Design of process invariant Delay Lock Loop (DLL) ECE 6770 - Final Report

Manohar Nagaraju (manohar.nagaraju@utah.edu)
Department of Electrical and Computer Engineering,
University of Utah

Abstract: Random device mismatch have a significant impact on the performance of analog circuits. This report discusses the design of a Delay Lock Loop (DLL) which is insensitive to process variation. The DLL is optimized for reduction in the variation of threshold voltage variations and is intended to be used in a clock and data recovery circuit with an input bit rate of 500 Mbps. The performance specifications of the DLL are also analyzed. The DLL is expected to be fabricated in the AMI 0.6 um technology.

Index Terms: Process variation, Delay Lock Loop, High speed counters

I. Introduction

Wafer-to-wafer and die to die variations present significant power-speed-yield trade-offs. The problem of process variations become all the more predominant with the scaling of devices for each new generation. It has been proved that there is an increase of 100% in energy consumption for the same performance due to threshold voltage variations in inverter chains (90nm) [1]. The problem of process variations is all the more complicated in analog circuits. They have a considerable effect on the bias conditions, gain, frequency response and bandwidth of the circuit. These challenges are complex and necessitate design-stage statistical analysis to determine conditions for maximum yield [2].

An improvement in process and fabrication control is not expected to have a high impact in controlling these process variations. This problem is expected to be overcome by careful design of circuits. With the above problem in mind, the plan for the present project will be to research the effects of process variations on the phase accuracy of DLL (which is chosen as a benchmark) outputs and build a process-invariant DLL.

II. Related work

James W. Tschanz, et al [3] have proposed using Adaptive body bias (ABB) technique to compensate for die-to-die parameter variations by applying optimum PMOS and NMOS body bias voltage to each die which maximizes the die frequency subject to a power constraint. However typically low V_t devices are required for this scheme and they have worse short-channel effects, and these effects degrade with body bias. Also, the effect of Drain induced barrier lowering (DIBL) becomes significant with body bias and within-die V_t variation depends on V_t-roll-off and DIBL resulting in a larger V_t variation with body bias. Moreover, as the technology is scaled, the control of the body's terminal on the channel charge diminishes.

Anand M. Pappu, et al [4] presents a design methodology to develop circuits that compensate for process variations without the need for post-fabrication efforts. The main advantage of this methodology stems from the fact that we can use it to optimize the circuit to reduce variation on a parameter which we consider important such as gain, bandwidth etc.

III. Design and Description

Having understood the significance of designing circuits for reduced process variation sensitivity, we now discuss the design of a DLL, which is the focus of the present project. The DLL is planned to be used in a Clock and Data recovery circuit. The basic block diagram of a DLL is as shown below:

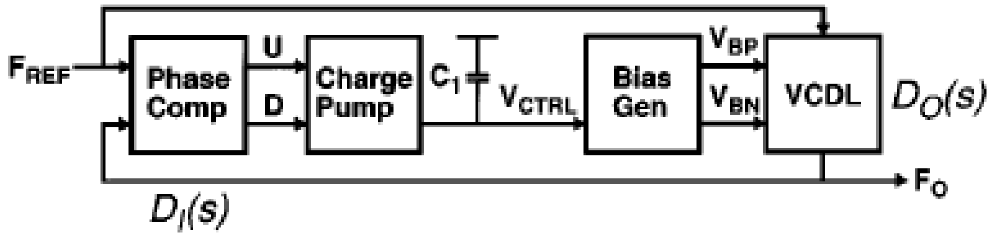

Fig 1: Block Diagram of the DLL.

The signal V_{in} is the reference frequency generated by a high quality crystal. It is the input to the voltage controlled delay line (VCDL) which basically consists of a series of delay cells each producing a fixed time delay between its input and output. The total delay through the VCDL should be equal to one time period of the input frequency. The output of the delay elements generate waveforms with edges that are evenly spaced within one period of the reference crystal. The phase detector senses the phase difference between the input and output of the delay chain and generates an error signal. This error signal is used to control a charge pump, whose output is filtered by a loop filter. The filter output is the control voltage that varies the time delay of each stage to minimize the phase error. When the DLL is in locked condition, the input and the output of the delay chain are in-phase. A DLL based clock recovery circuit or a frequency synthesizer has an inherent advantage over a corresponding Phase Lock Loop (PLL) using a Voltage Controlled Oscillator (VCO). In an oscillator, random timing errors accumulate because the timing jitter at the end of each oscillation is the starting point of the next and thus the random timing error of the output signal is the sum of timing errors of all previous oscillations which results in poor long-term jitter performance. In contrast, in a DLL based system, the random timing error accumulates only within one clock cycle as the next cycle is triggered by the next output oscillation from the reference crystal frequency. This results in an excellent long-term jitter performance.

a) DLL Performance Analysis:

The two main performance specifications for a DLL based clock recovery circuit is the phase noise and spurious tones. This section discusses briefly these performance metrics.

1) Phase Noise: It is defined as the random timing fluctuation of the clock as shown in figure 2. The random timing jitter in the time domain manifests itself as phase noise (Fig 3) in the frequency domain. An ideal oscillator generates a pure sine wave without any jitter which corresponds to a single line in the frequency domain representation.

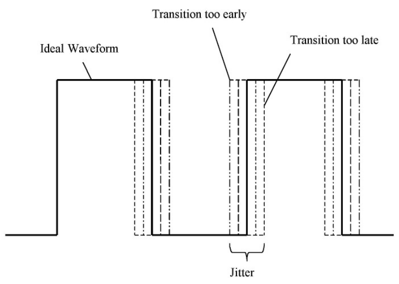

Fig 2: Timing Jitter

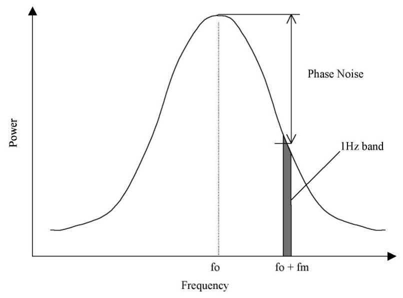

Fig 3: Oscillator power spectrum (composed of phase noise)

Phase noise is typically measured as the power spectral density of the noise as a function of the frequency compared to the carrier power at the carrier frequency specified in dBc/Hz. The phase noise of a DLL based system is affected by two main factors namely the phase noise of the input frequency by the crystal oscillator and the phase noise contribution by the delay chain. Generally the reference frequency is generated using high quality crystal oscillators and the phase noise due to this is ignored. The analysis of the phase noise of a delay chain is done in [2]. It assumes that the delay chain is driven by a perfect crystal and each element of the delay chain provides exactly the same delay. For the 5-stage delay chain, the power spectral density of the phase noise is given by:

$$S_X(\Omega) = \left[2 + \frac{12}{5}\cos(\Omega) + \frac{6}{5}\cos(2\Omega) + \frac{2}{5}\cos(3\Omega)\right] \cdot \sigma^2$$

where Ώ is digital frequency. The following figure shows the plot of this function.

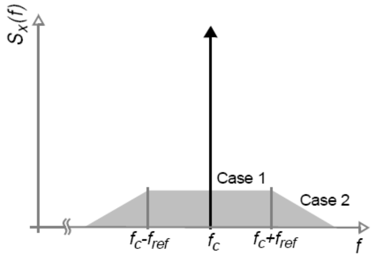

Fig 4: Phase noise plot for 5 stage delay example

The timing error in a DLL accumulates over only one cycle of the input frequency. Hence the random timing error in one cycle is independent or uncorrelated to the random timing error of the next cycle. Thus there is a flat region in the phase noise plot and rolls off as the two timing indices approach each other within the period of the reference crystal frequency.

3) Spurious Tones: While timing jitter manifests itself as phase noise with a range of frequencies in the frequency spectrum, spurious tones result in spikes in the frequency domain as shown below:

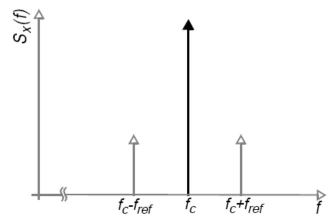

Fig 5: Spurious tones

The main contributor to spurious tones is mismatches between the different elements of the delay chain. If one of the delay stages has a mismatch mainly because of process variation, then the delay through it might be shorter/longer compared to the delay of the other stages. Thus the output of the delay chain is shifted by this amount. Since the DLL is locked to the reference frequency, this mismatch also occurs at the same frequency in the output spectrum. Figure shows this effect in the time domain for a 5 stage delay chain.

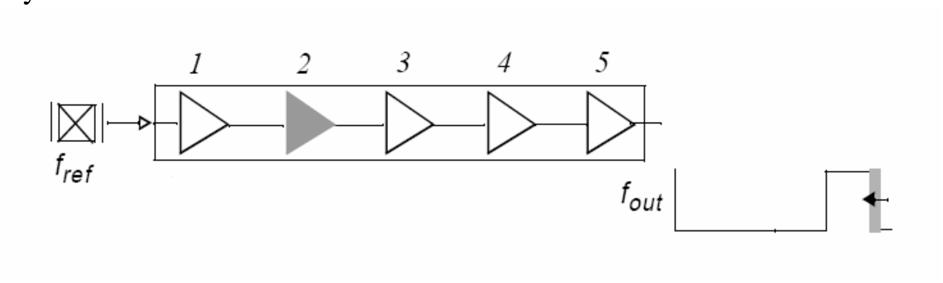

Fig 6: Delay stage mismatch.

b) Design Information:

The focus of the project will be to build a process invariant delay cell so that the time periods of the synthesized output will be equal. The time period of the synthesized waveform is affected by the delay of each cell. One point to be noted is that we assume that the other blocks of the DLL such as the edge combiner are functioning ideally. The main factor affecting the delay of the cell will be the mismatch of the various parameters of the transistors in the delay cell. Suppose for example the threshold voltage of a transistor driving the output in a delay cell decreases, the delay through the particular cell decreases. In order to overcome these variations of the transistor parameters of each delay cell careful circuit design is required. The most important parameters to be considered for process variations are threshold voltage, dL and dZ effects. However, the dL and dZ effects are not predominant in the technology we will be using for the project (0.5um) compared to say in a real industrial process like 45nm. So the scope of this project will be limited to mitigating the effects of threshold voltage variations only. The strategies planned to reduce the effect of the threshold voltage variations are:

1) Sizing: Determine effective sizing of the transistors in the standard delay cell so that process variation is minimized.

2) Calibration: Include means of calibrating the delay cell to adjust Vt of critical transistors to equalize the delays. Thus suggest an alternate/modified architecture for the delay cell.

c) Module Level Details and Circuits

The main blocks of the DLL are the Voltage Controlled Delay Line (VCDL), phase detector, charge pump and the calibration circuitry. The following sections explain briefly the design of each of the modules.

i) Voltage Controlled Delay Line:

The current starved inverter has been chosen as the delay cell for its simplicity in controlling the delay by using the bias voltage. Figure shows the schematic of the current-starved inverter:

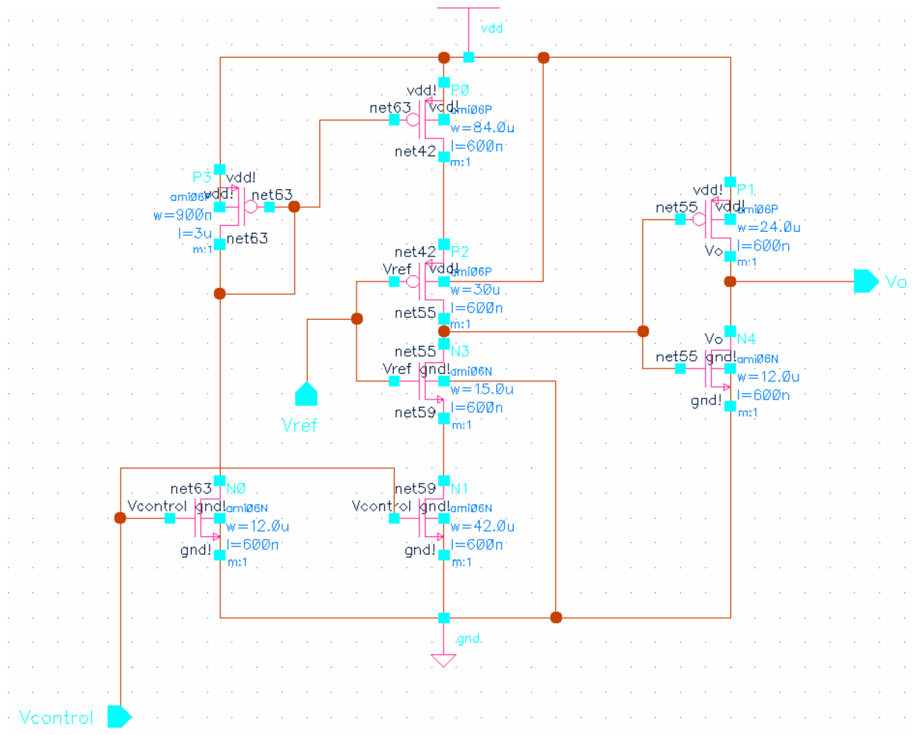

Fig 7: Delay Cell Schematic

The delay cell is a single-ended inverter, consisting of P2 and N3 in series with P0 and N1 operating in the triode region. The delay of the circuit is determined by the equivalent resistance of P0 and N1 which in turn is controlled by the bias voltage. An additional buffer comprising of P1 and N4 serves as an output buffer for high frequency operation. The circuit performs a rail to rail operation, so it consumes no static power.

Delay Cell Characterization: An accurate analysis of the circuit was performed in order to determine the delay through the circuit. It is to be noted that changing the value of Vcontrol does not affect the delay through the second stage inverter. So first an expression for the delay through the first stage was determined as a function of Vdc and the circuit parameters. The delay of the circuit for a transition from a ‘0’ to ‘1’ in turn consists of three parameters. When Vref changes from Vdd to zero, the PMOS transistors in turn drive the output node to high voltage. Initially both transistors P0 and P2 will be in saturation until the output voltage becomes equal to 'Vt', the threshold voltage. This time is given by

$$T_1 = \frac{(W/L)_{p3}[(V_{gs})_{p3} - (V_{tp})_{p3}]^2 C_{out} (V_{tp})_{p2}}{(W/L)_{P0}[(V_{gs})_{P0} - (V_{tp})_{P0}]^2 \mu_n C_{ox} (W/L)_{N0}[(V_{gs})_{N0} - (Vtn)_{N0}]^2}$$

Then P2 comes out of saturation but P0 is still working in the saturation. The time instant until which even P0 comes out of saturation is given by

$$T_2 = \frac{C_{out}[V_{dd} - \sqrt{(\mu_n/\mu_p)\frac{(W/L)_{N0}}{(W/L)_{P3}}\{V_{dc} - (V_{tn})_{No}\}^2}\,]}{\mu_p (C_{ox}/2)(W/L)_{P0}[(V_{gs})_{P0} - (V_{tp})_{P0}]^2}$$

The third region is when both transistors are in saturation. This time is given by:

$$T_3 = \frac{C_{out}}{\beta_p (V_{dd} - V_{tp})}$$

The transition time from low to high voltage is the sum of these three time instants. In order to minimize their variations due to variations in threshold voltage, we can differentiate the total time with respect to each of the threshold voltages which affect the delay namely those of N0, P0, P2 and P3 and equate it to zero. On the basis of this, we can conclude that $(W/L)_{N0}$ and $(W/L)_{P0}$ should be sufficiently high and $(W/L)_{P3}$ should be sufficiently low. This agrees well with one's intuition as in order to make the current and hence the delay less sensitive to voltage variations, one should have a sufficiently high current which is set by having a high ratio for the mirror transistors P0 and P3. A similar analysis was carried out for the fall time for which the bias current is set by the transistor N1 which is made sufficiently big in order to reduce the variations.

A monte carlo simulation was carried out for estimating the delay variation before and after optimization. The following figure shows the results. The standard deviation has shown a remarkable reduction from 4.459ps to 2.097ps which is more than a 50% reduction in the variation.

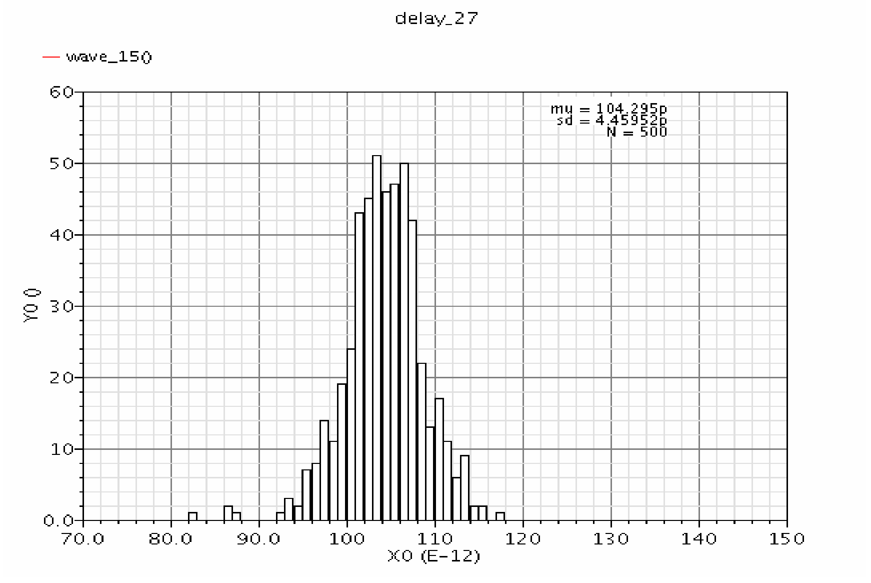

Fig 8: Delay variation for 500 simulations. (Before optimization)

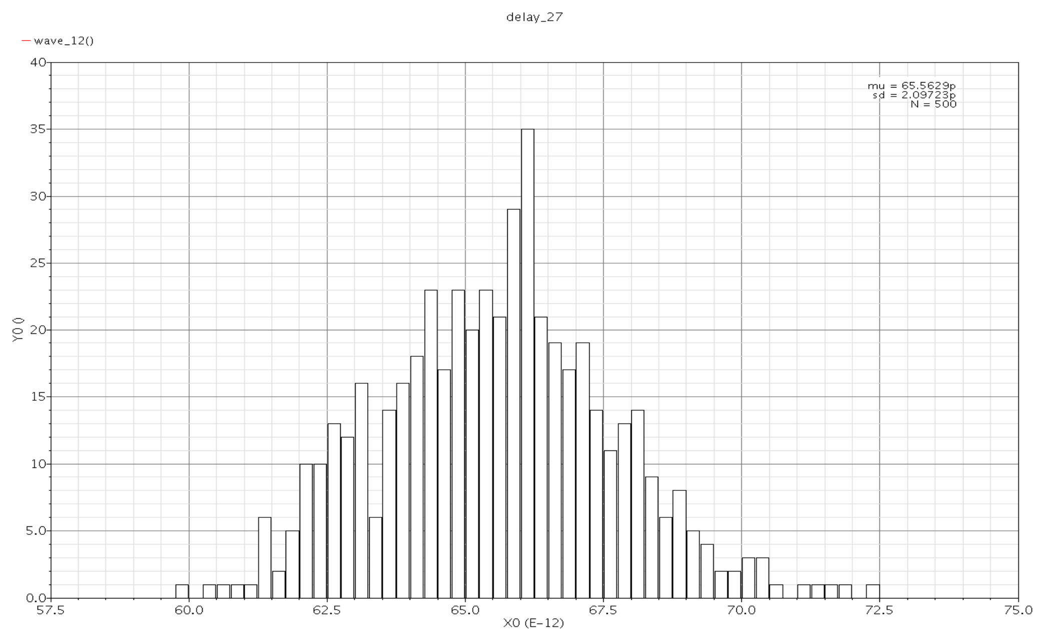

Fig 9: Delay variation for 500 simulations (After optimization)

ii) Phase Detector: A circuit that can detect both phase and frequency detectors proves extremely useful because it significantly increases the acquisition range of the DLL. Unlike XOR gates, PFD's generate two outputs that are not complementary Figure shows the schematic of the phase detector

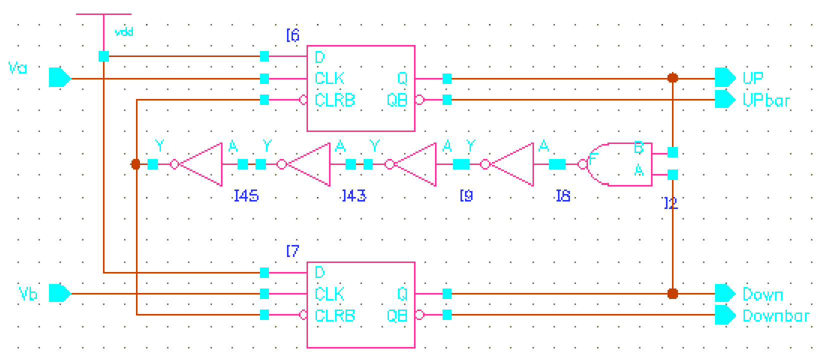

Fig 10: Schematic of the Phase Frequency Detector.

Signals Va and Vb act as clock signals of the two D flip flops. We note that UP and Down signals are zero, then a transition on Va causes UP tp go high. Subsequent transitions on Va have no effect on the UP signal, and when Vb goes high, the AND gate activates the reset of both the flip flops. Thus UP and Down signals are simultaneously high for a duration given by the total delay through the AND gate and the reset path of the flip flops. Accurate matching is required between the signals UP and Down currents as mismatches causes ripple on the control voltage, which can lead to spurs in the output spectrum. To achieve a better matching between the signals UP and Down, extra delay is inserted in the reset path to increase the pulse width of both the signals by the same amount.

iii) Charge Pump: A phase detector and a low pass filter arrangement without a Charge Pump has the disadvantage that the charge deposited on the capacitor after each phase comparison decays. In a charge pump on the other pump, there is negligible decay of charge between phase comparison instants. An important conclusion of this is that even infinitesimal phase error would result in an indefinite accumulation of charge on the filter capacitor. Shown in the Fig 11 is a charge pump with a unity gain feedback amplifier, which was constructed using an op-amp. This serves to keep both the voltages same when switches P1 and N1 controlled by UPbar and Down, thus eliminating the charge sharing problem between the parasitic capacitors and the loop filter. The Bandwidth of the DLL is dependent upon the biasing current of the charge pump, a higher current making it wide-band. The bias current will be made as an external input.

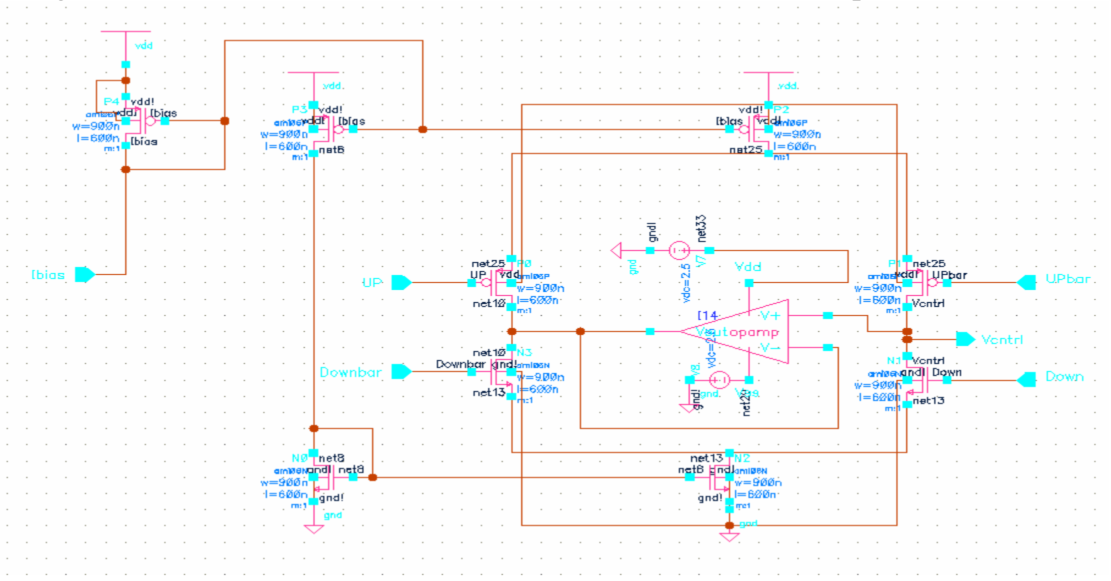

Fig 11: Schematic of the Charge Pump

The op-amp used in the charge pump is a basic two-stage op-amp optimized for a unity gain frequency of 36MHz and a phase margin of 105^{o}. Figure 12 shows the schematic of the two stage op-amp. The op-amp is biased using the same current source as the charge-pump with the current being adjusted by the use of current mirror circuit.

iv) Loop Filter: The loop filter just consisted of a capacitance which controls the Vcntrl fed to the delay cells. The value of the capacitance decides the settling time of the DLL, a low capacitance ensuring a fast settling time. The value of the capacitance can be calculated from the equation:

$$\omega_p = \frac{(K_d \times I_{cp})}{(C \times 2 \times \Pi)}$$

where w_p is the pole frequency, K_d is the delay cell gain, I_{cp} is the charge pump current and C is the capacitance. A nominal value of 50pF has been used as of now for a fast settling time and a proper, pole-zero analysis needs to be done depending on the bandwidth requirements.

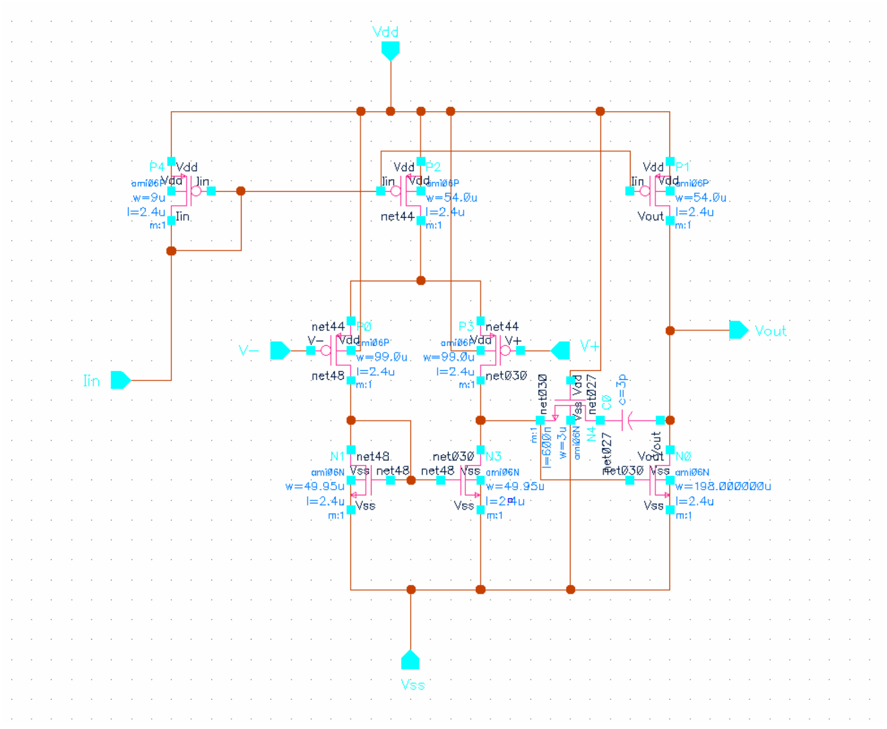

Fig 12: Schematic of the 2-stage op-amp

d) <u>DLL Operation:</u> All the components of the DLL were wired up together and tested for the lock operation. A total of 12 delay cells were used in the delay chain with 3 of the delay cells configured as a set in order to use it as a ring oscillator during initial setting of the control voltage. This will be explained in the later sections. An input frequency of 500MHz was supplied. The following figure shows the transient waveforms of each of the delay cells and the control voltage. The total delay through the delay chain slowly adjusts itself to the period of the input signal as shown below.

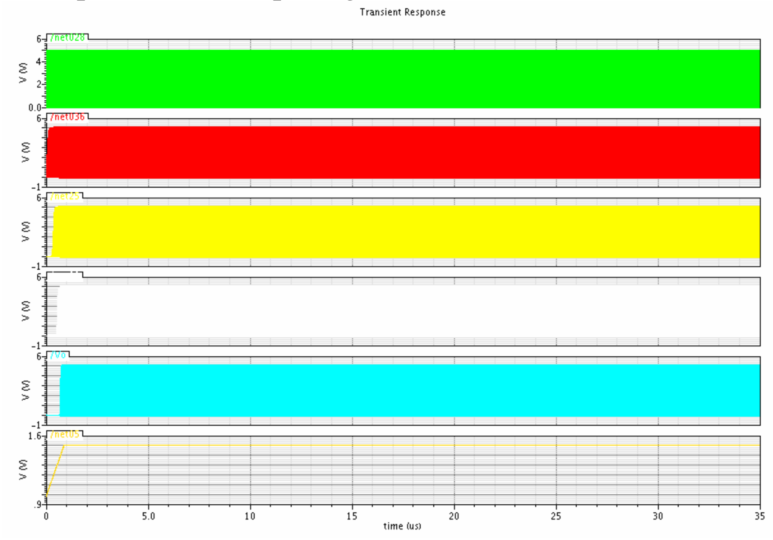

Fig 13: Transient simulation of the DLL

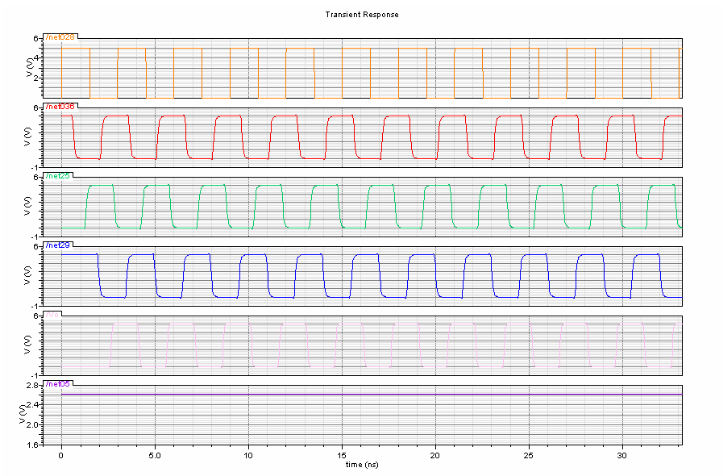

Fig 14: Transient simulation of the DLL – in unlocked state (at start time)

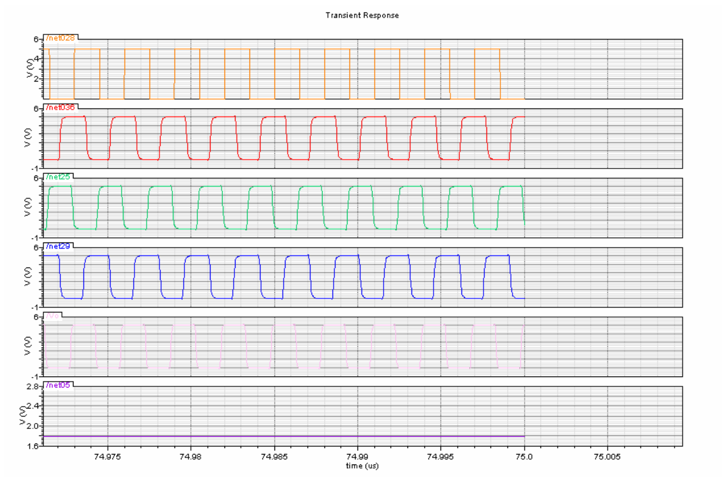

Fig 15: Transient simulation of the DLL – in locked state (at end time)

e) <u>Approach for mitigating process variation:</u>

As explained earlier 3 of the delay cells will be used as a ring oscillator. The idea is to configure each of the set of 3 delay cells as a ring oscillator. The frequency of each of the 4 ring oscillators is measured using a single high speed counter in a time multiplexed fashion. The frequency of oscillation of the ring oscillator is directly proportional to the delay through the delay cells. By adjusting the frequency of oscillations of each of the ring oscillators to the same value, we ensure that each delay cell produces a fixed delay independent of variations in the threshold voltage.

However, the ring oscillator produces a high frequency of around 1GHz and counting the frequency using a normal flip-flop did not yield expected results. Hence a flip flop using a ratioed logic D-latch was designed. This flip flop was used in a asynchronous counter in order to eliminate the logic gates otherwise necessary in a synchronous counter and maximize the frequency of operation. Figure 16 and 17 shows the schematic of the flip flop and the 8-bit asynchronous counter respectively. Figure 18 shows the output of the asynchronous counter when its clock input is fed from one stage of the ring oscillator discussed above.

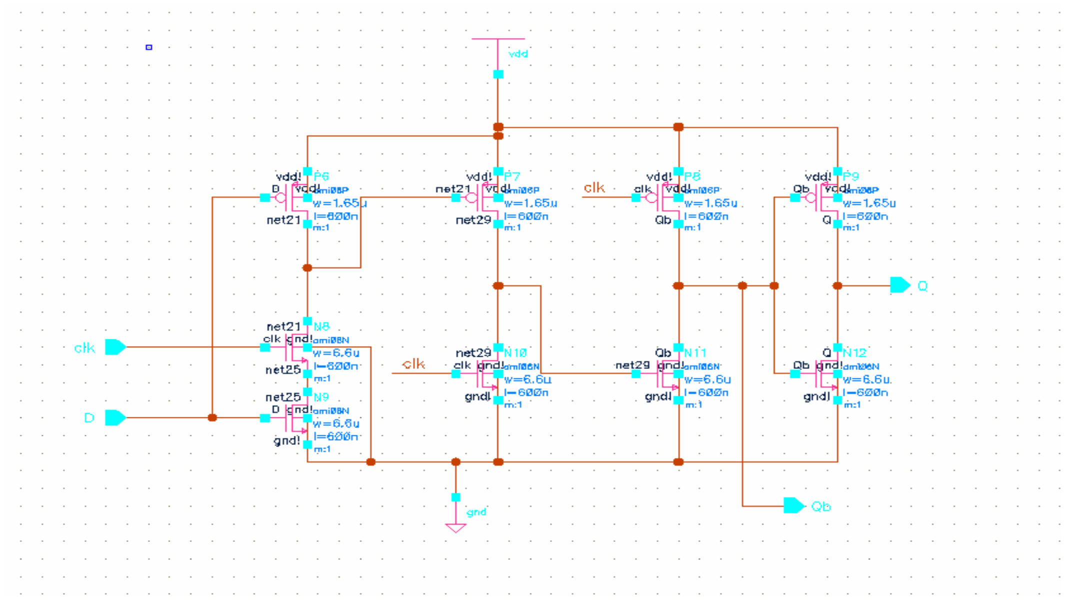
Fig 16: Ratioed Logic D flip flop

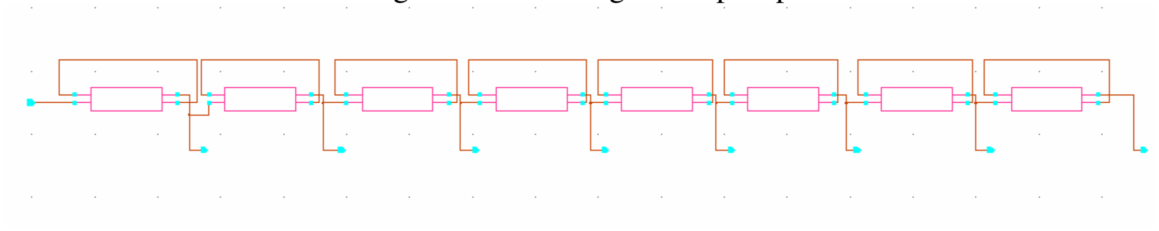
Fig 17: 8-bit asynchronous counter

The loop around one set of the delay cells (3 in number) was closed and the following figure shows the output from the ring oscillator and the counter output.

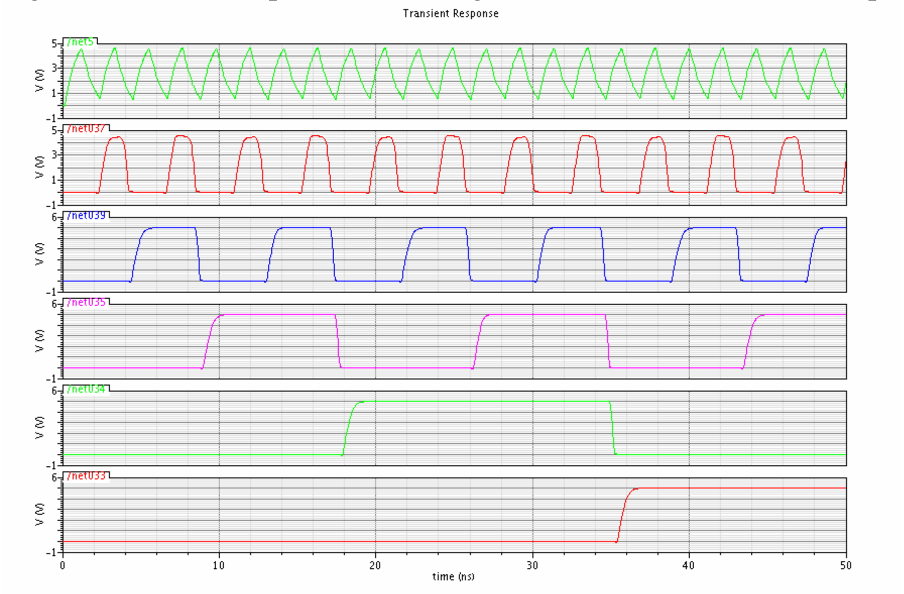

Fig 18: Output from the ring oscillator and the asynchronous counter

The 4 ring oscillators are operated sequentially with the output being fed to the same asynchronous counter in a time multiplexed fashion. The values of the count obtained from each of the ring oscillators are directly proportional to the delay through the cells. These count values can be used to calibrate the delay cell so that each of set of 3 delay cells is identical. The variation in the threshold voltage can be adjusted to equalize the delay by using body bias technique which in turn controls the current through the cell. A higher current results in lesser delay while a lower current results in higher delay. However using the adaptive body bias technique has the following disadvantages:

1) Short channel effects namely V_t-roll-off and drain induced barrier lowering (DIBL) degrade further with body bias.
2) Within-die V_t-variation due to within-die variation in the critical dimension will depend on V_t-roll-off and DIBL. Hence, within die V_t-variation increases with body biasing.
3) The increase in within-die V_t-variation due to adaptive body bias worsens with scaling and is more pronounced under aggressive V_t-scaling.
4) The control variable obtained from the asynchronous counter is digital in nature and requires a Digital-to-analog converter to obtain a suitable body voltage for Adaptive body biasing.

A possible solution to this problem is to directly control the current supplied to the current-starved inverter of the delay cell. This can be achieved by splitting the big current mirror transistor which supplies current to the inverter branch into smaller transistors of increasing width. Then depending on the value obtained from the digital logic, we can switch ON/OFF some of the transistors such that it supplies the required current. Figure 19 shows the schematic which implements the above proposal. In the schematic shown, the output bits from the counter (b_o to b_7) control the transistors which operate as switches in order to enable/disable the particular branch from sourcing current through

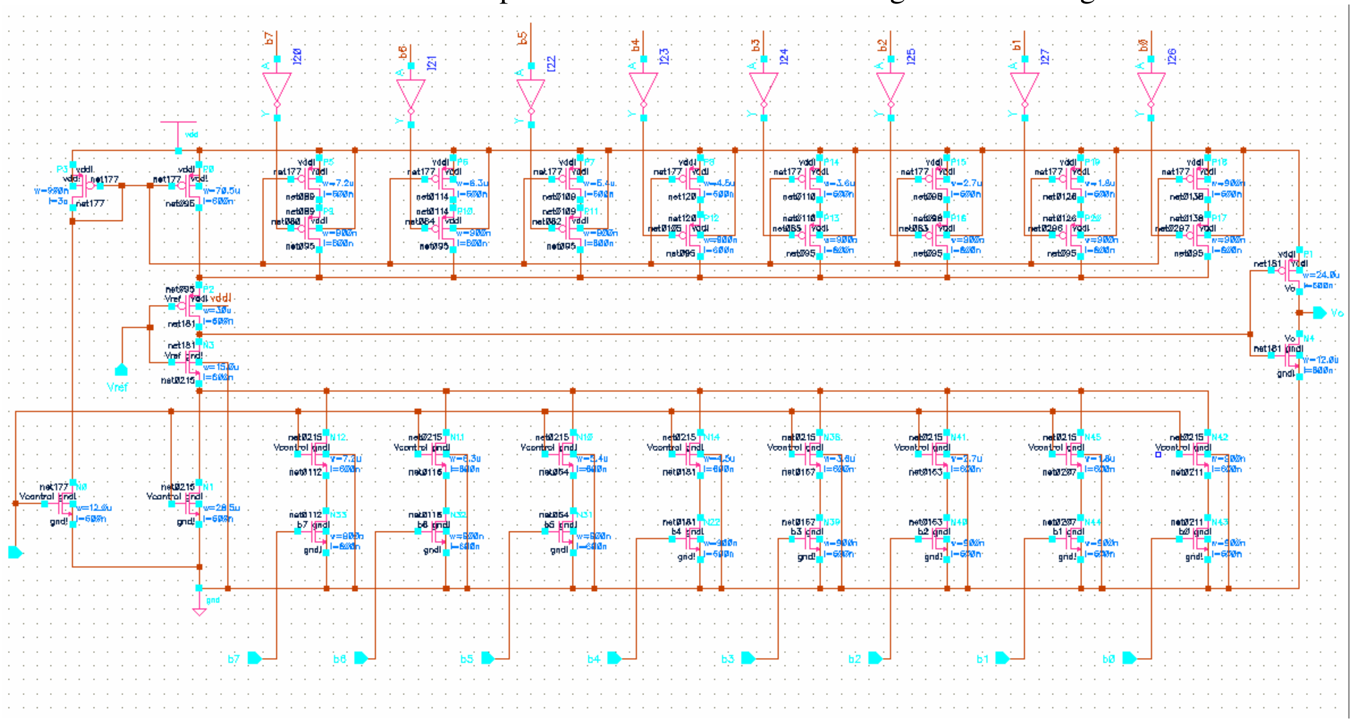

Fig 19: Modified schematic of the delay cell

the inverter branch. By having the transistor widths in an increasing order starting from the minimum width, we can have a wide range of control over the delay. In the present context, the transistors are starting from 0.9 um, the minimum width and increases in multiples of the minimum width up to 7.2 um. By this arrangement, simulated results for the delay through the circuit could be varied from 0.2 ps to 2.3 ps which would be sufficient for the present application.

The enabling of the each of the set of delay cells to function as a ring oscillator is controlled by digital logic. An input *initial_enable* triggers off the digital logic which in turn sequentially enables four output signals ring_osc1_enable, ring_osc2_enable, ring_osc3_enable and ring_osc4_enable for a fixed amount of time (Refer appendix for verilog code). Subsequent pulses on the input again enable each of the oscillators sequentially one at a time. So whenever a calibration is required, a pulse needs to be applied to this input. The behavioral verilog code was tested for its operation and then synthesized using Synopsys. These digital signals close the loop around the particular ring oscillator. All the four outputs of the ring oscillators are fed to the asynchronous counter using a multiplexer.

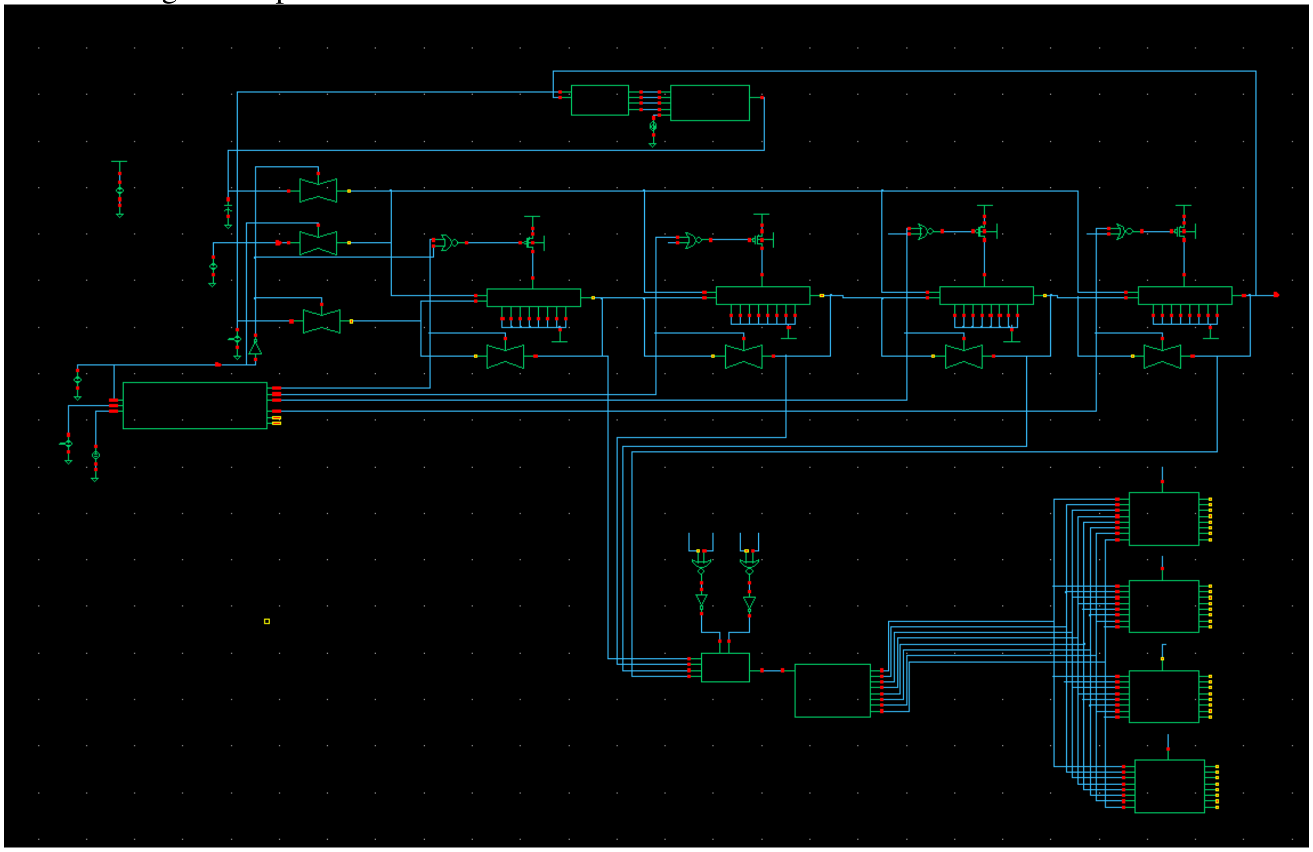

Fig 20: Schematic of the proposed DLL

However, since the output from one ring oscillator is connected to the next ring oscillator, the drive from one oscillator can affect the output of the next one. Hence, we disable the power supplied to the other ring oscillators when one is enabled. This is done by using a huge PMOS pass transistor to enable the power supply. The enabling condition is when the particular ring oscillator is intended to run and also during the normal operation of the DLL. The control voltage for the voltage controlled delay line (VCDL) was intended to be constant during the calibration process. Hence a constant control voltage was provided during this phase. This was achieved using transmission gates and a constant supply voltage of 2V which is planned to be made as an input pin. Figure 20 shows the full schematic of the proposed DLL with process variation calibration. For initial testing, all the binary inputs of the delay cell were given a high input.

However, there were issues with regard to the drive strength of the transistors and the ring oscillators do not function properly. Hence, the circuit was isolated with only the part required for the configuration of 2 of the ring oscillators taken into account and the sizes of the transistors were adjusted. Also, the feedback path closed by the transmission gate was not capable of driving the first inverter of the ring oscillator. Hence, two inverters were added in the feedback path. Figure 21 shows the reduced schematic and figure 22 shows the waveforms obtained from the same. It is to be noted that the second output displayed is being driven when either of them is enabled. However, it is being driven only upto half the supply rail when the second oscillator is enabled. Another important point to consider during this design was not to add any switches in between the ring oscillators as this directly affects the DLL operation in normal mode.

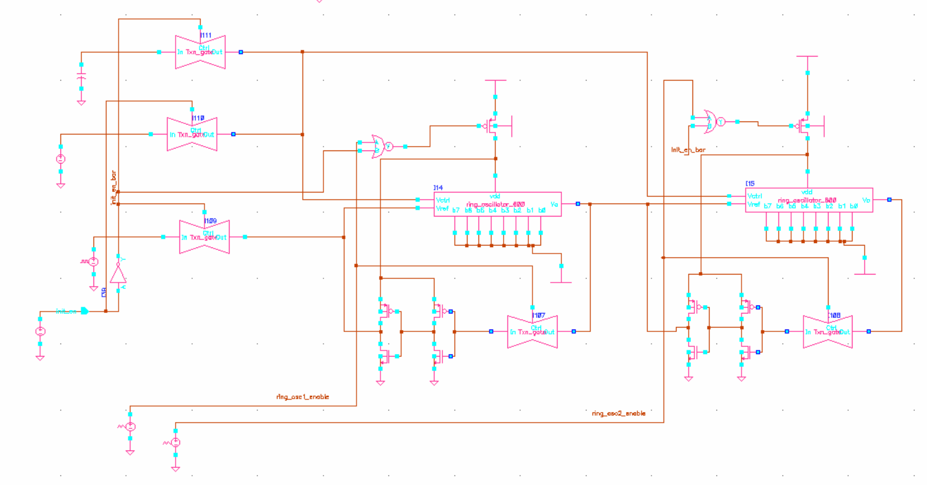

Figure 21: Reduced schematic of the proposed DLL

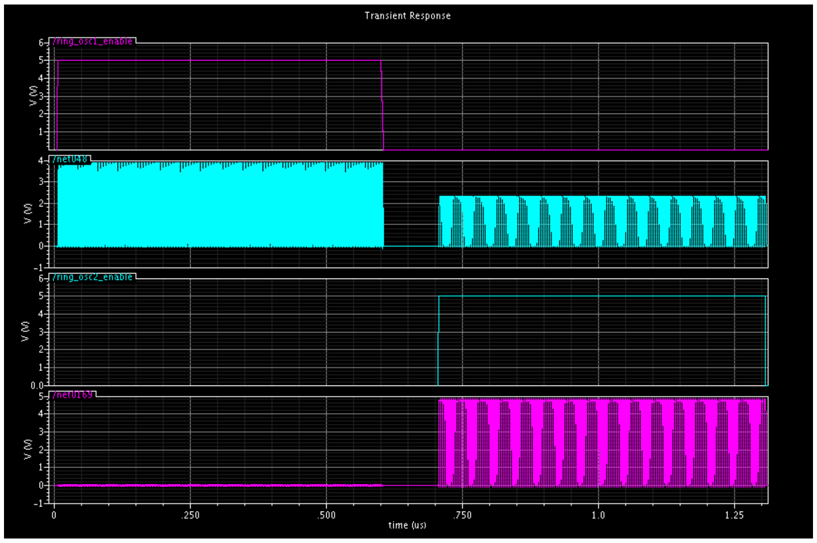

Fig 22: Ring oscillator outputs

The simulation of the whole system is to be done and there might be minor changes in the device sizes in order to operate the ring oscillators properly. The system level analysis of the DLL needs to done using Matlab. Together with this, to run the simulations faster for the whole system in Cadence, Verilog-A models for the phase detector and charge pump were used (refer Appendix).

IV. Fabrication and Testing

The DLL is planned to be fabricated in the AMI 0.6 micron technology over the summer. Following is a list of the system level inputs and outputs for the design:

Inputs:

V_{IN}: The input signal

initial_enable: A pulse signaling the start of the calibration of the delay cells.

V_{ctrl_in}: Bias voltage of 2v to operate the delay cells during the calibration.

Power supply: V_{dd} (+5V) and gnd (0V)

Outputs:

V_{out}: The output signal

Apart from this, we can tape out any intermediate signals for debugging mainly the output from each of the set of delay cells (oscillators during calibration), the count values etc.

For testing of the DLL, since the 0.6 um is not expected to have a high degree of process variation, some of the transistors can be given a body voltage in order to increase the threshold voltage. For the twin well process (0.6 um), only the body voltage of PMOS could be changed and hence we plan to make this is an input to our system.

References:

1) T.-C. Chen, “Where CMOS is going: Trendy hype versus real technology,” in IEEE

Int. Solid-State Circuits Conf. (ISSCC) 2006 Dig. Tech. Papers, Feb. 2006, pp. 22-28.

2) P.R.Kinget, “Device mismatch and tradeoffs in the design of analog circuits,” IEEE J. Solid-State Circuits, vol. 40, no. 6, pp. 1212-1224, Jun. 2005

3) J. Tschanz et al., “Adaptive Body Bias for Reducing Impacts of Die-to-Die and Within-Die Parameter Variations on Microprocessor Frequency and Leakage,” Proc. Int’l Solid-State Circuits Conf. (ISSCC 02), IEEE Press, Piscataway, N.J., 2002, pp. 422-423.

4) Anand M. Pappu, Xuan Zhang, Andre V. Harrison, and Alyssa B. Apsel, “Process-Invariant Current Source Design: Methodology and Examples.

5) George Chien and Paul R. Gray, " A 900-MHz Local Oscillator using a DLL-based Frequency Multiplier Technique for PCS Applications," IEEE Journal of Solid-State Circuits, Volume 35, Issue 12, Dec. 2000 Page(s):1996 – 1999

6) Low-jitter and process independent DLL and PLL based on self biasedtechniques Maneatis, J.G. Solid-State Circuits Conference, 1996. Digest of Technical Papers. 43rd ISSCC. 1996 IEEE International Volume, Issue, Feb 1996 Page(s):130 - 131, 430

7) D. A. Johns and K. Martin, “Analog Integrated Circuit Design”. NewYork: Wiley, 1997.

Appendix:

I) Verilog code for enabling the ring oscillators sequentially:

```
//Verilog HDL for "Research", "Controller" "behavioral"
```

```
module ring_osc_controller ( initial_enable, CLK, reset,
ring_osc1_enable, ring_osc2_enable,
    ring_osc3_enable, ring_osc4_enable,temp_enable, flag);


input initial_enable;
input CLK;
input reset;
output ring_osc1_enable;
output ring_osc2_enable;
output ring_osc3_enable;
output ring_osc4_enable;
output temp_enable;
output flag;

reg [7:0] osc_count_temp;


reg ring_osc1_enable;
reg ring_osc2_enable;
reg ring_osc3_enable;
reg ring_osc4_enable;
reg temp_enable;
reg flag;



always @(posedge initial_enable or posedge flag or posedge reset)
begin
    if (reset == 1)
      temp_enable = 0;
    else if (flag == 1)
      temp_enable = 0;
    else if (initial_enable == 0)
      temp_enable = 0;
    else if  (initial_enable == 1)
      temp_enable = 1;

end


always @(posedge CLK or posedge reset)
begin
      if (reset == 1)
            osc_count_temp <= 8'h00;
      else if(temp_enable == 1)
      begin
            osc_count_temp <= osc_count_temp + 8'h01;
            if(osc_count_temp == 8'h01)
                  ring_osc1_enable <= 1;
            else if(osc_count_temp == 8'h3D)
            begin
                  ring_osc1_enable <= 0;
                  ring_osc2_enable <= 1;
            end
            else if(osc_count_temp == 8'h79)
            begin
                  ring_osc2_enable <= 0;
                  ring_osc3_enable <= 1;
            end
            else if(osc_count_temp == 8'hB5)
            begin
                  ring_osc3_enable <= 0;
                  ring_osc4_enable <= 1;
            end
            else if(osc_count_temp == 8'hF1)
            begin
                  ring_osc4_enable <= 0;
                  flag = 1;
            end
      end
      else if(temp_enable == 0)
      begin
            flag = 0;
            ring_osc1_enable <= 0;
            ring_osc2_enable <= 0;
            ring_osc3_enable <= 0;
            ring_osc4_enable <= 0;
      end

end


endmodule
```

II) Verilog-A model for the phase detector:

```
// VerilogA for PFD_va, veriloga
//
// Implements a 3-state phase-frequency detector, outputs are active
low

`include "constants.h"
`include "discipline.h"

`define TIME_TOL 10e-12
`define VOLT_TOL 1e-6

module PFD_va(in1, in2, up_b, dn_b);
output up_b, dn_b;
input in1, in2;
electrical in1, in2, up_b, dn_b;

parameter real VOH = 1.80;
parameter real VOL = 0.00;
parameter real Vmid = 0.9;
parameter real tdel = 1e-10;
parameter real trise = 5e-11;
parameter real tfall = 5e-11;

integer state;
real vout, up_b_out, dn_b_out;
```

```
analog begin

   @(initial_step) begin
     state = 2;
     up_b_out = VOL;
     dn_b_out = VOL;
   end

   @(cross (V(in1)-Vmid, -1, `TIME_TOL, `VOLT_TOL))
   begin
      if ( state == 3 )
        state = 3;
      else
        state = state + 1;

      if ( state == 2 ) begin
        up_b_out = VOH;
        dn_b_out = VOH;
      end
      else begin
        up_b_out = VOL;
        dn_b_out = VOH;
      end
   end

   @(cross (V(in2)-Vmid, -1, `TIME_TOL, `VOLT_TOL))
   begin
      if (state == 1 )
        state = 1;
      else
        state = state - 1;

      if ( state == 2 ) begin
        up_b_out = VOH;
        dn_b_out = VOH;
      end
      else begin
        up_b_out = VOH;
        dn_b_out = VOL;
      end

   end

   V(up_b) <+ transition(up_b_out, tdel, trise, tfall);
   V(dn_b) <+ transition(dn_b_out, tdel, trise, tfall);

end
endmodule
```

III) Verilog-A model for the charge pump:

```
// VerilogA for EE536, CP_va, veriloga
//
// The up_b and dn_b inputs are active low
// The up and down currents are specified in Volts at the up_cur and
dn_cur
// inputs.  One volt represents 1 uA.

`include "constants.h"
`include "discipline.h"

module CP_va(up_b, dn_b, up_cur, dn_cur, out);
input up_b, dn_b, up_cur, dn_cur;
output out;
electrical up_b, dn_b, up_cur, dn_cur, out;

integer up_flag, dn_flag;
real up_current, dn_current;

analog begin
  @(initial_step) begin

    up_current = V(up_cur)*1e-6;
    dn_current = V(dn_cur)*1e-6;

    if (V(up_b)>0.9)
      up_flag = 0;
    else
      up_flag = 1;

    if (V(dn_b)>0.9)
      dn_flag = 0;
    else
      dn_flag = 1;
  end

  @(cross(V(up_b)-0.9, +1)) begin
    up_flag = 0;
  end

  @(cross(V(dn_b)-0.9, +1)) begin
    dn_flag = 0;
  end

  @(cross(V(up_b)-0.9, -1)) begin
    up_flag = 1;
  end

  @(cross(V(dn_b)-0.9, -1)) begin
    dn_flag = 1;
  end

  I(out) <+ - (up_flag * up_current) + (dn_flag * dn_current);
end

endmodule
```