Behavioral Modeling of Delay-Locked Loops and its Application to Jitter Optimization in Ultra Wide-Band Impulse Radio Systems

E. Barajas[1], R. Cosculluela[1], D. Coutinho[1], D. Mateo[1], J. L. González[1], I. Cairò[2], S. Banda[2], M. Ikeda[3]
[1]Electronic Engineering Department, Universitat Politècnica de Catalunya, Barcelona, Spain
[2]Barcelona R&D Laboratory, EPSON EUROPE Electronics GmbH, Sant Cugat, Spain
[3]Communications Device R&D Department, SEIKO EPSON Corporation, Hirooka, Shiojiri-shi, Japan

Abstract

This paper presents a behavioral model of a delay-locked loop (DLL) used to generate the timing signals in an integrated ultra wide-band (UWB) impulse radio (IR) system. The requirements of these timing signals in the context of UWB-IR systems are reviewed. The behavioral model includes a modeling of the various noise sources in the DLL that produce output jitter. The model is used to find the optimum loop filter capacitor value that minimizes output jitter. The accuracy of the behavioral model is validated by comparing the system level simulation results with transistor level simulations of the whole DLL.

1. Introduction

Ultra Wide-Band (UWB) communication techniques have received increasing attention since United States Federal Communications Commission (FCC) adopted a "First Report and Order" [1]. Impulse Radio (IR) implementation of UWB systems has very interesting features such as low complexity, low power consumption, low cost, high data rate, and the ability of coexistence with other radio systems [2].

The usual UWB-IR transmitter consists of a pulse generator that is triggered regularly by a timing circuitry. Data is transmitted by modifying some parameter of the pulse (for example, its sign in BPSK modulation or its position in PPM modulation). The transmitted output waveform has a very low duty cycle since the sub-nanosecond pulses are sent every frame. For usual data rates the frame time is several nanoseconds long. A time hopping (TH) technique is commonly used to allow multiple users access and to avoid peaks in the spectrum of the UWB signal. A pseudorandom code locates each successive pulse in a different position along its frame. The UWB-IR receiver can be implemented in several ways, but we focus in this paper in a very low power implementation. One of the most efficient ones is a coherent receiver using a matched filter to detect the received pulse [2]. The matched filter receiver, whose architecture is depicted in Fig. 1, decides the received symbol after integrating the result of the multiplication of the received signal with a locally generated Template Waveform (TW). The received signal is processed in the analog domain to optimize power consumption [3,4]. In the ideal case, the TW shape is matched to the received pulse waveform to obtain the optimum receiver. However, the received pulse and template waveforms must be precisely synchronized before the multiplication. The template waveforms are generated from a trigger signal provided by the timing controller, as indicated in Fig. 1. Any timing misalignment between the arriving pulse and the locally generated template waveform reduces the energy recovered by the matched filter, thus degrading the system performance, as shown in Fig. 2. The figure results correspond to various pulse waveforms and various templates waveforms for system operating at a rate of $200 \cdot 10^6$ pulses/s with a transmitted pulse width of 49.50 ps [5]. From this figure, it is clear that timing errors in the trigger signal generation in the range of 10–15 ps can be tolerated but the reception is severely degraded if the timing error becomes larger.

The timing controller usually consists of a delay-locked loop and a multiplexer. The multiplexer selects one of the outputs of the voltage controlled delay line (VCDL) according to the time-hopping sequence. Therefore, the number of taps of the VCDL depends on the discrete number of positions during the frame time that the time-hopping sequence can select. The input reference clock

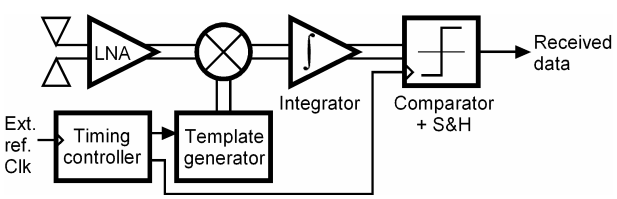

Fig. 1: Block diagram of an ultra wide-band impulse-radio receiver.

978-3-9810801-2-4/DATE07 © 2007 EDAA

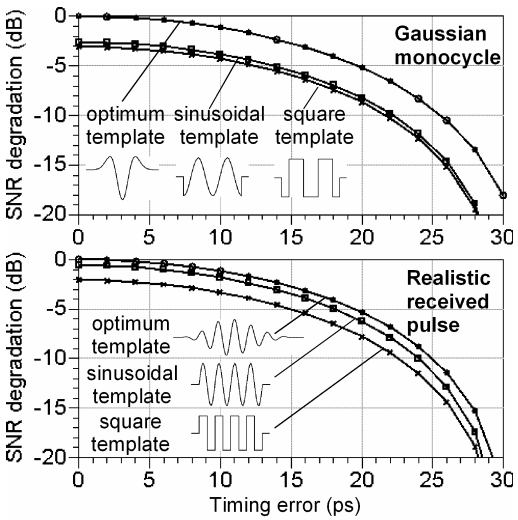

Fig. 2: SNR degradation vs. timing error for two different received pulses (from [5]).

period is equal to the frame time. Actually, since some guard time is required between consecutive frames to avoid inter pulse interference, the number of taps of the VCDL is slightly larger than the number of possible time-hopping values. Assuming that the receiver and the transmitter have synchronized frame start and time-hopping sequence, the only source of timing errors is the time jitter of the outputs of the DLL.

In this paper we present a design strategy to properly size the DLL blocks and optimize its output time jitter performance. The design strategy is based on the implementation of a behavioral model in VerilogA [6] of the DLL blocks that includes jitter generation. The behavioral models are derived from the transistor level schematics of each of the blocks. Transistor level simulation of the whole system is very time consuming. The behavioral block models are then combined in a system model of the whole DLL and the optimum loop filter that minimizes the output jitter is found by simulation. The paper is organized in the following way. Section 2 describes the sources of jitter in the DLL. Section 3 describes the VerilogA models for each block in the DLL. Section 4 presents the DLL simulation and optimization results and also a verification analysis that compares behavioral system level results with transistor system level results for a particular loop filter capacitor value. The paper is concluded in section 5.

2. Sources of jitter in a DLL

The block diagram of a conventional DLL for the generation of timing signals for UWB impulse radio systems is shown in Fig. 3a. The reference clock *ref_clk* is fed to the first stage of the VCDL. Each stage provides a delayed version of this signal. The VCDL last stage output *out_clk* is compared with the *ref_clk* signal in the phase and frequency detector (PFD). This block provides two pulsed outputs that indicate the phase difference and sign between its two inputs. These signals are used to increase or decrease the value of the control voltage (*Vc*) of the VCDL. The delay of each cell in the VCDL depends on *Vc*. The loop acts as a feedback system, compensating any phase difference between *out_clk* and *ref_clk*. Therefore, once the loop is in the locked state, the two signals have exactly the same frequency and are aligned in phase, being *out_clk* exactly a one period delayed version of *ref_clk*. Since the output of the PFD is a pulsed signal, the useful information is contained in its average value. The charge pump (CP) and the loop filter (in most of the cases it is just a capacitor to ground) are used to obtain this average value. In the particular circuit-level implementation used in our work, the VCDL consists of supply-regulated inverters, and therefore a buffer is required at the output of the loop filter to provide enough current for the control voltage signal (*Vc*). This is a brief description of the ideal operation of the DLL. More details can be found for example in [7].

In real circuits, however, thermal noise and other sources of electronic noise modify the ideal operation described above. This non ideal behavior results in time jitter in the timing signals generated by the DLL. The contributions of the various blocks to the output jitter happen in different signal domains, as shown in Fig. 3.b. For example, the *ref_clk* signal has an associated phase error θ_{in} superimposed to the ideal input phase ϕ_{in} every cycle. The output of the CP is in the voltage domain, and

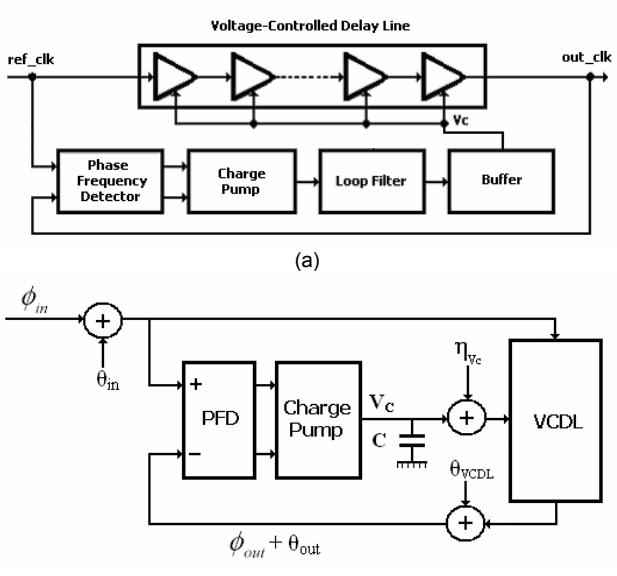

Fig. 3: (a) DLL block diagram, and (b) linear model including noise sources.

therefore the error is a voltage noise superimposed to V_C, which is due to noise sources in the PFD and noise sources in the CP itself. The capacitor filters the noise contribution of the PFD and the CP, and therefore this filtered version of the voltage noise η_{Vc} can be added after the capacitor. Any other noise source due to the loop filter itself or to the buffer can also be included in $\eta_{Vc.}$ Finally, the VCDL output is in the phase domain and contributes with a phase error represented by θ_{VCDL}. Using the linear model with noise sources of Fig. 3.b, the total r.m.s. normalized output jitter (in rad) can be calculated by adding the contribution of each of the blocks according to the following expressions [8]:

$$\sigma_{OUT} \approx \sqrt{\sigma_{in}^2 + \sigma_{VCDL}^2 + \sigma_{Vc}^2},$$
$$\sigma_{in} = \frac{2\pi}{T} \cdot \frac{\theta_{in}}{\sqrt{\left(1 - K_T^2\right)}} \cdot \sqrt{\left(K_T - 1\right)^2 + K_T^2 \cdot \left(1 - K_T^2\right)} \quad , (1)$$
$$\sigma_{VCDL} = \frac{2\pi}{T} \cdot \frac{\theta_{VCDL}}{\sqrt{\left(1 - K_T^2\right)}} \quad \sigma_{Vc} = \frac{2\pi}{T} \cdot K_{VCDL} \cdot \eta_{Vc}$$

where $K_T = \left(2\pi K_{VCDL} I_{CP}\right) / C$ is the loop gain in rad/V (always smaller than one), with K_{VCDL} representing the VCDL gain in s/V, I_{CP} the CP pulses amplitude in A, T the reference clock period in s, and C the loop filter capacitor value in F. The loop gain is actually the product of the loop bandwidth and the reference clock period:

$$K_T = w_n T = \frac{K_{VCDL} I_{CP}}{TC} \cdot T \,. \qquad (2)$$

From (2) it is easy to show that the contribution of the input jitter to the total output jitter increases weakly with the loop bandwidth, whereas the contribution of the VCDL jitter is reduced significantly when the loop bandwidth is increased. The contribution of the PFD, the CP, or the filter itself is not affected by the loop bandwidth, only by the VCDL gain. From the parameters that determine the loop bandwidth the most suitable for total jitter optimization is the capacitor value. The charge pump current is usually minimized at the circuit level design phase of the PFD and the CP to reduce the power consumption. The reference clock period is fixed by the system specifications and the gain of the VCDL is usually determined by the available supply voltage and the value of the VCDL tap delay, which are fixed by the specifications as well. The loop bandwidth also impacts other system parameters such as the lock-in time, but for UWB-IR applications the timing jitter is the most important performance parameter. Therefore, the goal of the DLL system level optimization is the minimization of the output jitter by finding the optimum loop filter capacitor value. Nevertheless, the presented model can be used to optimize any other DLL parameter.

The above analytical model predicts the value for the *out_clk* jitter. Indeed, it can be shown that the worst case jitter is found actually at the last delay cell of the VCDL [9], but for an impulse radio systems any of the intermediate VCDL delay cell outputs can also be selected. For system analysis purposes it would be interesting to easily determine the jitter at any output of the VCDL chain. Furthermore, some parameters are not constants, but depend on other magnitudes (e.g. K_{VCDL} is a non linear function of Vc). Introducing such dependencies in the analytical model of (1) is not straightforward. Most of these limitations are avoided by implementing a behavioral model of all the blocks, and specifically, by implementing each of the delay cells of the VCDL individually, as shown in the next section. Such a model is suitable for system level exploration and optimization including jitter.

3. DLL behavioral model including jitter

3.1. VCDL model

The VCDL is composed of 11 identical cells of a nominal delay of 227.27 ps. Each cell is fully differential. In this way, a total of 22 phases are obtained [10], allowing for 22 different delayed versions of the input reference clock of 200 MHz. The supply voltage of the delay cells is used to change its delay from 470 ps to 125 ps, by varying such voltage from 1.1 V to 1.8 V. The output jitter of each cell depends on the control voltage and is obtained using SpectreRF PSS + PNOISE analysis [11]. Fig. 4 shows the edge-to-edge jitter obtained with the simulator for the whole range of Vc values. This data is introduced in MATLAB and fitted using an exponential equation, which is used in the VerilogA model to generate the random variable that is added at each cycle to the delay between the input and the output. The behavioral model senses the input crossing at 50% of the power supply of 1.8 V and generates a delayed transition that includes jitter. An exponential fit is also used to implement the VCDL cell delay dependence on Vc, also shown in Fig. 4. The VerilogA code for this module is shown in the next page.

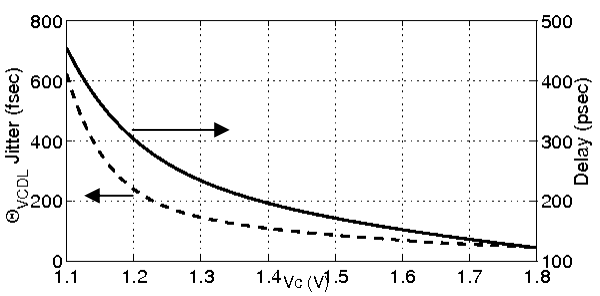

Fig. 4: Edge-to-edge r.m.s. jitter and delay for a VCDL cell versus the control (supply) voltage.

```
// VCDL Delay element
`include "constants.vams"
`include "disciplines.vams"

module vcdl ( VC, CK_IN, CK_OUT );
 input VC, CK_IN; output CK_OUT;
 electrical VC,   CK_IN, CK_OUT;
 parameter real Fref=200M from (0:inf);
 parameter real Vlo=0, Vhi=1.8;
 parameter real tt=50p from (0:inf);
 parameter real ttol=1f from (0:(1/Fref));
 parameter integer seed0=-500;
 parameter real Da1=1.29e+7,   Db1=-10.08,
  Dc1=828.5, Dd1=-1.065;
 parameter real Ja1=6.753e+10, Jb1=-17.18,
  Jc1=2194,  Jd1=-2.164;
 real Delay,jitter, dTp, dT, vout;
 integer seed;

 analog begin
  @(initial_step) begin
   seed=seed0;
  end
  @(cross(V(CK_IN)-0.9,+1,ttol))begin
   jitter = ( (Ja1*exp(Jb1*V(VC)))
   +(Jc1*exp(Jd1*V(VC))) )*1f;
   Delay  = ( (Da1*exp(Db1*V(VC)))
   +(Dc1*exp(Dd1*V(VC))) )*1p;
   dT=jitter*$rdist_normal(seed,0,1);
   dTp=dT+Delay-tt/2;
   vout=Vhi;
  end
  @(cross(V(CK_IN)-0.9,-1,ttol))begin
   jitter = ( (Ja1*exp(Jb1*V(VC)))
   +(Jc1*exp(Jd1*V(VC))) )*1f;
   Delay  = ( (Da1*exp(Db1*V(VC)))
   +(Dc1*exp(Dd1*V(VC))) )*1p;
   dT=jitter*$rdist_normal(seed,0,1);
   dTp=dT+Delay-tt/2;
   vout=Vlo;
  end
  V(CK_OUT)<+ transition(vout,dTp,tt);
 end
endmodule`include "constants.vams"
```

3.2. PFD+CP and buffer models

The inputs of the PFD are the two signals *ref_clk* and *out_clk*. The contribution to the system noise of this block is characterized together with the CP. The VerilogA code for the PFD+CP is shown below. It generates a signed current with a duration proportional to the time difference between the two PFD input signals, as shown in Fig. 5. This current is fed into a capacitor that integrates it and generates the *Vc* voltage. The PFD+CP add some noise to the control signal that is characterized in the following way. A r.m.s. current noise is found at the output of the PFD+CP in the locked state (i.e. when the time difference between its two outputs is zero) using SpectreRF PSS + PNOISE analysis. This noise in the current domain is transformed into an input equivalent jitter that would produce such perturbation in the CP current. The output current noise is transformed by the loop filter capacitor into voltage noise (represented by η_{Vc} in Fig. 3b). Additionally, the PFD+CP verilogA model naturally transforms any jitter found at their inputs into current noise at its output.

```
// VerilogA for CP_PFD
`include "constants.vams"
`include "disciplines.vams"

module cp_pfd (CK_REF, CK_OUT, IOUT);
 input CK_REF, CK_OUT; output IOUT;
 electrical CK_REF, CK_OUT, IOUT;
 parameter real Fref=200M from (0:inf);
 parameter real tt=50p from (0:inf);
 parameter real ttol=1f from (0:(1/Fref));
 parameter real Vlo=0,Vhi=1.8;
 parameter real a1=0.02237,   b1=-0.01568,
   c1=0.009939,  d1=-0.0005568;
 parameter real a2=-0.1211,   b2=-0.06336,
   c2=0.09319,   d2=-0.02578;
 parameter real a3=1.037e-06, b3=-2.906e-05,
   c3=-  0.0006997, d3=-0.004114, f3=-0.01039;
 parameter jitter=2000f;
 real Delay,flux,timeREF,timeOUT,voutREF,voutOUT;
 real dT;
 integer seed;

 analog begin
  @(initial_step)begin
   voutREF=Vlo;   voutOUT=Vlo;    flux=0;
   seed=-3456;
  end
  @(cross(V(CK_REF)-0.9,+1,ttol))begin
   //CP_PFD  noise  modeled as input jitter
   dT=jitter*$rdist_normal(seed,0,1);
   timeREF=$abstime+dT;
   voutREF=Vhi;
   if (voutOUT>0.9) begin
     Delay=(timeREF-timeOUT)*1e+12;
     if((abs(Delay) >  25) && (voutREF>0))
        flux = Delay*(a1*exp(b1*abs(Delay))
        +c1*exp(d1*abs(Delay)));
     else if((abs(Delay) <= 25) && (voutREF>0))
        flux = pow(abs(Delay)/25,1)*Delay*
        (a1*exp(b1*abs(Delay))
        +c1*exp(d1*abs(Delay)));
   end
  end
  @(cross(V(CK_OUT)-0.9,+1,ttol))begin
   timeOUT=$abstime;
   voutOUT=Vhi;
   if (voutREF>0.9) begin
     Delay=(timeREF-timeOUT)*1e+12;
     if((abs(Delay) >  25) && (voutOUT>0))
       flux =  Delay*(a1*exp(b1*abs(Delay))
       +c1*exp(d1*abs(Delay)));
     else if((abs(Delay) <= 25) && (voutOUT>0))
       flux = pow(abs(Delay)/25,1)*Delay*
       (a1*exp(b1*abs(Delay))
       +c1*exp(d1*abs(Delay)));
   end
  end
  @(cross(V(CK_REF)-0.9,-1,ttol))begin
    voutREF=Vlo;
  end
  @(cross(V(CK_OUT)-0.9,-1,ttol))begin
    voutOUT=Vlo;
  end
  I(IOUT) <+ transition(flux*1u, 0, tt);
 end
endmodule
```

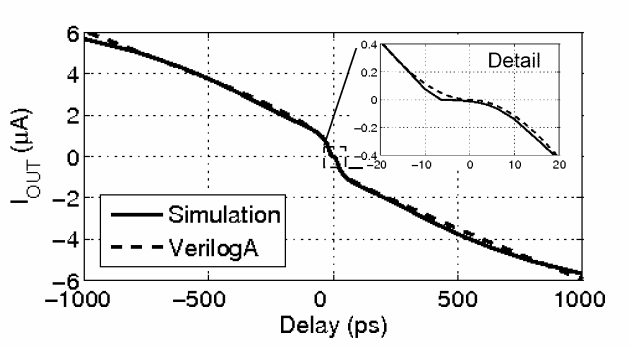

Fig. 5: Time to current characteristic of the PFD+CP block.

The buffer also contributes with random voltage noise that is added to *Vc*, as shown in the characterization results of Fig. 6. In this case, the noise model is fitted to the transistor level simulation data (shown with crosses in the figure) for *Vc* < 1.62 V only, to keep the model simple. Higher values of *Vc* are only possible during the start-up phase of the DLL. Its VerilogA code follows:

```
// Noise of buffer
`include "constants.vams"
`include "disciplines.vams"

module buffer (IN, OUT);
 input IN; output OUT; electrical IN, OUT;
 parameter real Fref=200M from (0:inf);
 parameter real tt=50p from(0:inf);
 parameter real offset=6m;
 parameter real noiseRMS=0.5m;
 real dV;
 real vout;
 integer Seed;

 analog begin
  @(initial_step)begin
    Seed=-1459;
  end
  dV=2*noiseRMS*$rdist_normal(Seed,0,1);
  vout=V(IN)+offset+dV;
  V(OUT)<+transition(vout,0,tt);
 end
endmodule
```

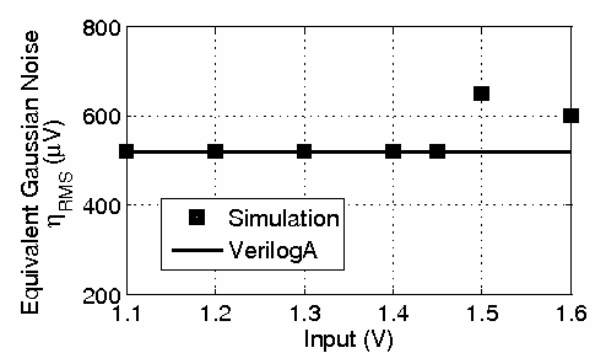

Fig. 6: r.m.s voltage noise at the buffer's output.

3.3. Reference input model

The *ref_clk* input also contributes with noise in the phase domain represented by θ_{in} in Fig. 3.b. In this case, the jitter is a constant parameter that is set as a constraint in the system level design process. The VerilogA code is very similar to the VCDL cell model but without the dependence on *Vc*.

4. System level simulation and verification

The VerilogA modules described in the previous section are connected together as shown in Fig 3 and the complete DLL is simulated at system level. VerilogA code is added to the VCDL cell modules for writing the transition times of each cycle to a file. A post-processing of this file allows the computation of the mean and the standard deviation of the edge-to-edge times of any VCDL cell output. The standard deviation is the jitter we want to minimize. Fig. 7 shows the results of a series of system level behavioral simulations sweeping the loop filter capacitor value for three different *ref_clk* input jitter values. In these simulations I_{CP} = 8.8 µA, T = 5 ns, and K_{VCDL} = 8.8ns/V in the locked state. For this particular DLL the jitter is minimized for loop capacitor values in the range of a few pF that correspond to bandwidths in the range of a few

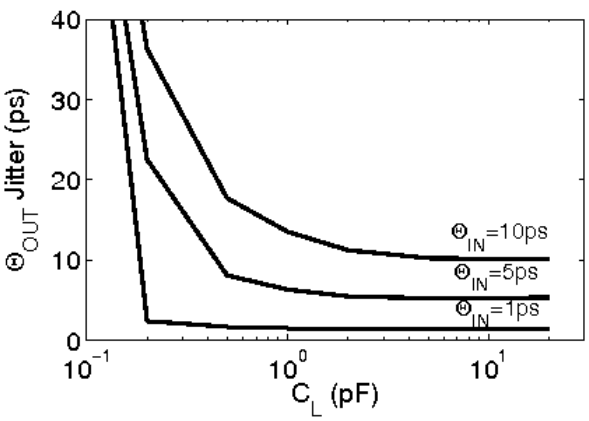

Fig. 7: System level behavioral simulation results for output jitter.

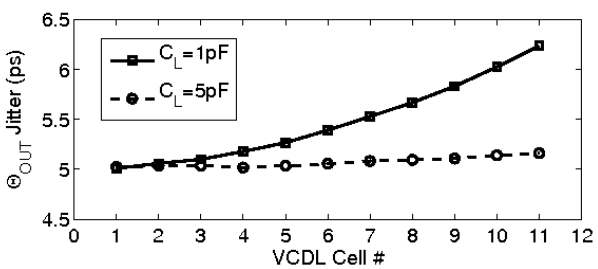

Fig. 8 Jitter at the different VCDL cells output.

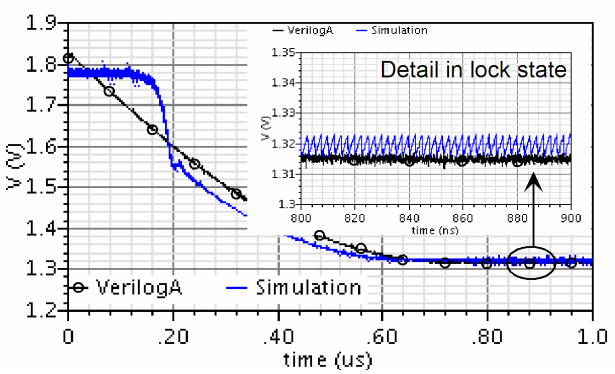

Fig. 9: Comparison of transistor level (and behavioral model DLL transient response.

MHz. The flexibility of the model is illustrated in Fig. 8, where the jitter at the different VCDL cells obtained by a single simulation of the DLL is shown, for two different loop capacitor values and θ_{in} = 5 ps.

The behavioral system level model of the DLL is validated against transistor level simulations for the particular value of the loop filter capacitor of 5 pF and θ_{in} = 0 ps. Fig. 9 shows the DLL dynamic response of the filtered CP output from the power-down state until the loop achieves lock for system level and transistor level simulations. Transistor level simulation consists on a transient analysis. Once the transient analysis has achieved the periodic steady state (i.e. the DLL is locked) a PSS + PNOISE analysis is performed. Such analysis is used to obtain the output jitter from the transistor level simulation of the whole DLL. The output jitter for the VerilogA behavioral simulation results in 910 fs, and the transistor level simulations gives a result of 954 fs[1]. Both simulations give the same result for the steady state control voltage value (1.315 V) and lock time (700 ns).

5. Conclusions

In this paper a behavioral model in VerilogA of a DLL used in a UWB Impulse Radio receiver has been presented. The behavioral model has been implemented in a modular way: one module for each one of the cells of the VCDL, another for the Phase-Frequency Detector, Charge Pump and Loop Filter blocks, and another for the control voltage buffer. Each one of these modules includes jitter generation and its dependency with some parameters such as the VCDL control voltage or the loop capacitor value. The jitter data has been extracted from transistor level simulation of each block.

In this way, it is possible to simulate the whole DLL at system level including also jitter generation, enabling an agile block sizing procedure for DLL output jitter minimization. Such optimization at transistor level would be unfeasible due to the great amount of time it would require. For example, the simulation times corresponding to Fig. 9 are 2437 s for the transistor level case (tran+PSS+PNOISE) and 195 s for the behavioral model. The simulations were launched on a Pentium 530 CPU running at 3 GHz in 32 bits mode.

The behavioral model has been checked against transistor level simulations for some specific values of the design parameters obtaining almost the same dynamic behavior and output jitter in the locked state.

[1] The comparison is done with θ_{in} = 0, since it is difficult to specify a input jitter in SpecreRF tran or PSS+PNOISE simulations.

References

[1] FFC, "First Report and Order: Revision of part 15 of the commission's rules regarding ultra-wideband transmissions systems," FCC ET Docket 98-153, Apr.2002.

[2] M. Z. Win and R. A. Scholtz, "Ultra-wide bandwidth time-hopping spread-spectrum impulse radio for wireless multiple-access communications," *IEEE Communications Letters*, vol. 48, no. 4, pp. 679-689, Apr.2000.

[3] M. Verhelst, W. Vereecken, M. Steyaert, and W. Dehaene, "Architectures for Low Power Ultra-Wideband Radio Receivers in the 3.1-5GHz Band for Data Rates < 10Mbps," in *Intl.Symp.on Low Power Electronics and Design,* 2004, pp. 280-285.

[4] P. Heydari, "A study of low-power ultra wideband radio transceiver architectures," in *2005 IEEE Wireless Communications and Networking Conference,* 2 ed 2005, pp. 758-763.

[5] E. Barajas, R. Cosculluela, D.o Coutinho, M. Molina, D. Mateo, J.L. González, I. Cairò, S. Banda, M. Ikeda, "A Low-Power Template Generator for Coherent Impulse-Radio Ultra Wide-Band Receivers," in *IEEE Conference on Ultra Wideband Systems and Technologies,* 2006.

[6] K.S. Kundert, O. Zinke, *The Desinger's Guide to Verilog AMS*, New York : Kluwer Academic Publishers, 2004.

[7] A. Chandrakasan, W.J. Bowhill, F. Fox, *Design of High-Performance Microprocessor Circuits*, New York: IEEE Press, 2001.

[8] R.L. Aguilar, D.M. Santos, "Modeling Charge-pump Digital Delay Locked Loops," in *Proc IEEE Conf. on Electronics, Circuits and System* , September , 1999.

[9] R.C.H. van de Beek, E.A.M. Klumperink, C.S. Vaucher, B. Nauta, "Low-jitter clock multiplication: a comparison between PLLs and DLLs," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing,* , vol.49, no.8, pp. 555- 566, Aug 2002.

[10] H.-Y. Huang, J.-H. Shen, "A DLL-Based Programmable Clock Generator Using Threshold-Trigger Delay Element and Circular Edge Combiner," in *IEEE Asia-Pacific Conf. on Advanced Systems Integrated Circuits*, 2004, pp. 76-79.

[11] *Virtuoso Spectre RF Simulator*, 2006 Cadence Design Systems, Inc., http://www.cadence.com

[12] Ken Kundert, "Modeling and simulation of jitter in PLL frequency synthesizers," Available from http://www.designers-guide.org/Analysis.