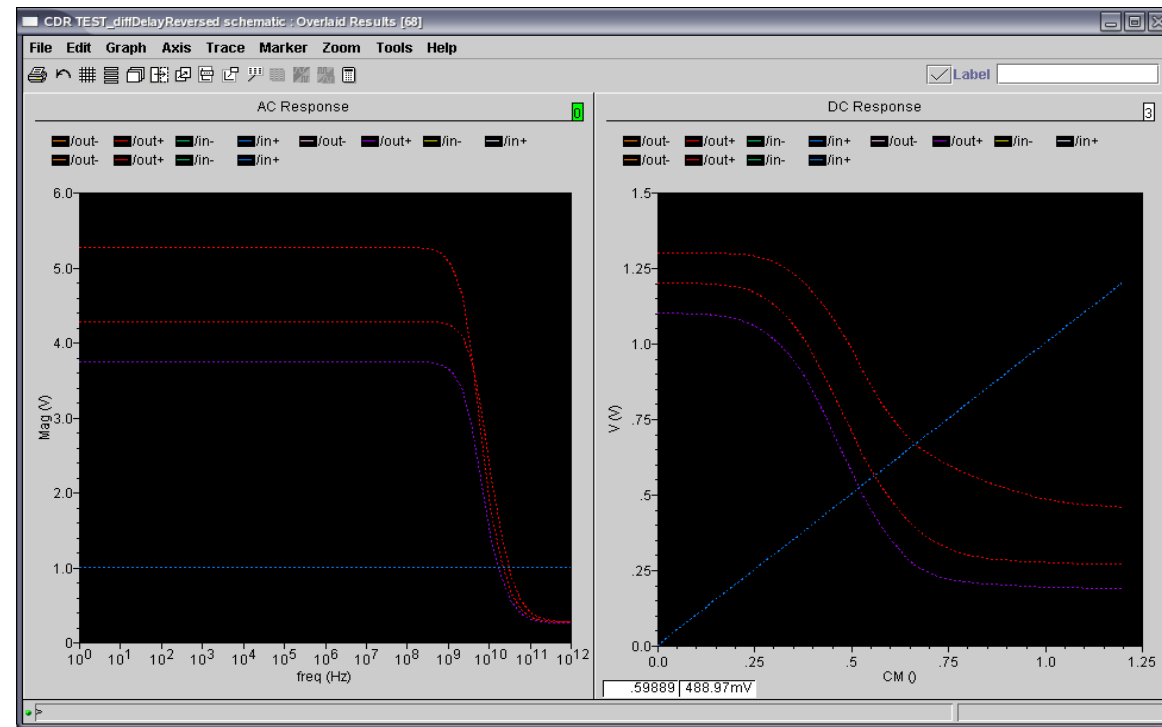
Ö.Ç.

High resolution pictures; zoom in.

DELAY CELL BIASING

Biassing point of the delay cell for typical mean and corners.

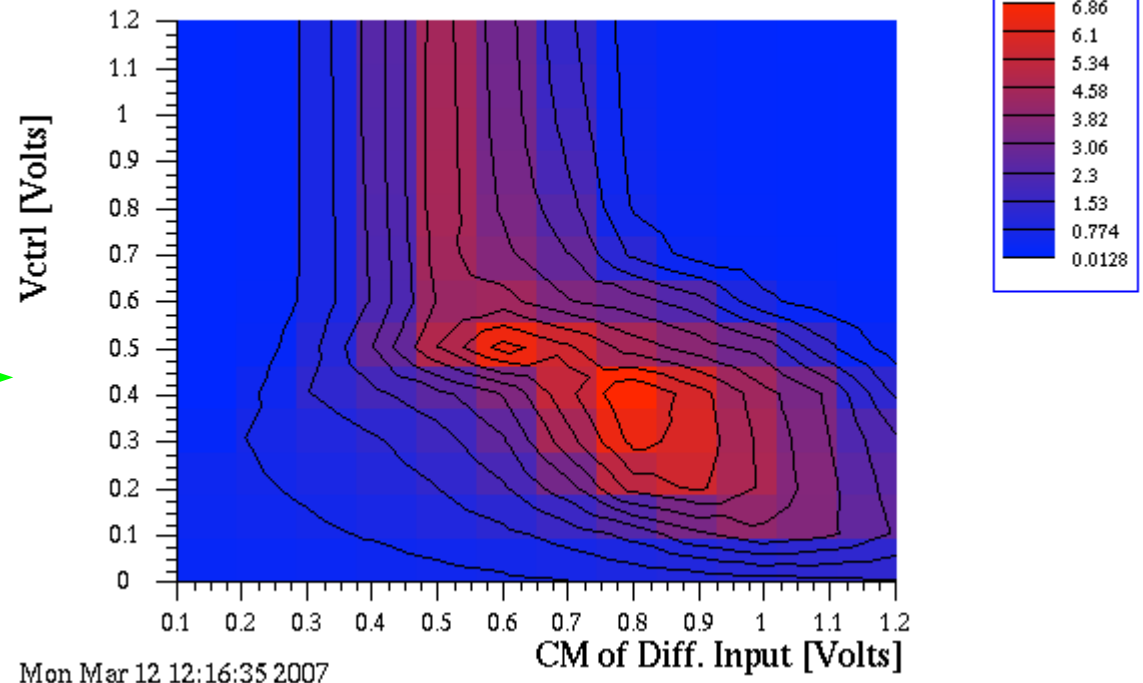
First plot is the AC gain as a function of frequency and the second one is the differential output as a function of differential input.



Variation of AC gain of the delay cell as a function of Vctrl and Common Mode of differential input (from AC analysis).

Worksheet

Delay Cell Biasing



High resolution pictures; zoom in.

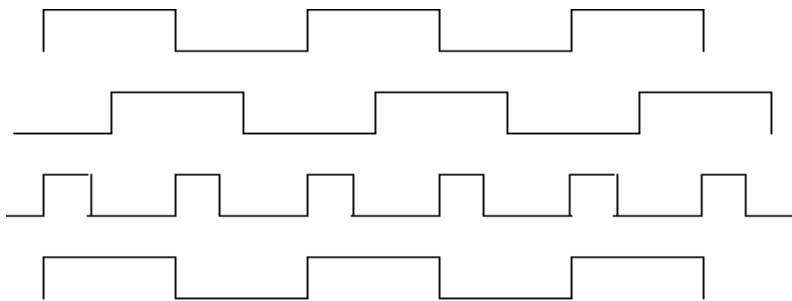
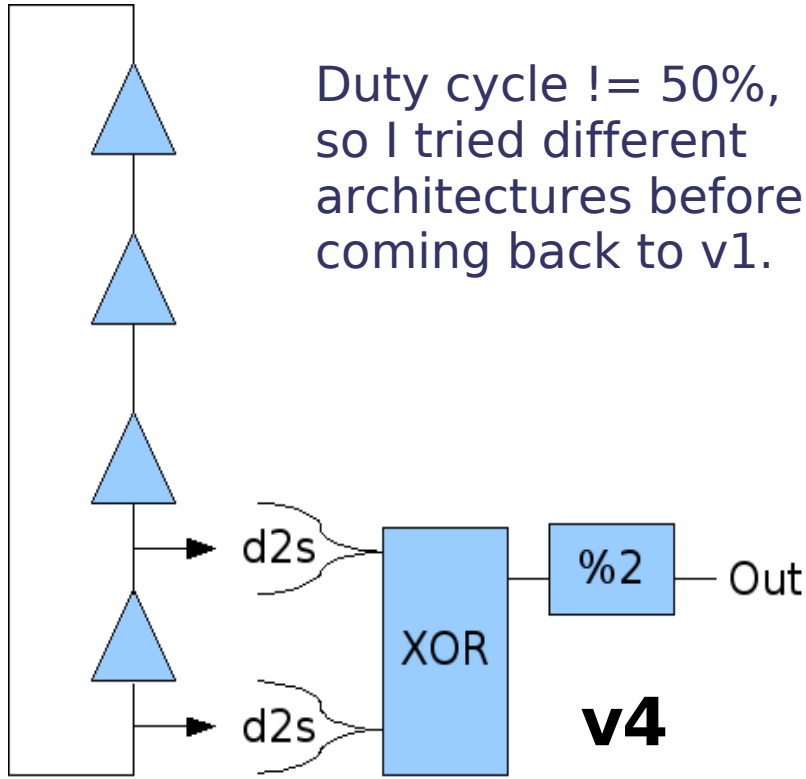
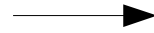
D2S CONVERTER

- Considering CM variation, proceed as for delay cell :
 - Highest gain (and the centroids for Gaussians of ANY variation) should be @ trip point ($V_{in}=V_{out}=V_{dd}/2=600\text{mV}$)
 - Size the transistors to set the trip point to 600mV
 - Maximize gain (and centralize ALL kinds of variation) @ trip point
 - Consider process corners

D2S CONVERTERS 1/3

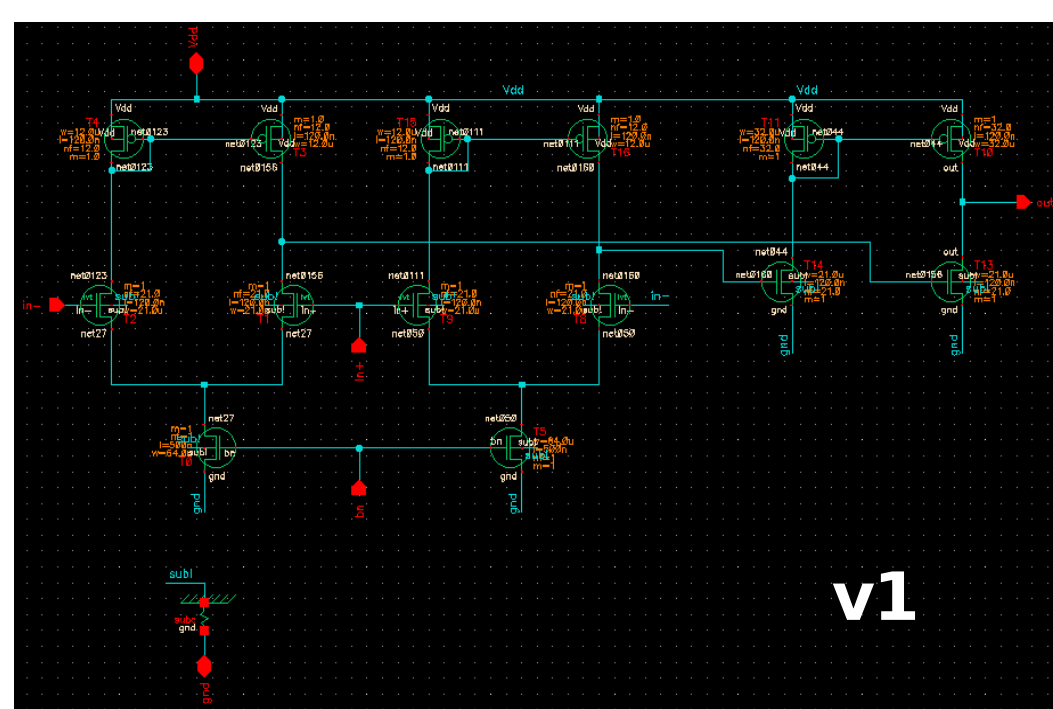
First trial was failure :

Duty cycle \neq 50%,
so I tried different
architectures before
coming back to v1.

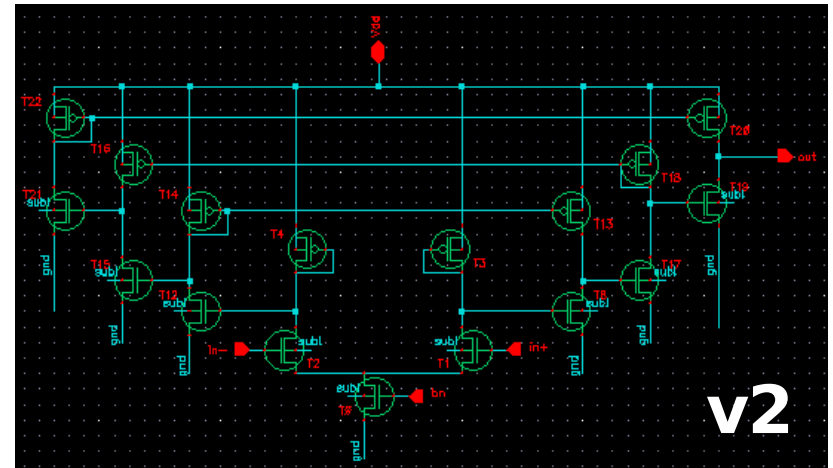


Phase_0
Phase_90
XOR out
%2 out

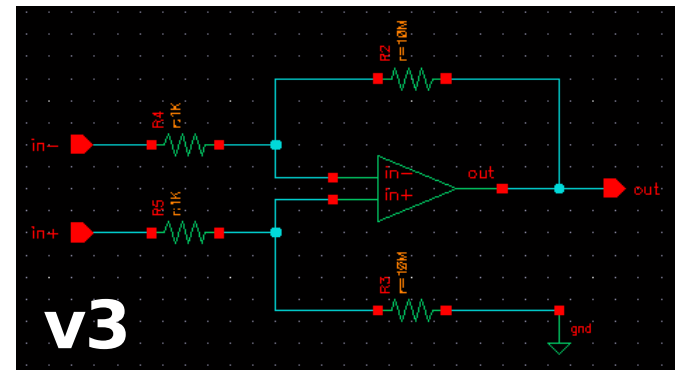
Ö.Ç.



v1



v2



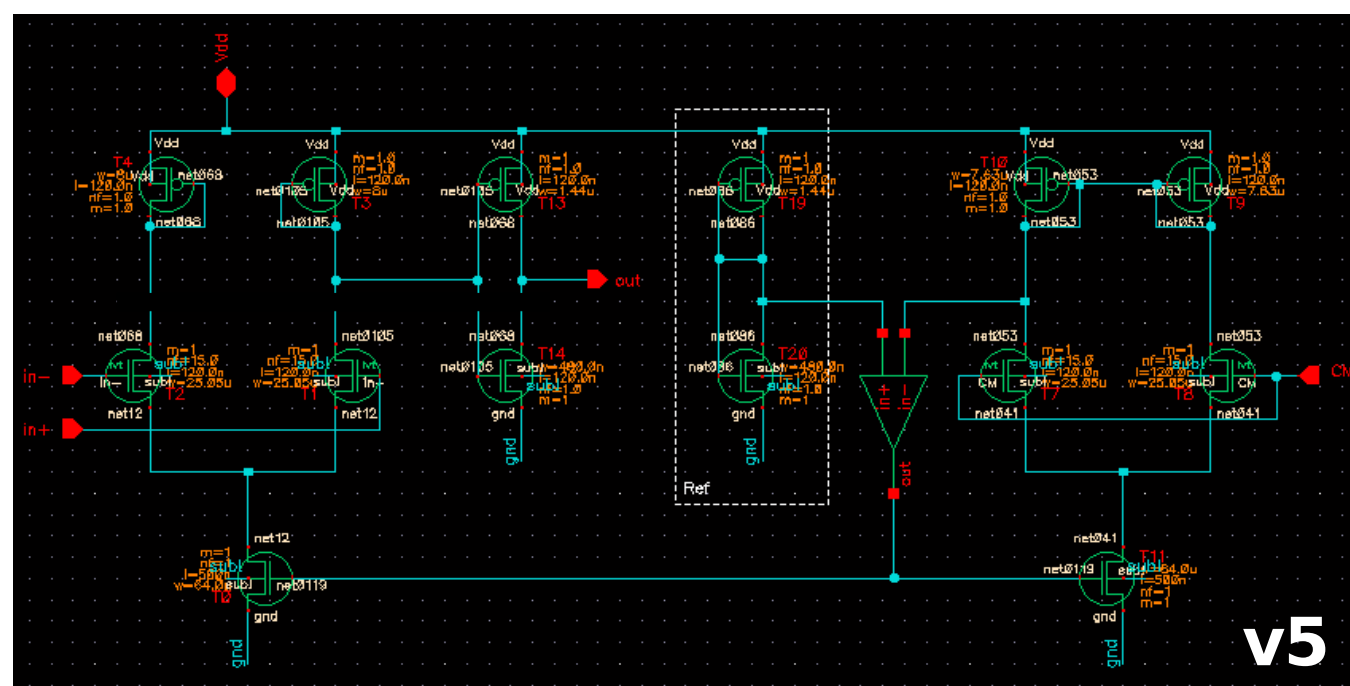
v3

D2S CONVERTERS 3/3

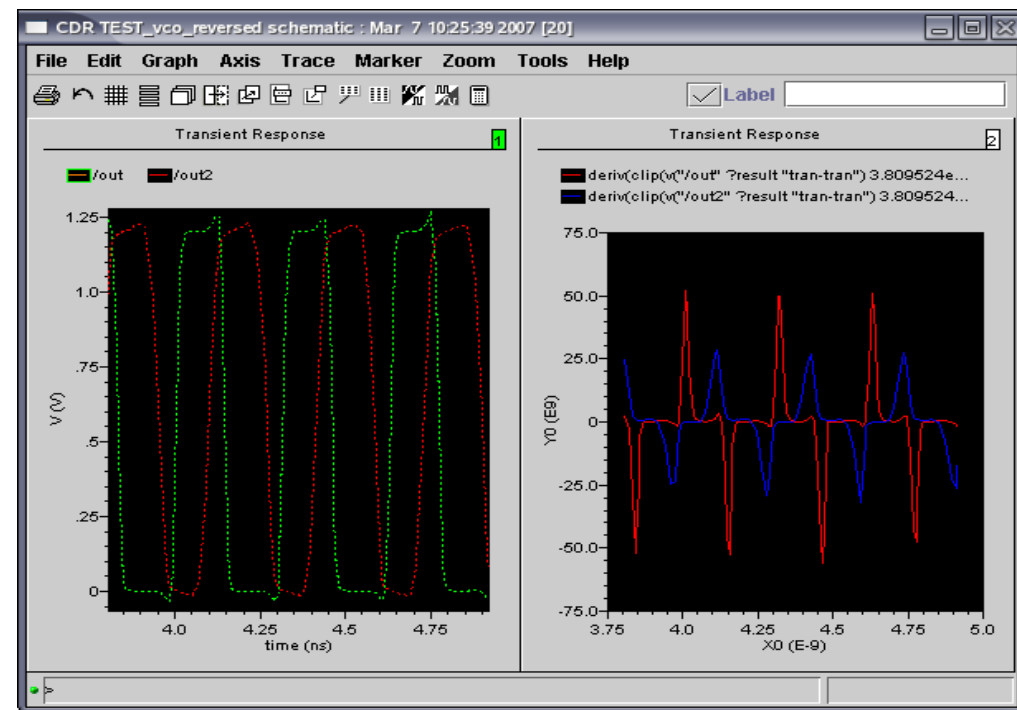
Duty cycle was better but rise time was not.

Both the approaches could exist in one implementation(?)

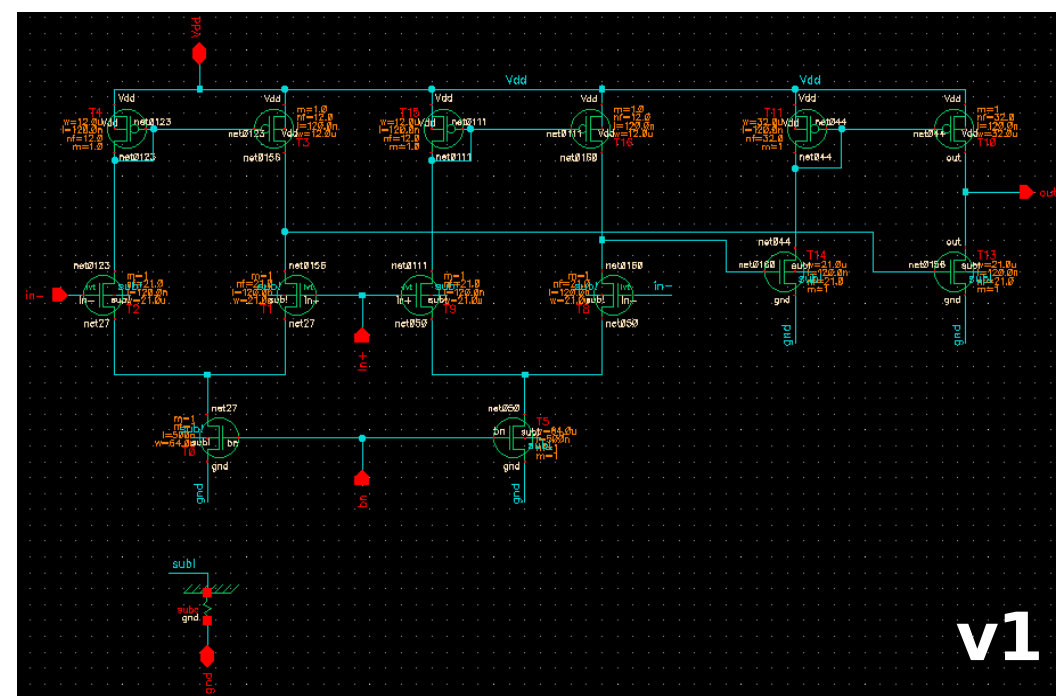
I stayed with **v1**, anyway.



v5



D2S Outputs & their derivatives



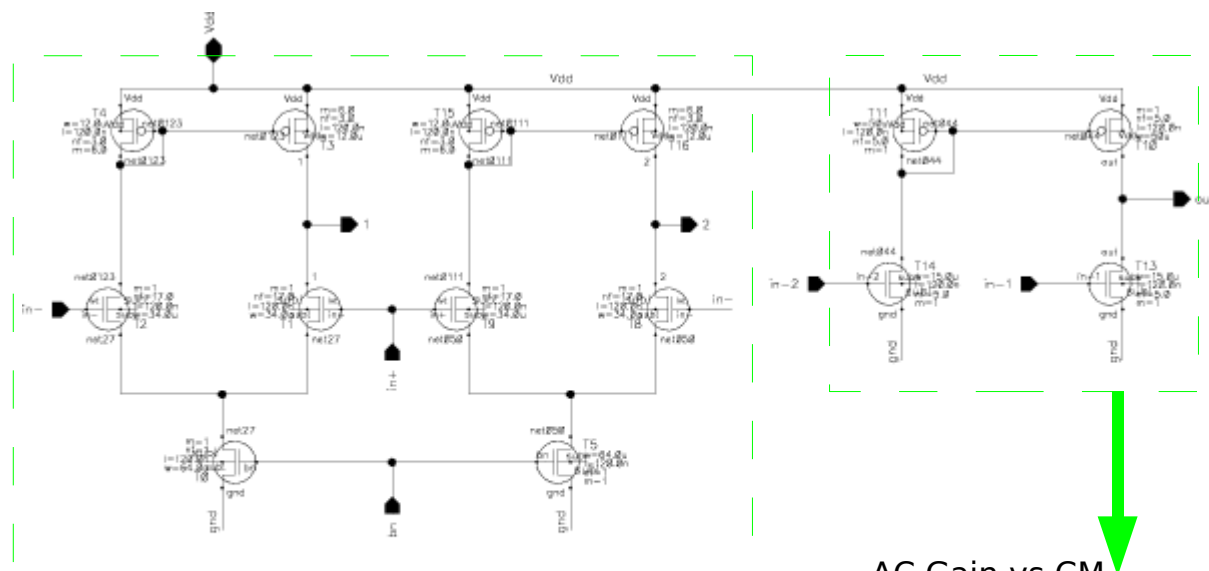
v1

Ö.Ç.

High resolution pictures; zoom in.

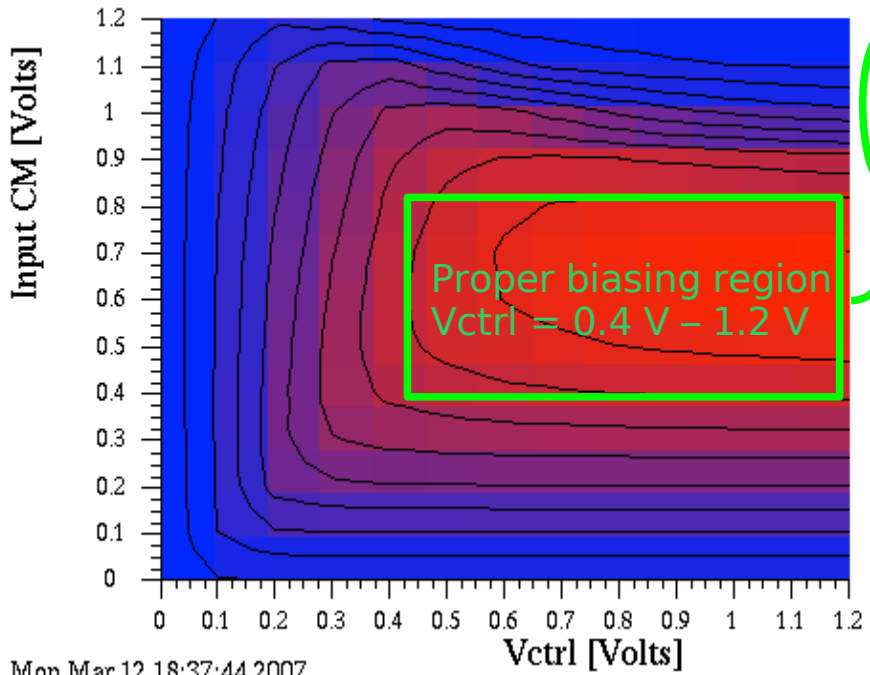
D2S FINAL VERSION

AC Gain of the whole D2S as a function of Input Common Mode and V_{ctrl} . (Note that differential circuits have a dependency on the input common mode, i.e. vertical change in color)

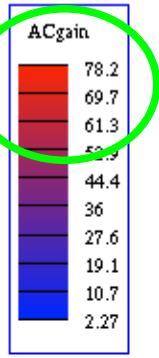


Worksheet

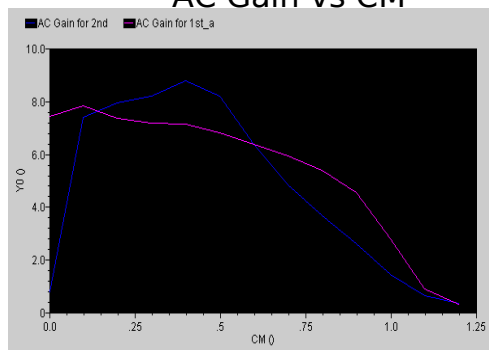
D2S Biasing



Mon Mar 12 18:37:44 2007

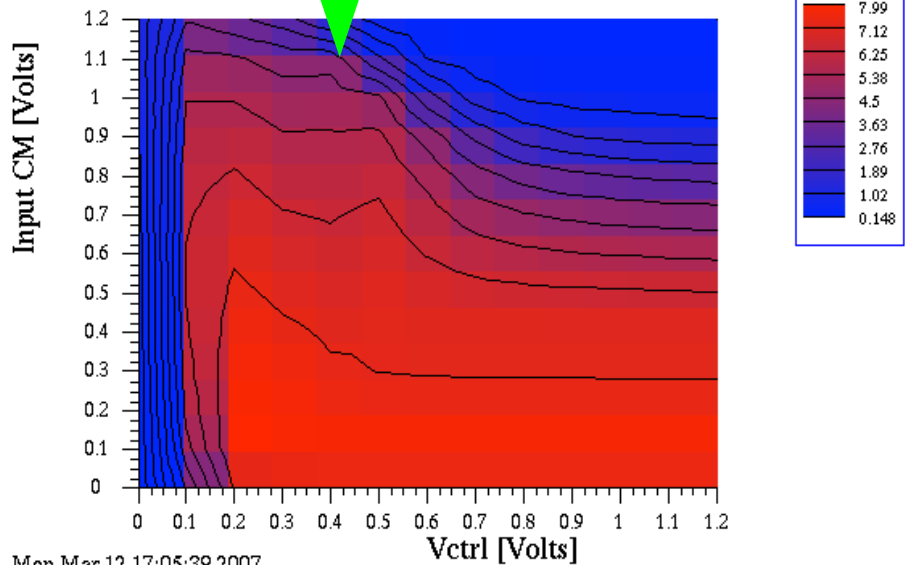


AC Gain vs CM



Worksheet

D2S - 1st Stage Biasing



Mon Mar 12 17:05:39 2007

Ö.Ç.

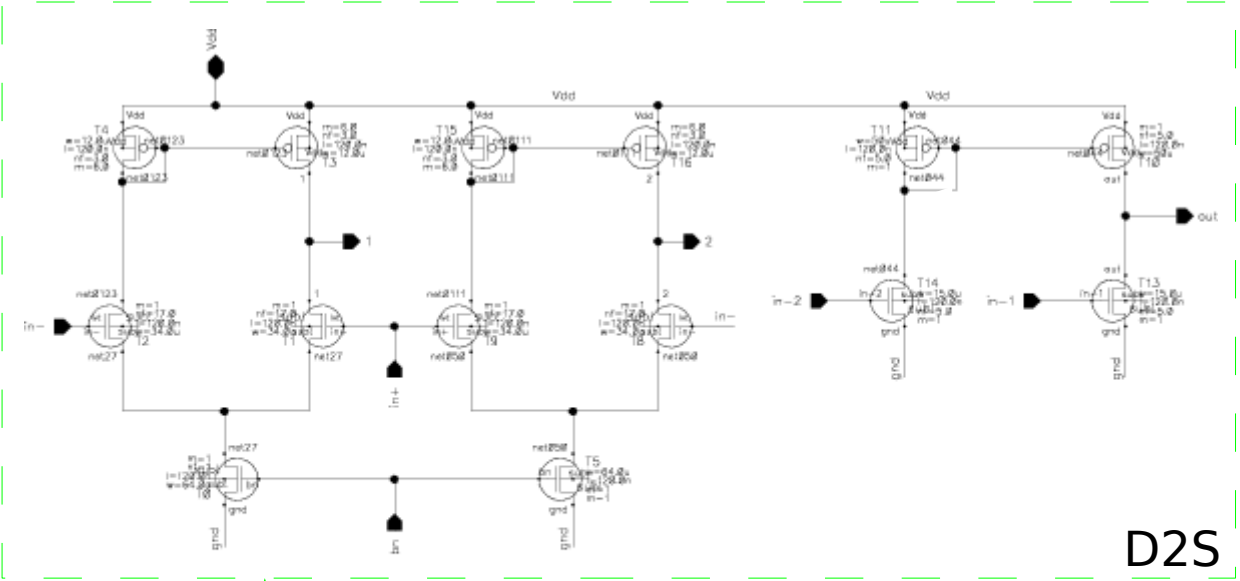
High resolution pictures; zoom in.

VCO+D2S BEHAVIOUR

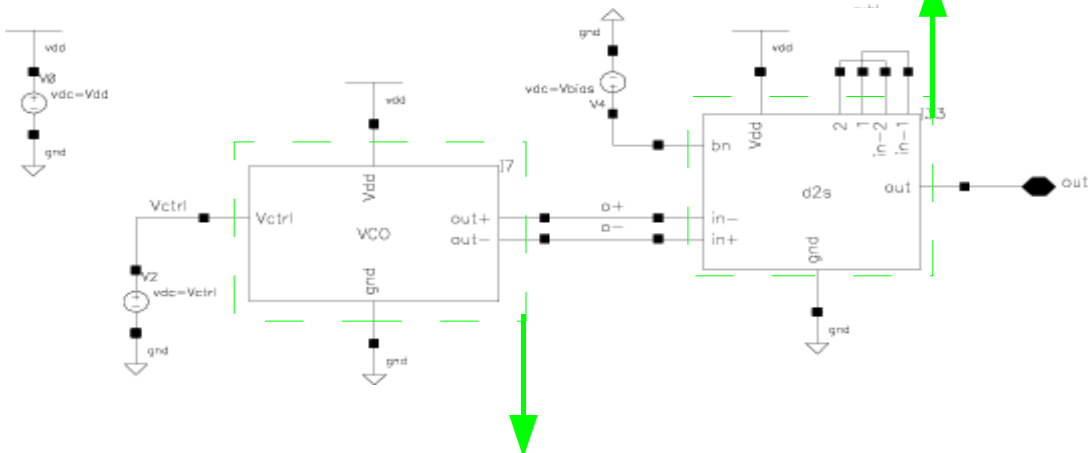
Putting things all together

VCO+D2S, THE CIRCUIT

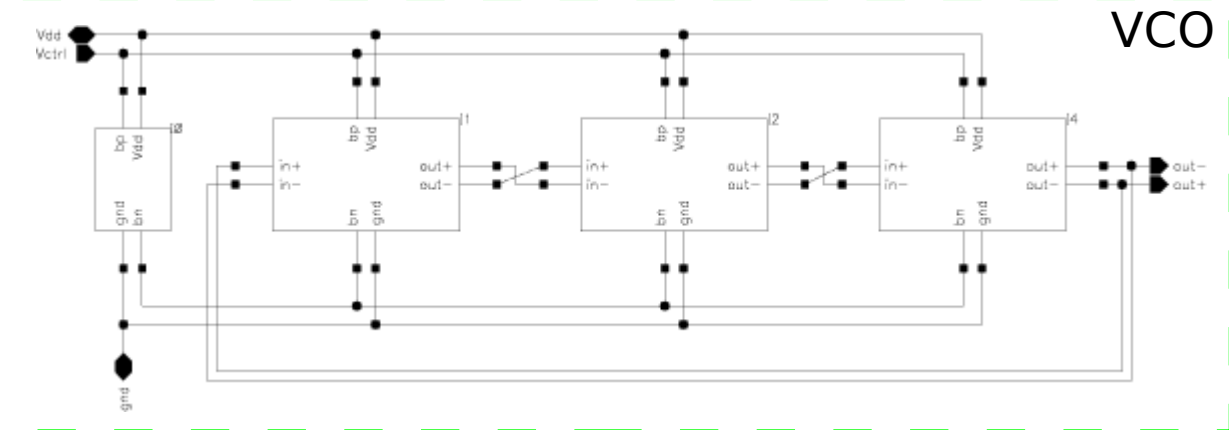
We will use this circuit to **extract behavioral parameters** to be imported by **verilogHDL** model.



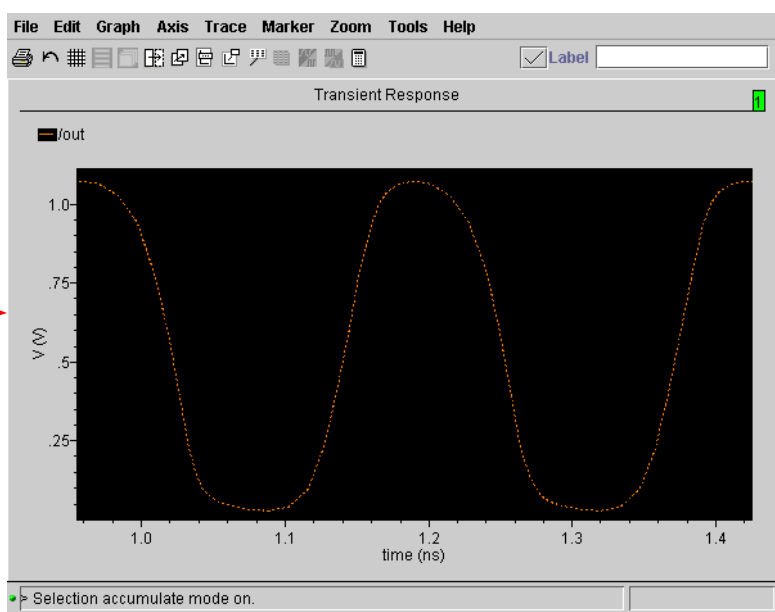
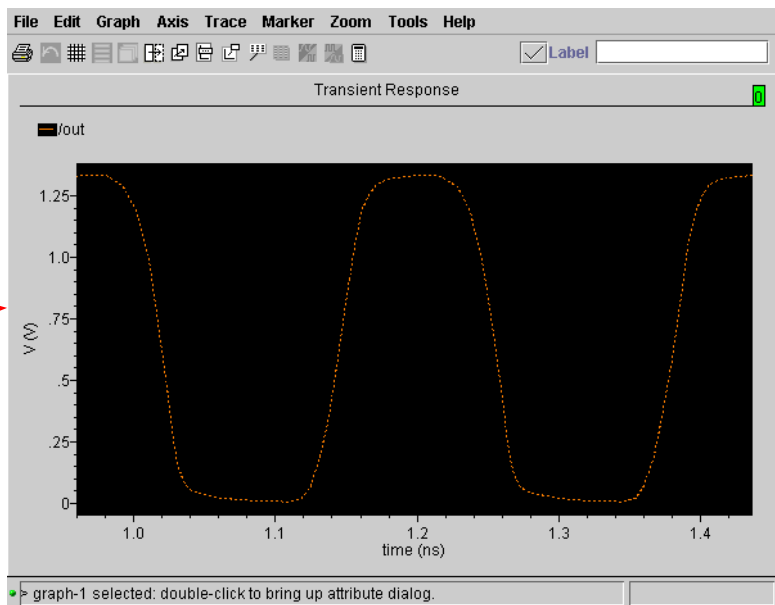
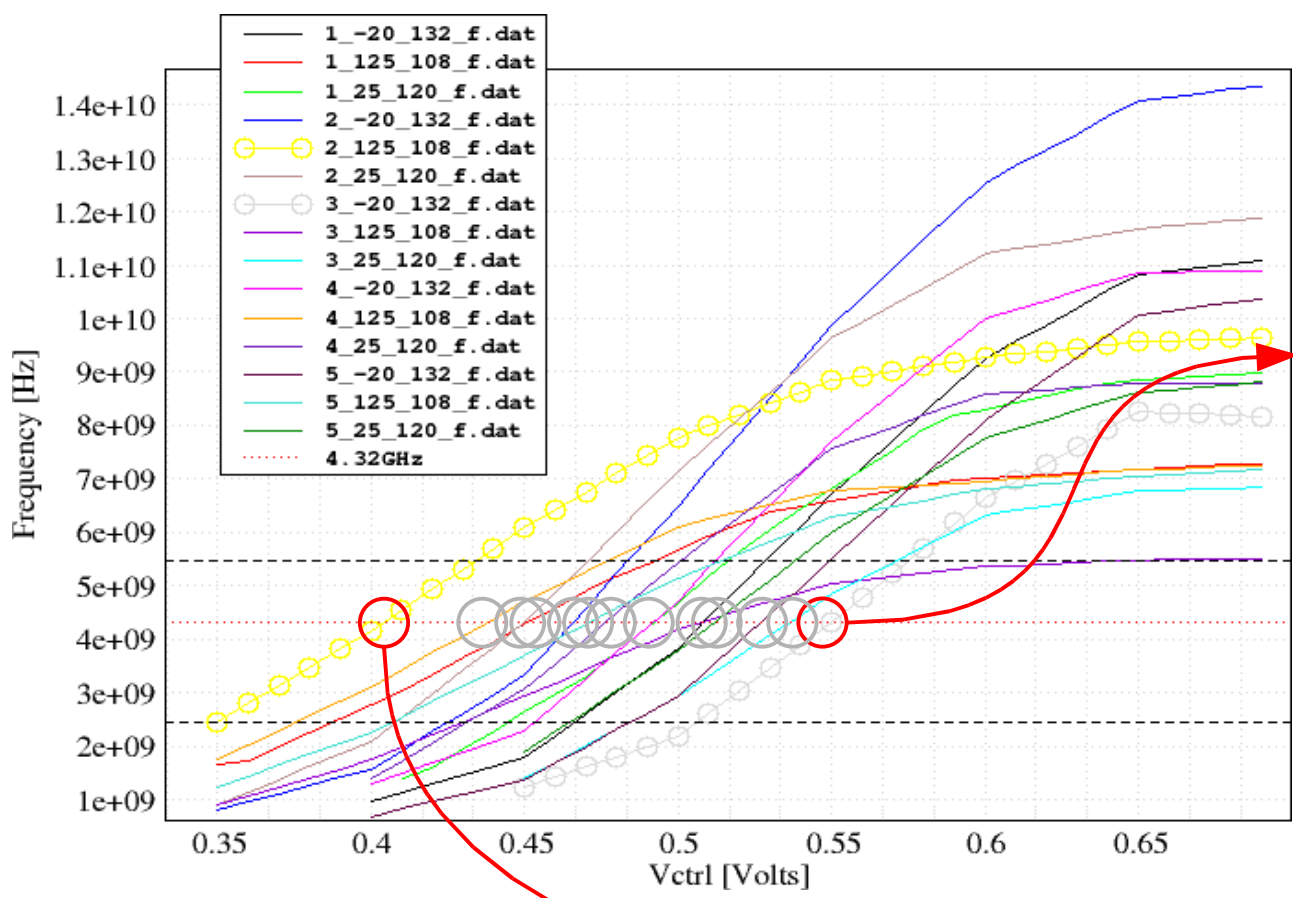
D2S



The VCOs are intended to be the **“plant”** of a **control loop** (e.g. Charge Pump Phase Locked Loop CP-PLL).



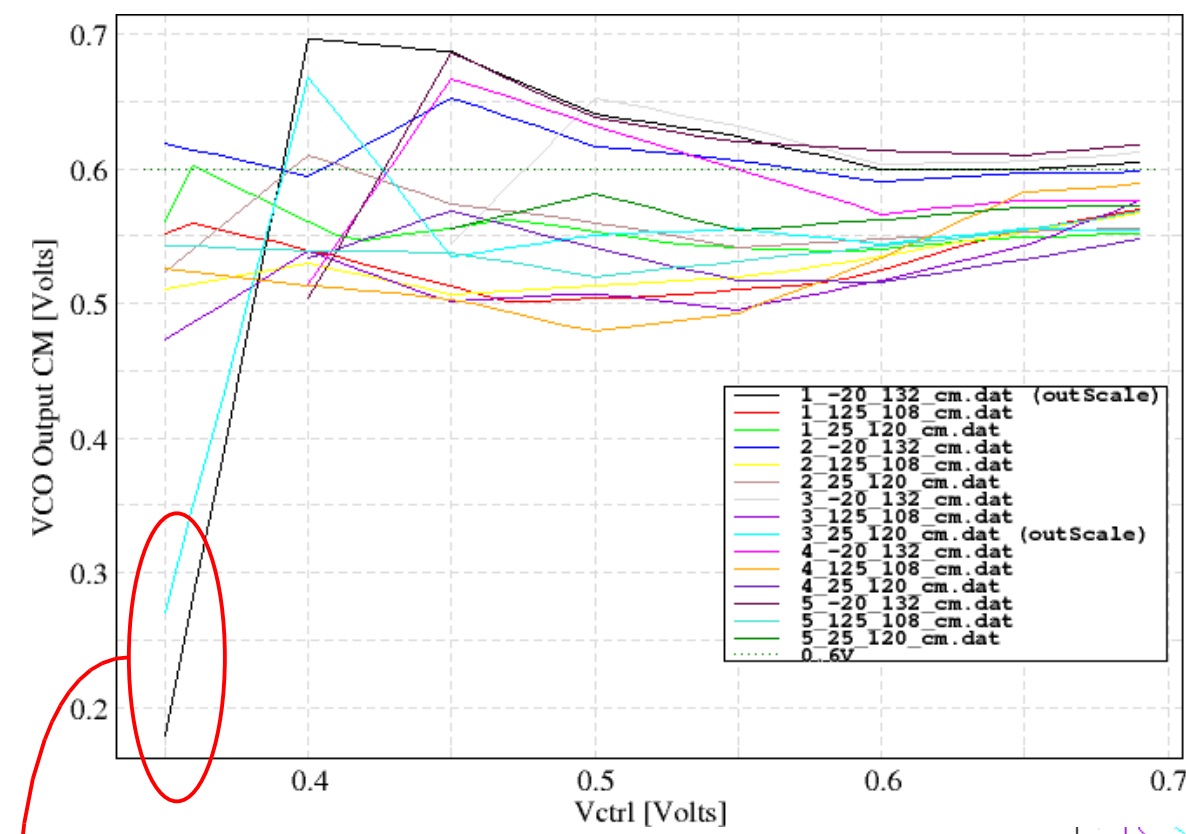
VCO



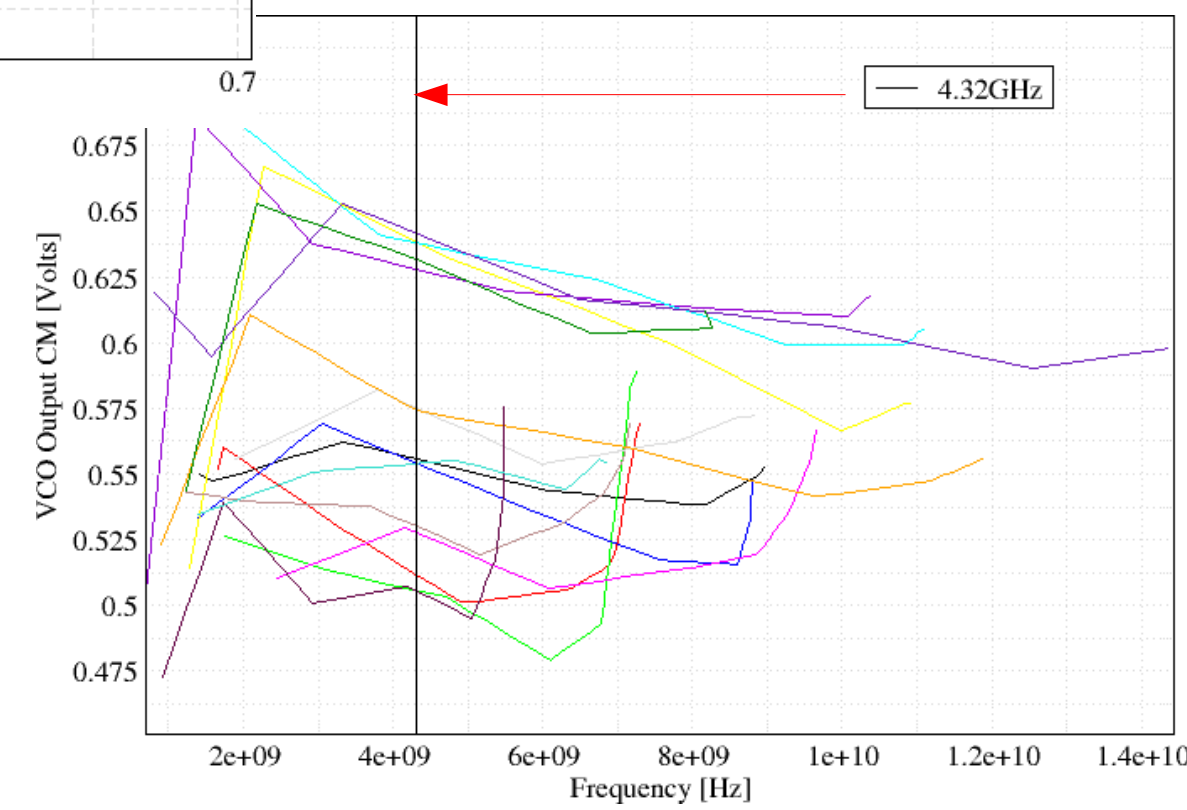
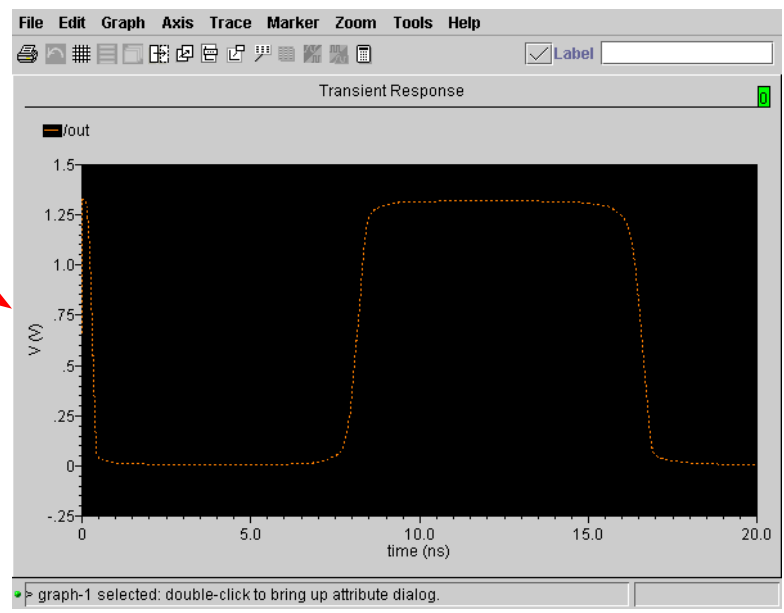
- **Wave forms** at **4.32GHz** for the two "extremes"
- See the **next page** for all the other waveforms @ **4.32GHz** (marked with gray circles)

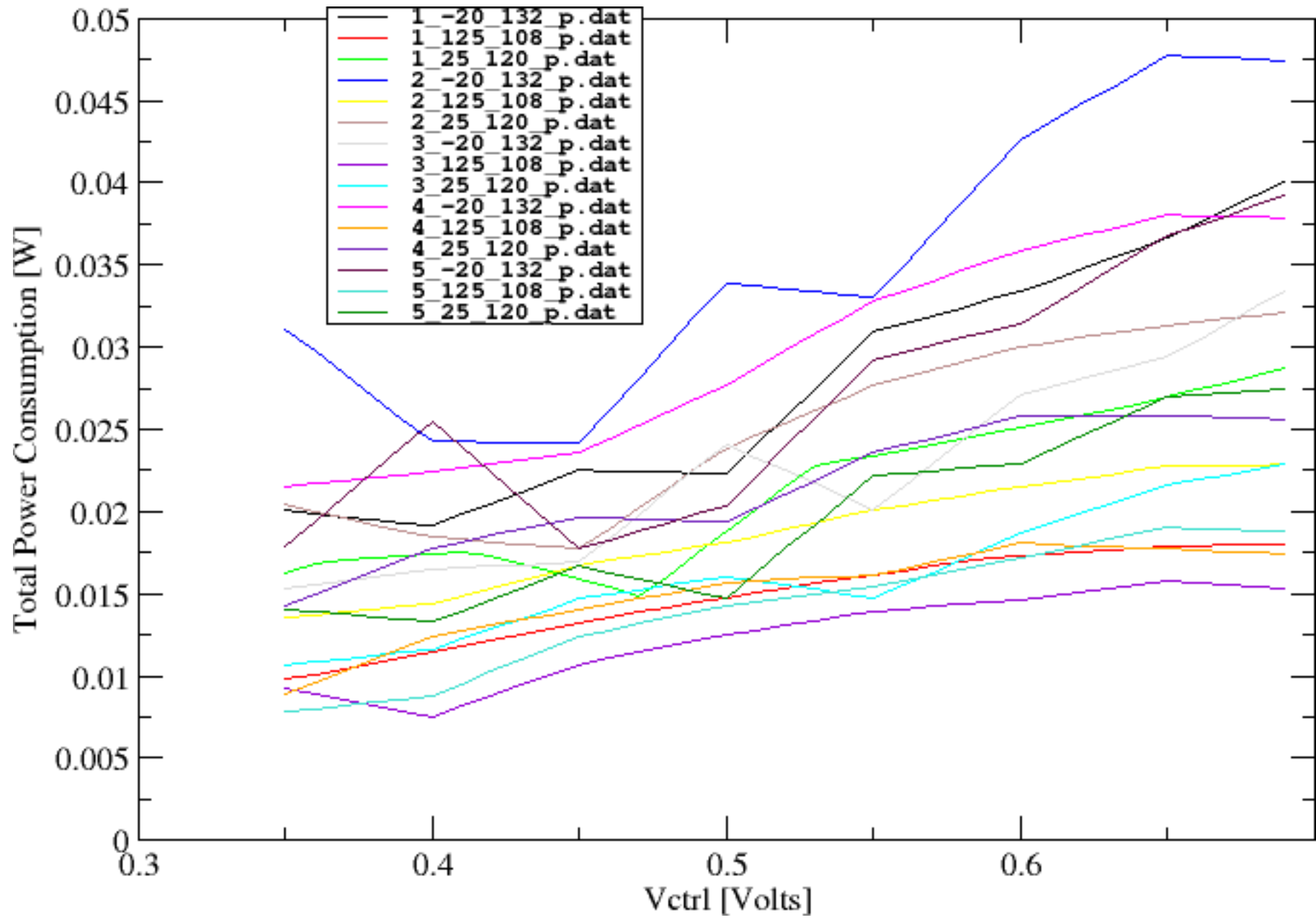
Ö.Ç.

VCO+D2S "ALL" THE CORNERS As specified at the GBT meeting on 14-03-2007 @ CERN



- Transient simulations have been performed for **2ns** & with steps of **Vctrl+=50mV**
- Frequencies lower than **1/(2ns)** can not be seen; this is the reason for the two un-expected points (labeled as outScale on the legend)
- See the wave form for a transient simulation of **20ns**
- **475mV < CM < 700mV** within the tuning range





Ö.Ç.

Intentionally left blank.

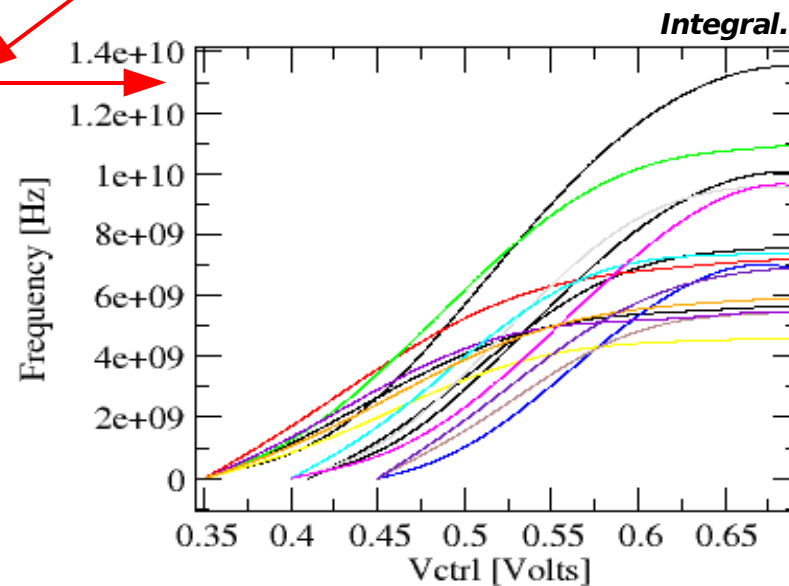
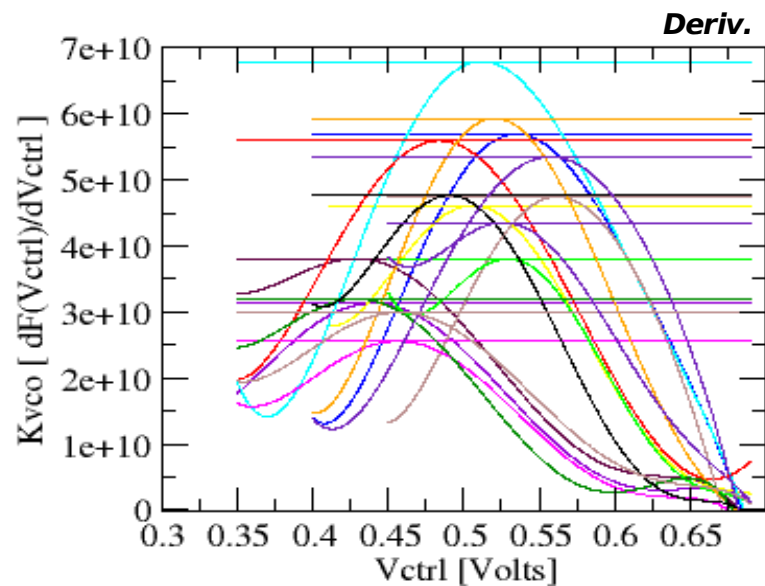
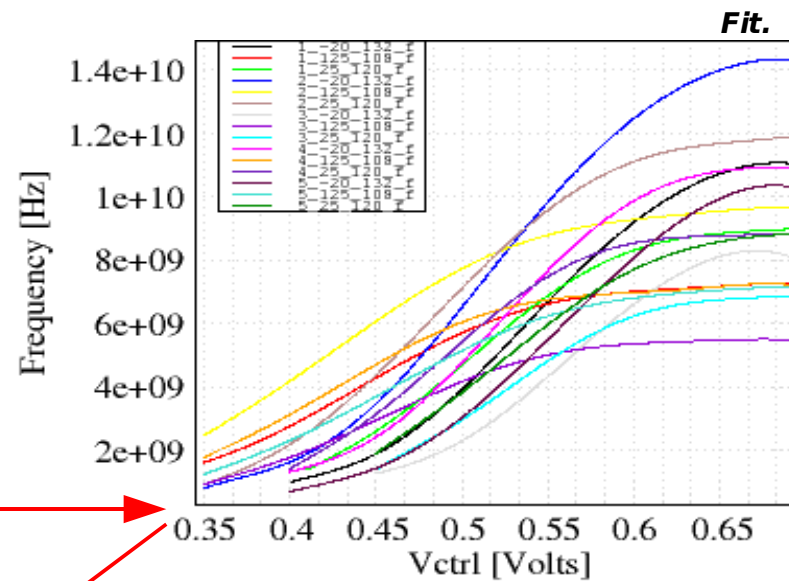
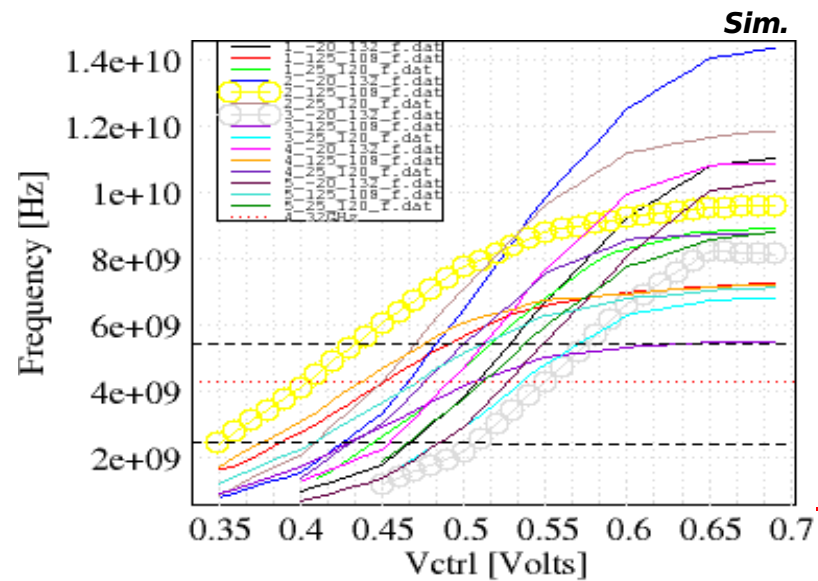
Ö.Ç.

POST-LAYOUT SIMULATION

Intentionally left blank.

Ö.Ç.

PARAMETER EXTRACTION & VerilogHDL MODEL @ Schematic Level



```

Corner_T_Vdd_What ->
5_25_120_f

Fitting with formula: y
= a0 + a1 * x + a2 *
x^2 + a3 * x^3 + a4 *
x^4 + a5 * x^5 + a6 *
x^6 + a7 * x^7 + a8 *
x^8 + a9 * x^9
Initial guesses:
a0 = -3.12334e+11
a1 = 3.46581e+12
a2 = -1.44348e+13
a3 = 2.46106e+13
a4 = 1
a5 = -5.74841e+13
a6 = 7.31251e+13
a7 = -2.93451e+13
a8 = 1
a9 = 1
Tolerance = 0.01
Relative error in the
sum of squares is at
most tol.
Computed values:
a0 = -3.39792e+12
a1 = 3.18654e+13
a2 = -1.1307e+14
a3 = 1.65853e+14
a4 = 1
a5 = -2.99961e+14
a6 = 3.42665e+14
a7 = -1.24852e+14
a8 = 1
a9 = 1

Chi-square: 1.69709e+16
Correlation
coefficient: 0.999935
RMS per cent error:
0.00381902
Theil U coefficient:
0.00395927

etc. x15
    
```

Simulated results are **approximated by 9th order fit curves** and mathematical **phrases were obtained** to be used within the VerilogHDL model.

Ö.Ç.

High resolution pictures; zoom in.

```

File Edit Search Preferences Shell Macro Windows Help
vco.v TEST_vco.v probe_vco.v
1 /*
2 File      : vco.v
3 Creator   : Ozgur Cobanoglu
4 Institute : Univ. & INFN of Turin, VLSI Group
5 Versions  : v0, 22-03-2007, OC, initial version
6           : v1, 26-03-2007, OC, probe is added
7
8 Information : This is a Voltage Controlled Oscillator module that
9               corresponds to the actual transistor level schematic.
10              Frequency as a function of control voltage, the control
11              curve of the VCO, is parametrized and the parameters
12              for all 15 corners have been included in the model.
13              The following table lists the possible corners settable :
14              Corner Meaning (C_T_V)
15              0      1_m20_132
16              1      1_125_108
17              2      1_25_120
18              3      2_m20_132
19              4      2_125_108
20              5      2_25_120
21              6      3_m20_132
22              7      3_125_108
23              8      3_25_120
24              9      4_m20_132
25              10     4_125_108
26              11     4_25_120
27              12     5_m20_132
28              13     5_125_108
29              14     5_25_120
30              Meaning has the format as the following :
31              C_T_V -> C = "fixed_cor_sw" parameter in the file "design.scs"
32                     T = temperature in Celcius (m stands for minus)
33                     V = value of Vdd potential - 108 is 1.08V
34                               120 is 1.20V
35                               132 is 1.32V
36
37 Warnings   : 1 - Parameters extracted are schematic level but not post-layout.
38              2 - Duty cycle variation due to different corners is also included
39                  but the duty cycle error is the one @ 4.32GHz. This value is
40                  used for all the tuning range within a specific corner, since the
41                  vco is expected to stay locked @ 4.32GHz during the operation.
42                  That is, it changes with the corner but not with the frequency.
43              3 - Random jitter is also included but the parameter does not depend
44                  on any simulation; I set it to -5ps<randomJitter<5ps arbitrarily.
45
46 */
47
48 `timescale 1ps / 1fs

```

All the corners, random jitter and duty cycle errors for all the corners @ 4.32GHz are included in the model

```

File Edit Search Preferences Shell Macro Windows Help
vco.v TEST_vco.v probe_vco.v data_probe_vco.dat read_data_probe_vco.C
141 always begin
142     riseJitter = randomJitter*($random/2147483647.0); // randomJitter is in units of ps
143     fallJitter = randomJitter*($random/2147483647.0); // RAND_MAX=2147483647.0
144     if (VCOout) # (500.0/freq+dutyCycleError/2.0+fallJitter) VCOout = ~VCOout;
145     if (~VCOout) # (500.0/freq-dutyCycleError/2.0+riseJitter) VCOout = ~VCOout;
146 end
147
148 always @(Corner or Vctrl) begin
149     vctrl = Vctrl/1000.0; // Vctrl is provided in mV and is converted to V internally
150     if (Corner < 0 || Corner > 14) begin
151         // See the module header; these are the corners defined at the GBT meeting
152         $display("\nERROR : Parameter covers the following range : 0 < Corner < 14");
153         $display("          : But it is %0d\n", Corner);
154         $finish;
155     end
156     if (Corner == 0) begin
157         if (vctrl < range4corner0_first || vctrl > range4corner0_last) begin
158             // Parameter extraction has been performed for a range but not between 0 to Vdd
159             // This is the proper operation range of the VCO
160             $display("\nERROR : Parameter covers the following range : %0f mV < Vctrl < %0f mV",
161                 range4corner0_first, range4corner0_last);
162             $display("          : for Corner=%0d, but it is %0d\n", Corner, Vctrl);
163             $finish;
164         end
165         a0 = -1.64755e+12;
166         a1 = 1.63633e+13;
167         a2 = -6.09447e+13;
168         a3 = 9.30479e+13;
169         a4 = 1.0;
170         a5 = -1.78625e+14;
171         a6 = 2.08998e+14;
172         a7 = -7.78976e+13;
173         a8 = 1.0;
174         a9 = 1.0;
175         dutyCycleError=dutyCycleError0;
176     end
177     if (Corner == 1) begin
178         if (vctrl < range4corner1_first || vctrl > range4corner1_last) begin
179             // Parameter extraction has been performed for a range but not between 0 to Vdd
180             // This is the proper operation range of the VCO
181             $display("\nERROR : Parameter covers the following range : %0f mV < Vctrl < %0f mV",
182                 range4corner1_first, range4corner1_last);
183             $display("          : for Corner=%0d, but it is %0d\n", Corner, Vctrl);
184             $finish;
185         end
186         a0 = -1.96339e+11;
187         a1 = 2.3548e+12;
188         a2 = -1.05071e+13;

```

The output

Range checks
@ every new
Corner and/or
Vctrl

Behavior
parameters
for different
corners
@ every
change
in Corner
and/or Vctrl

```

File Edit Search Preferences Shell Macro Windows Help
451     a9 = 1.0;
452     dutyCycleError=dutyCycleError13;
453     end
454     if (Corner == 14) begin
455         if (vctrl < range4corner14_first || vctrl > range4corner14_last) begin
456             // Parameter extraction has been performed for a range but not between 0 to Vdd
457             // This is the proper operation range of the VCO
458             $display("\nERROR : Parameter covers the following range : %0f mV < Vctrl < %0f mV",
459                     range4corner14_first, range4corner14_last);
460             $display("          : for Corner=%0d, but it is %0d\n", Corner, Vctrl);
461             $finish;
462         end
463         a0 = -3.39792e+12;
464         a1 = 3.18654e+13;
465         a2 = -1.1307e+14 ;
466         a3 = 1.65853e+14;
467         a4 = 1.0;
468         a5 = -2.99961e+14;
469         a6 = 3.42665e+14;
470         a7 = -1.24852e+14;
471         a8 = 1.0;
472         a9 = 1.0;
473         dutyCycleError=dutyCycleError14;
474     end
475
476     // Fit polinomial for the control curve that produces the frequency
477     // at which the VCO oscillates; frequency error is almost zero.
478
479     freq = (a0 +
480            a1 * vctrl +
481            a2 * vctrl*vctrl +
482            a3 * vctrl*vctrl*vctrl +
483            a4 * vctrl*vctrl*vctrl*vctrl +
484            a5 * vctrl*vctrl*vctrl*vctrl*vctrl +
485            a6 * vctrl*vctrl*vctrl*vctrl*vctrl*vctrl +
486            a7 * vctrl*vctrl*vctrl*vctrl*vctrl*vctrl*vctrl +
487            a8 * vctrl*vctrl*vctrl*vctrl*vctrl*vctrl*vctrl*vctrl +
488            a9 * vctrl*vctrl*vctrl*vctrl*vctrl*vctrl*vctrl*vctrl*vctrl) / 1.0e9;
489 end
490
491 // This is the probe module collecting statistics related to VCO functioning,
492 // The output is always produced and visualized by a ROOT script having a similar
493 // name with the data file in the same directory
494
495 probe_vco probe(.fileHandle(data_probe_vco),
496                .reference(ClkRef),
497                .signal(VCOout),
498                .phaseErrorLimit(phaseErrorLimit));
499
500 endmodule

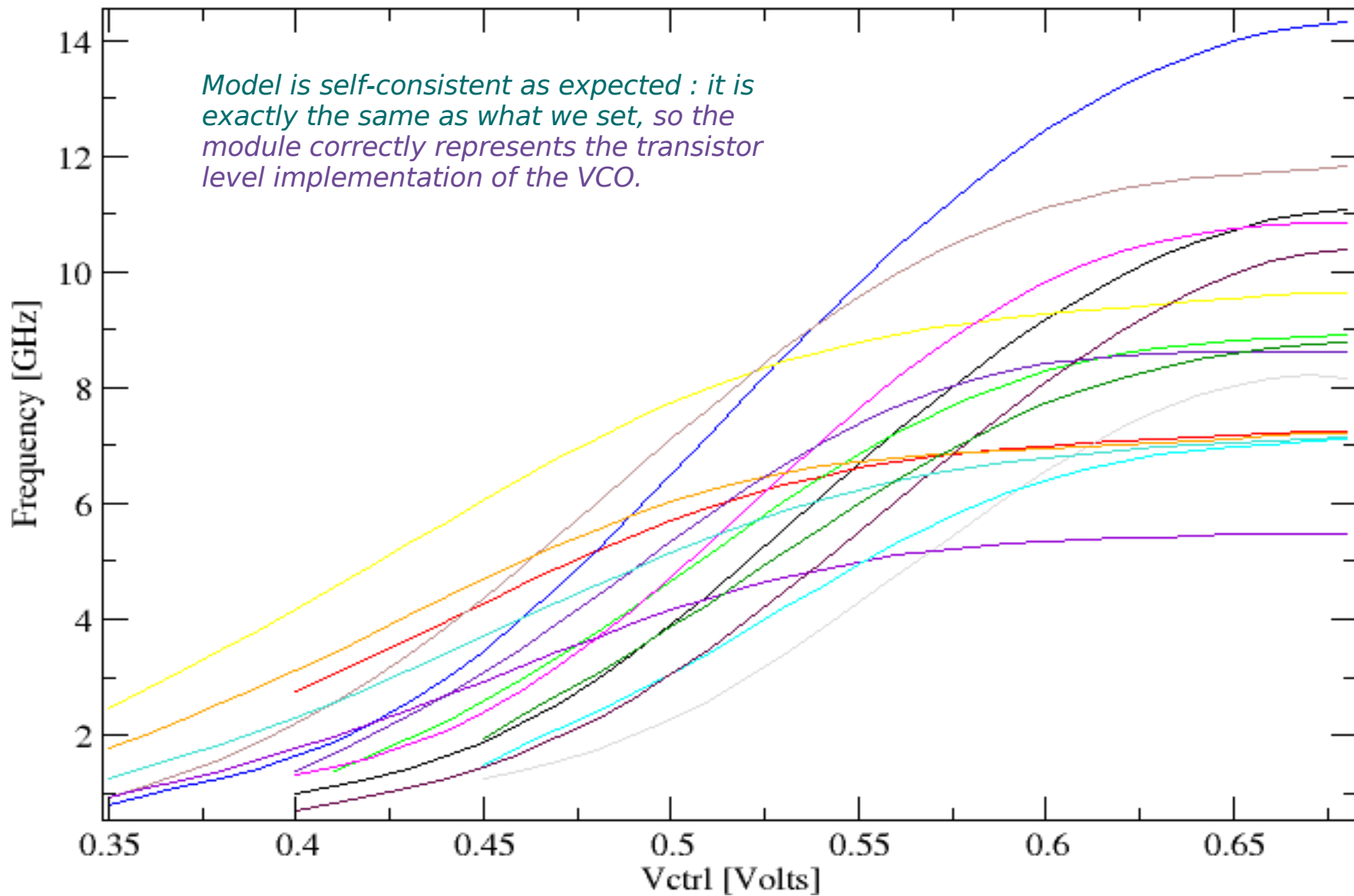
```

Frequency calculation that is, fit polynomial

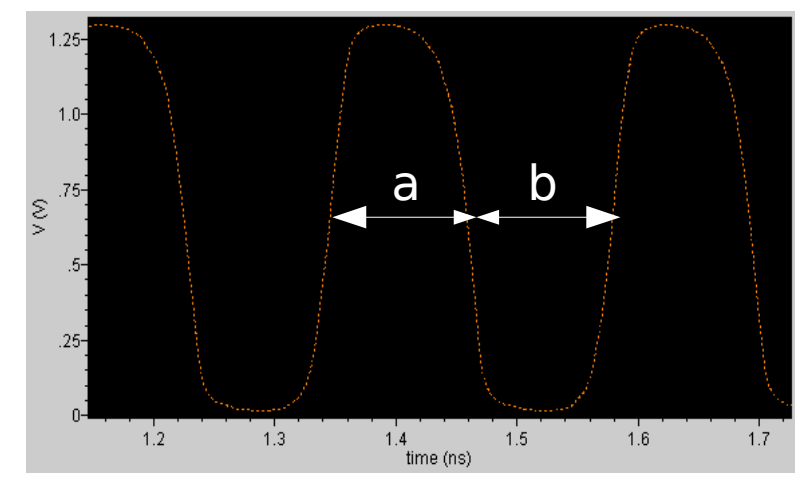
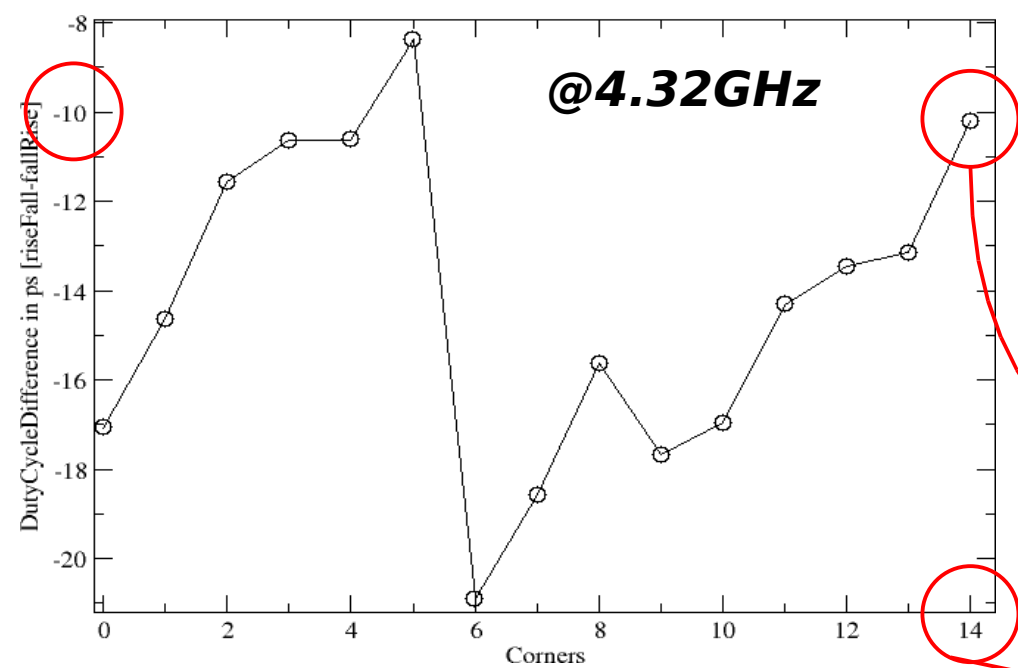
A probe for each module to collect statistics

VerilogHDL Output

File = "vco.v"

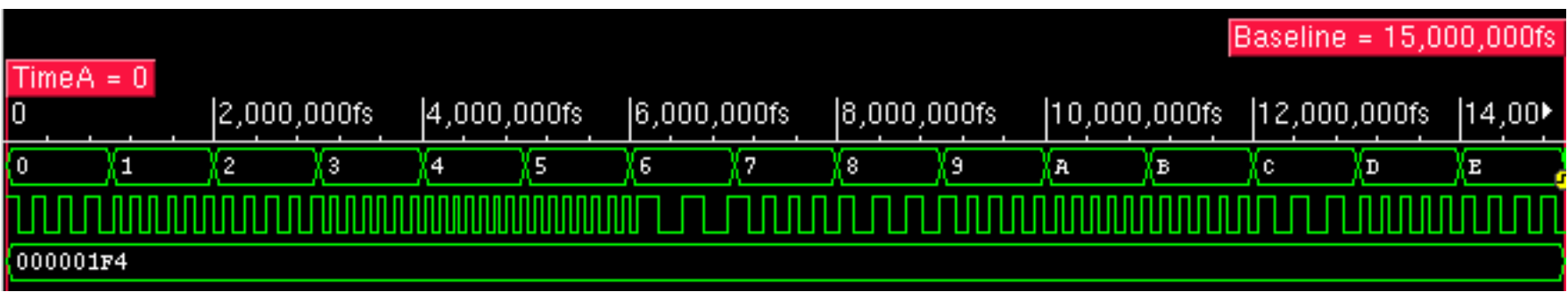
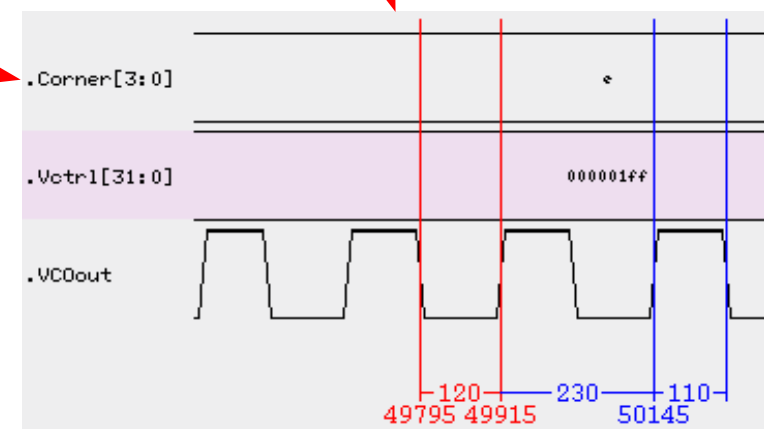


ACTUAL VerilogHDL CODE 5/5



DutyCycleDifference = a-b

Duty cycle error is included in VerilogHDL model compiled by two different compilers : **VerilogXL** and **iVerilog**.

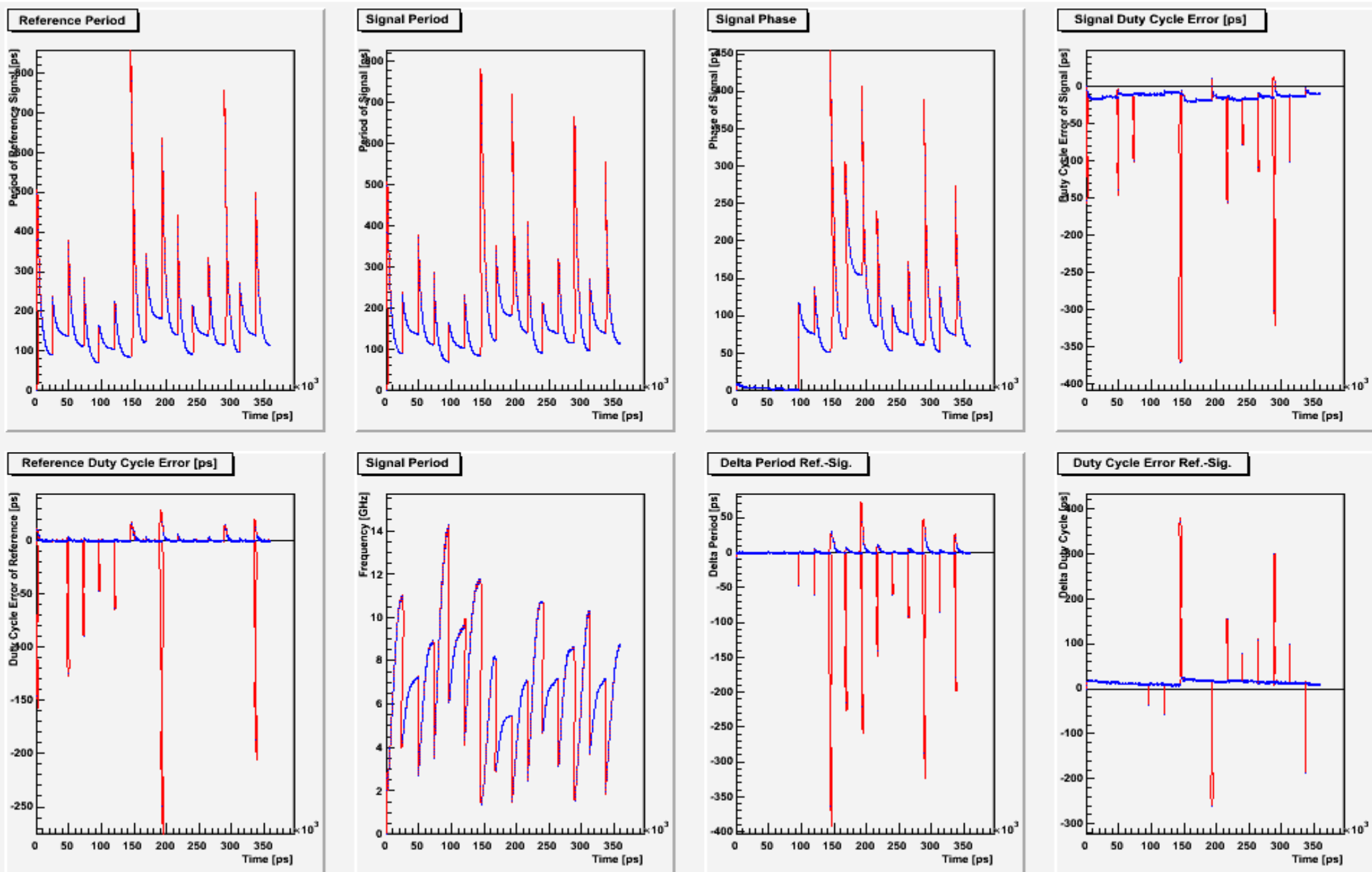


.Corner
.VCOout
.Vctr1

ACTUAL VerilogHDL CODE - PROBES

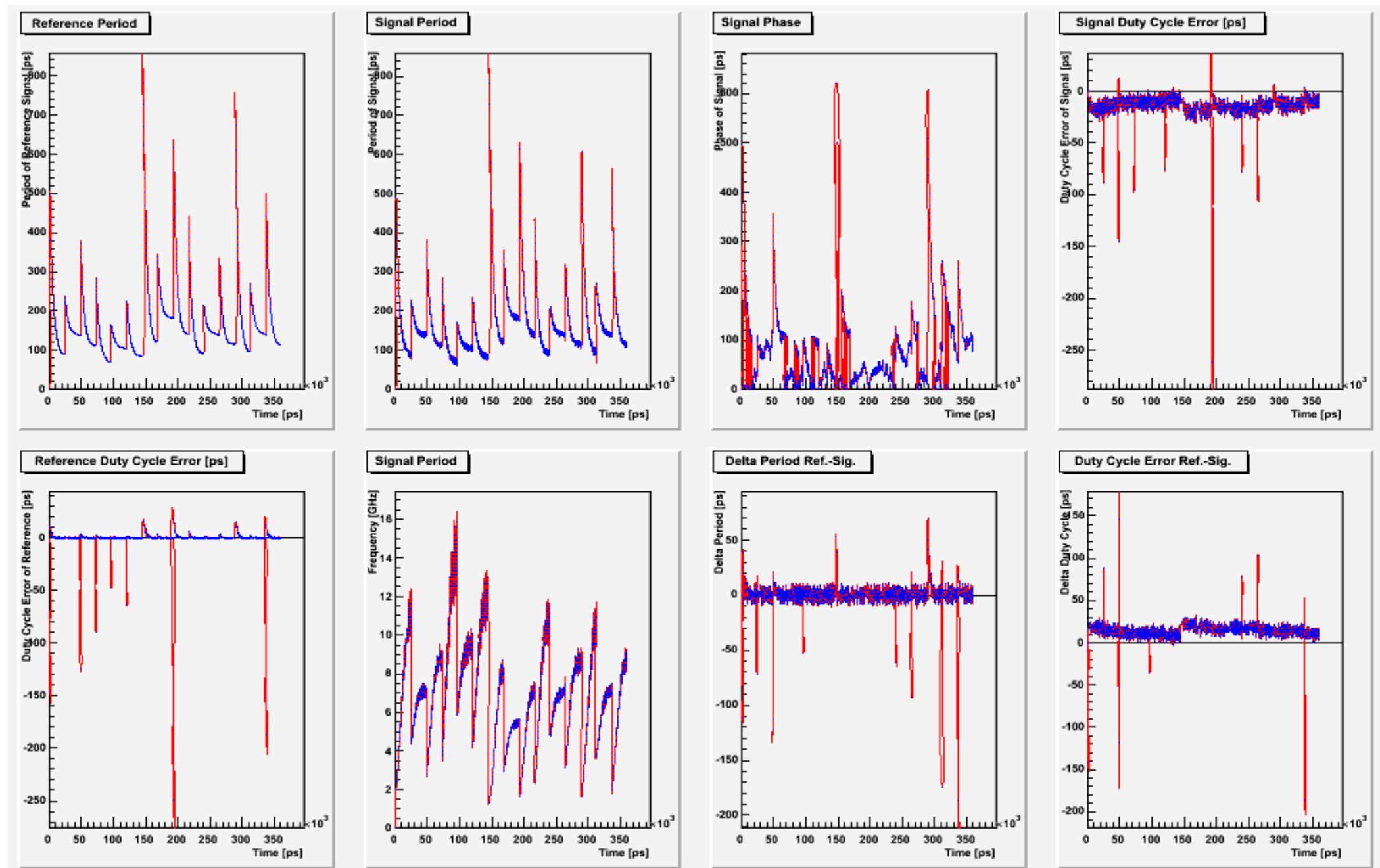
- Figure shows an almost full scan for VCO : all corners and almost all possible control voltages. Values correctly correspond to the ones set within the model.

- Each module has a **probe** which creates data files with interesting statistics
- Each probe data file is **read by a ROOT script**
- Because examining waveforms to debug a model is not a secure practice**

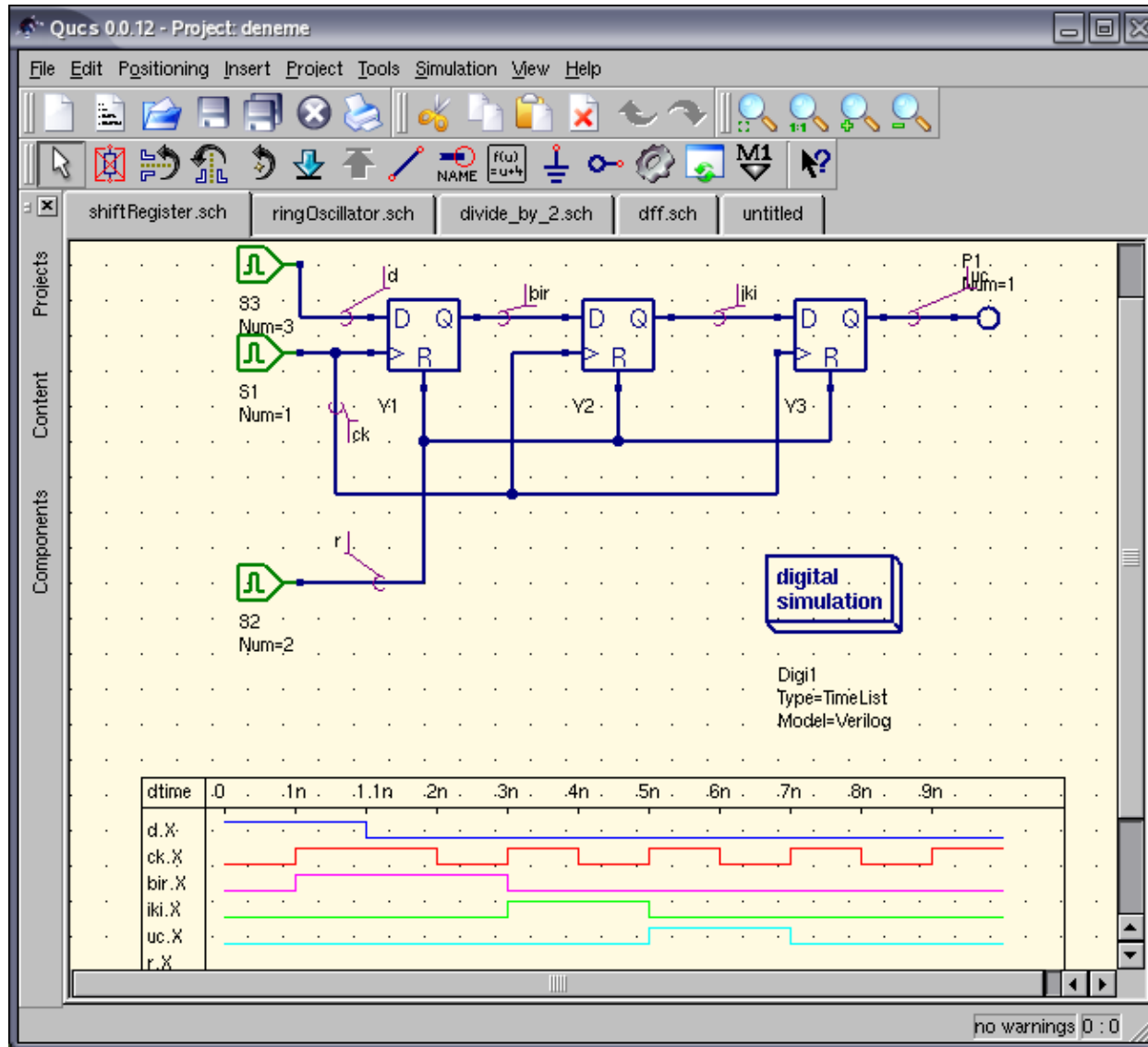


ACTUAL VerilogHDL CODE - PROBES

- Figure shows an almost full scan for VCO : same as the previous page but a **jitter of (+) or (-) 5ps** is introduced



OPEN TOOLS - Mixed Simulator QUCS, iVerilogHDL, FreeVHDL



- All-in-one tool
- Ability to synthesize HDL code
- Both VHDL and VerilogHDL support for digital simulations
- Very fast, simple models, multiple languages etc.

Ö.Ç.

```
d_flipflop.cpp - /home/oc/QUCS/qucs/qucs/components/
File Edit Search Preferences Shell Macro Windows Help

81 QString D_FlipFlop::verilogCode(int NumPorts)
82 {
83     QString t = "";
84     if(NumPorts <= 0) // no truth table simulation
85     if(strlen(Props.getFirst()->Value.latin1(), 0) != 0.0) {
86         t = Props.getFirst()->Value;
87         if(!Verilog_Time(t, Name))
88             return t; // time has not VerilogHDL format
89         t = "#" + t;
90     }
91
92     QString s = "";
93     QString q = Ports.at(2)->Connection->Name;
94     QString d = Ports.at(0)->Connection->Name;
95     QString r = Ports.at(3)->Connection->Name;
96     QString c = Ports.at(1)->Connection->Name;
97     QString v = "net_reg" + Name + q;
98
99     s = "\n // " + Name + " D-flipflop\n" +
100     " assign " + q + " = " + v + ";\n" +
101     " reg " + v + ";\n" +
102     " initial " + v + " = 1'b0;\n" +
103     " always @(posedge " + c + ") \n" +
104     " " + v + " <= " + t + " + d + ";\n" +
105     " always @ " + r + " \n" +
106     " if (" + r + ") \n" +
107     " assign " + v + " = 0;\n" +
108     " else \n" +
109     " deassign " + v + ";\n\n";
110
111     return s;
112 }
```

Verilog code generator C++ code

```
digi.v - /home/oc/qucs/
File Edit Search Preferences Shell Macro Windows Help

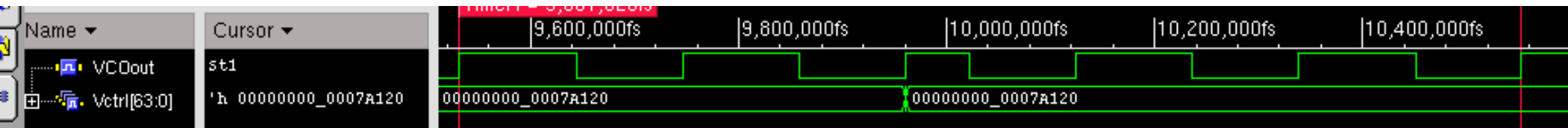
34 // Y1 D-flipflop
35 assign netQ = net_regY1netQ;
36 reg net_regY1netQ;
37 initial net_regY1netQ = 0;
38 always @(posedge netCK)
39     net_regY1netQ <= netQB;
40 always @netR
41     if (netR)
42         assign net_regY1netQ=0;
43     else
44         deassign net_regY1netQ;
```

Verilog code generated

High resolution pictures; zoom in.

PROBLEM – Vctrl Spikes

- So far, the VCO model was similar to a mathematical function like $F(Vctrl)$, that is, the output frequency is a function of input control voltage, that is, for every value change of Vctrl we would have a new value for the output frequency
- In real life, this definition is not enough to define a VCO practically. One must define “**how**” the input control voltage will be translated to output wave, that is, not the output frequency but the output phase.
- To arrive at the problem statement, consider the following VCO output :



- A very narrow (like 2fs, which is almost impossible in real life) spike of half a volt is applied to the Vctrl, thus the VCO “updates” the period variable within the module two times which are extremely close (2fs) to each other. Period becomes very different from the initial value for a very small amount of time then it returns back to its initial value. So the average control voltage is almost un-affected. Even though, this looks proper, it is not; because the next transition should “wait” this period long, thus an effective phase jump occurs. In real life however a 2fs spike can not change the output phase of a VCO this much.
- If the VCO is the “plant” of a control system like a PLL, then the verilog implementation given so far can not be used in case of a high feedback divide ratio. So an updated version that can handle this problem must be developed.
- One can add too **many delay cells sequentially** and apply the control voltage to each of the delay cells and **divide the output frequency by the number of delay cells** used. This technique can not really “solve” the problem but it significantly decreases the size of “phase jump”. Another technique could be calculating the **output phase instead of output frequency** and/or using “**analog**” clock signals instead of squares.