

A Fast and Accurate SystemC / SystemC-AMS Model for Super-Regenerative Radio

THESIS

CAS-MS-2011-02

submitted in partial fulfillment of the
requirements for the degree of

Master of Science

In

Microelectronics

by

Kezheng Ma
Born in Beijing, China

This work was performed in:

Holst Centre / IMEC-NL
Eindhoven, the Netherlands

Circuits and Systems Group
Department of Microelectronics
Faculty of Electrical Engineering, Mathematics and Computer Science
Delft University of Technology

Delft University of Technology
Department of
Microelectronics

The undersigned hereby certify that they have read and recommend to the Faculty of Electrical Engineering, Mathematics and Computer Science for acceptance a thesis entitled “A fast and accurate SystemC / SystemC-AMS model for super-regenerative radio” by Kezheng Ma in partial fulfillment of the requirements for the degree of Master of Science.

Dated: 29/06/2011

Chairman: prof.dr.ir. A.J. van der Veen

Advisor: dr.ir. T.G.R.M. van Leuken

Committee Members: dr.ir.W.A.Serdijn
dr.L.Huang
ing.H.W.Pflug

Abstract

Nowadays, communication system has become a very important part of modern live. In order to dramatically reduce the cost and design time of a radio system, a unified design environment that can be used efficiently by system designers for modelling and simulation is very desirable. However, one of the main problems is that the simulation of a radio system is very time consuming or even impossible. Therefore, to examine the behaviour of a radio system, a fast and precise model is highly desirable. To address this problem, in this thesis, a new super-regenerative model based on the recently published SystemC/SystemC-AMS (BETA version) language is presented. The system is composed of both analog circuit model as well as the digital baseband which is impossible to model in the past. The simulation results are verified by both the simulation results of other model and the measurement results. The thesis shows the feasibility of a unified design environment for mixed signal modelling based on SystemC/SystemC-AMS in order to reduce the cost and design time of electrical systems.

Contents

| | | |
|--------|--|----|
| 1 | Introduction | 1 |
| 1.1 | Motivation | 1 |
| 1.2 | Thesis goals..... | 2 |
| 1.3 | Contributions..... | 2 |
| 1.4 | Thesis organization | 3 |
| 2 | Background..... | 4 |
| 2.1 | Introduction to SystemC/SystemC-AMS | 4 |
| 2.1.1 | SystemC..... | 4 |
| 2.1.2 | SystemC-AMS..... | 5 |
| 2.2 | Introduction to super-regenerative radio | 6 |
| 2.3 | Introduction to PPM modulation | 9 |
| 3 | Analog parts modelling of the super regenerative radio | 10 |
| 3.1 | The mixer block | 10 |
| 3.2 | The power amplifier | 11 |
| 3.3 | The super block | 11 |
| 3.4 | The quench block | 12 |
| 3.5 | The envelop detector..... | 14 |
| 3.6 | Low-pass filter..... | 15 |
| 3.7 | The VGA..... | 15 |
| 3.8 | ADC..... | 15 |
| 4 | Phase Locked Loop (PLL) modelling | 17 |
| 4.1 | Frequency analysis of the PLL | 17 |
| 4.2 | The SystemC/SystemC-AMS model | 17 |
| 4.2.1 | Reference clock block | 18 |
| 4.2.2 | The PFD block..... | 19 |
| 4.2.3 | The rfdelay block | 20 |
| 4.2.4 | Inverter | 21 |
| 4.2.5 | Charge pump and loop filter | 21 |
| 4.2.6 | Control_m block | 23 |
| 4.2.7 | VCO | 23 |
| 4.2.8 | Converter..... | 24 |
| 4.2.9 | Divider | 24 |
| 4.2.10 | Sigma-Delta Modulator..... | 24 |
| 5 | Frequency Locked Loop (FLL) modelling..... | 27 |
| 5.1 | Phase-Alied FLL (PA-FLL) | 27 |
| 5.1.1 | Structure of PA-FLL..... | 27 |

| | | |
|-------|--|----|
| 5.2 | Type-2 FLL | 30 |
| 5.2.1 | The structure of the FLL | 30 |
| 6 | The digital baseband of the model | 33 |
| 6.1 | The transmitter SystemC/SystemC-AMS model | 33 |
| 6.1.1 | Transmitter data link layer | 33 |
| 6.1.2 | Transmitter physical layer | 34 |
| 6.2 | The receiver SystemC/SystemC-AMS model | 36 |
| 6.2.1 | The buffer | 36 |
| 6.2.2 | The threshold estimation | 38 |
| 6.2.3 | The Detection | 38 |
| 6.2.4 | The confirmation stage | 39 |
| 6.2.5 | The timing stage | 39 |
| 6.2.6 | The SFD detection | 40 |
| 6.2.7 | Data collection | 41 |
| 7 | Simulation results | 42 |
| 7.1 | The simulation results of PLL and FLL | 42 |
| 7.1.1 | Comparison with Matlab simulation results | 42 |
| 7.1.2 | Comparison with ADI SimPLL simulation results | 43 |
| 7.1.3 | Comparison with Cadence simulation results | 44 |
| 7.1.4 | Comparison with measurement results | 45 |
| 7.1.5 | Simulation results for sigma-delta modulator | 46 |
| 7.1.6 | Simulation results for the AC domain analysis of the PLL | 49 |
| 7.1.7 | Simulation results for PA-FLL | 53 |
| 7.1.8 | Simulation results for FLL type-2 | 53 |
| 7.2 | Simulation results for the Analog System | 54 |
| 7.2.1 | The simulation results for PLLs | 55 |
| 7.2.2 | The simulation results of the transmitter | 55 |
| 7.2.3 | The simulation results of the receiver | 55 |
| 7.2.4 | Simulation results for critical current searching | 57 |
| 7.2.5 | Simulation results for optimal slope searching | 57 |
| 7.3 | Simulation results for digital baseband | 57 |
| 7.3.1 | The transmitter simulation results | 57 |
| 7.3.2 | Threshold estimation | 57 |
| 7.3.3 | Simulation results for detection | 58 |
| 7.3.4 | Simulation results for confirmation | 58 |
| 7.3.5 | Simulation results for timing | 59 |
| 7.3.6 | Simulation results for SFD detection | 59 |
| 7.3.7 | Simulation results of data collection | 60 |

| | | |
|--------|---|----|
| 7.3.8 | Data verified | 61 |
| 7.3.9 | Compare with measurement results | 61 |
| 7.3.10 | Improvement of the PPM modulation | 63 |
| 8 | Conclusion | 64 |
| 9 | References..... | 65 |

List of abbreviation

| | |
|--------|--|
| PLL | Phase Locked Loop |
| FLL | Frequency Locked Loop |
| RTL | Register Transfer Level |
| TDF | Timed Data Flow model |
| LSF | Linear Signal Flow Model |
| ELN | Electrical Linear Networks Model |
| RF | Radio Frequency |
| LNA | Low Noise Amplifier |
| PA | Power Amplifier |
| ED | Envelop Detector |
| LPF | Low-pass Filter |
| VGA | Variance Gain Amplifier |
| ADC | Analog to Digital Converter |
| DAC | Digital to Analog Converter |
| VCO | Voltage Control Oscillator |
| PFD | Phase/Frequency Detector |
| LF | Loop Filter |
| CP | Charge Pump |
| PA-FLL | Phase-Aliened Frequency Locked Loop |
| ELD | Early Late Detector |
| DL | Decision Logic |
| DCO | Digital Control Oscillator |
| FVC | Frequency to Voltage Converter |
| PPM | Pulse Position Modulation |
| OOK | On-Off-Keyed |
| PSD | Power Spectrum Density |
| VHDL | Very High Speed Integrated Circuit Hardware Description Language |
| SNR | Signal to Noise Ratio |

1 Introduction

1.1 Motivation

In recent years, the super regenerative circuit is re-examined due to its simplicity and power efficiency. It is widely used in short range communication system for robotics, sensor networks and home automation. However, a number of difficulties make the simulation for the radio system quite hard and to examine the behaviour of a super-regenerative radio system, a fast and precise model is highly desirable.

The first problem is the difficulty of including PLL into the analog system due to its high complexity. Therefore, the distortion caused by the frequency differences between the transmitter and receiver (which is important in many applications) cannot be simulated.

Another problem is the accurate and efficient whole radio system modelling, which includes both the analog circuits modelling as well as the digital baseband modelling. As a result, the designers can only evaluate the behaviour of the analog circuits but not the performance of the whole system.

SystemC/SystemC-AMS is a new published effective system-level simulation and modelling language, which offers many advantages which is able to solve the problems mentioned above. First of all, SystemC/SystemC-AMS is a C++ based language which ensures its calculation speed. With the help of fast simulation speed, the simulation time of the system is dramatically reduced. Therefore, the PLL is able to be included into the analog system to simulate the frequency difference between the transmitter and receiver which can be used to calculate the distortion caused by the difference. In addition, with the help of SystemC/SystemC-AMS, it is able to simulate the digital and analog circuits simultaneously. The analog parts use time driven methods while the digital parts use event driven methods. With this capability, it is very convenient to include the digital baseband to the radio system to evaluate the whole performance of the system (such as bit error rate).

In this thesis, a new mixed signal super-regenerative radio model based on the recently published powerful language SystemC/SystemC-AMS (BETA version) is proposed. In this model, both the analog circuits and the digital circuits are included. The analog part include analog blocks such as Mixer, Power Amplifier, PLL, Envelop Detector, Low-pass Filter, VGA, ADC and so on. Also, many imperfections such as high order harmonics, VCO noise, Charge Pump current mismatch, leakage current, distortion caused by the frequency differences between the transmitter and receiver are included. In the digital system, both the digital baseband for transmitter and receiver are included. In the transmitter part, the data is first encoded according to the PPM algorithm and then transmitted to the analog environment. In the receiver part, the synchronization algorithm for PPM is implemented and the data is decoded and compare with the transmitted data to calculate the bit error rate. Figure 1 shows the structure of the considered whole system, which is divided into 3 parts: the transmitter digital part, the complete analog link and the receiver digital part. The transmitter digital part is used to generate the digital baseband signal to be transmitted. Then, the digital signal will be transferred through the analog part of the radio including the channel. Finally, the receiver digital part is used to recover the data from the output signal of the analog link.

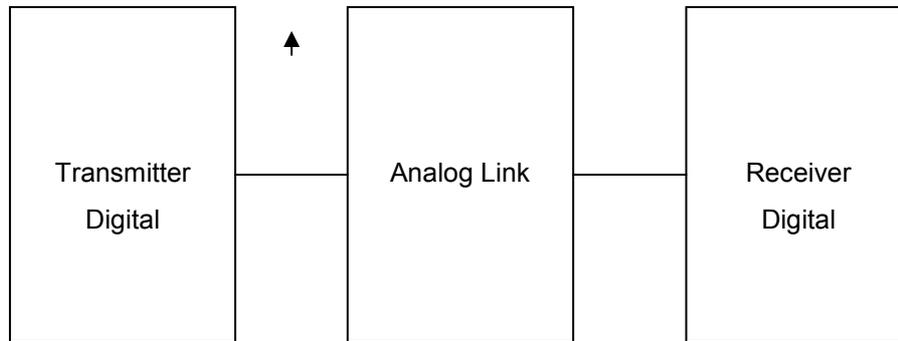


Figure 1 The structure of the whole system.

As mentioned above, with the help of SystemC/SystemC-AMS, it is possible to simulate the whole radio system to evaluate the system performance which is not easy in the past. Also, by comparing its simulation results with the measurement results and the simulation results of the other models, the accuracy of the model and the functionality of the SystemC-AMS are validated.

1.2 Thesis goals

This thesis describes a super-regenerative radio model based on the recently published powerful language SystemC/SystemC-AMS (BETA version). In the model, the performance of the whole system is able to simulate. The primary objectives of this work are to:

- Make a fast and accurate PLL model
- Make a model for super-regenerative radio analog circuits
- Include the PLL model into the analog system of a super-regenerative radio
- Make a digital baseband model with SystemC
- Combine the analog system and digital system to simulate the whole performance of the radio system
- Verify the functionality of the model new published language SystemC-AMS

1.3 Contributions

The main contributions of this thesis are shown below:

- Make fast and precise PLL and FLL models

- Make the model to describe the behaviour of the super-regenerative receiver
- Propose a new synchronization algorithm for the PPM modulation to improve system performance
- Implemented the digital baseband algorithms for PPM with SystemC
- Implemented the digital baseband algorithms for PPM with VHDL on FPGA

1.4 Thesis organization

The thesis is organized into the following chapters:

In Chapter 2, there is a brief introduction to SystemC/SystemC-AMS and basic concepts of the super-regenerative radio system. The discussion of the advantages of the SystemC/SystemC-AMS is shown. Also the basic ideas of the PPM modulation are discussed.

Chapter 3 provides an overview of the SystemC/SystemC-AMS model for the analog system of the radio. The organization and key components of the radio system are shown.

Chapter 4 shows the modelling of the PLL used in the analog system. Except the basic functions of the PLL, many imperfections are included in the model proposed in this paper, e.g. VCO noise, PLL Dead Zone, Phase Frequency Detector (PFD) reset delay, Charge Pump (CP) current mismatch, CP leakage current, the reference jitter and Delay difference between the CP control signal.

Chapter 5 shows the FLL SystemC/SystemC-AMS model. Like PLL, many imperfections are included too.

Chapter 6 presents the digital baseband modelling. The digital baseband can be divided into 2 parts, the transmitter part and the receiver part. The transmitter part is used for the PPM modulation and the receiver part is used to realize a synchronization algorithm and verify the corrected data.

Chapter 7 provides the simulation results of the system. The results are verified by both the simulation results of other model and the measurement results.

Chapter 8 includes concluding remarks, and recommendations for future work.

2 Background

This chapter provides an overview of the basic concepts of this thesis. In the first part, SystemC/SystemC-AMS and its advantages are discussed and in the second part, the basic ideas about super-regenerative radio are shown. In the third part, a modulation technology called PPM modulation is presented.

2.1 Introduction to SystemC/SystemC-AMS

SystemC/SystemC-AMS is an effective system-level simulation and modelling language. SystemC-AMS is a new published language. As a result, not much work has been done to verify it. One purpose of this thesis is to verify the correctness of this language. SystemC/SystemC-AMS has many advantages. First of all, it is a C++ based language which ensures fast calculation speed. Moreover, it can take advantage of C++ power by using a large number of existing C++ functions and libraries. Secondly, SystemC/SystemC-AMS support hardware and software co-design which is very important to the design of a very complex system. Last but not least, SystemC/SystemC-AMS is an open source language, which requires no license fee for the usages.

2.1.1 SystemC

SystemC is a C++ based language. It adds a class library to C++ which enables it to describe hardware. SystemC is able to describe the concepts that are familiar with hardware designer, such as signals, modules and ports.

SystemC does not add any new syntax to C++. Instead, it only adds a library class to describe the concepts in hardware design. Thus, SystemC is essentially C++. For this reason, the designers can conveniently use the standard C++ developing tools to simulate, debug and execute all kinds of algorithms and structures. More importantly, with SystemC, we can make a hardware model in system level as well as RTL level. We can first make a system level model to simulate and optimize the design quickly. Then we can transfer the design to RTL level for synthesis.

SystemC has many advantages. The first one is its fast simulation speed. Nowadays, system designs become more and more complex. As a result, a fast simulation speed is strongly preferred. As shown in Figure 2, we can see that the iteration time is the highest in chip level and fastest in system level. Thus, to make simulation faster, it is preferred to simulate design in system level. Besides, expressing the design in system level makes the exploration of algorithms and structures easy and quick.

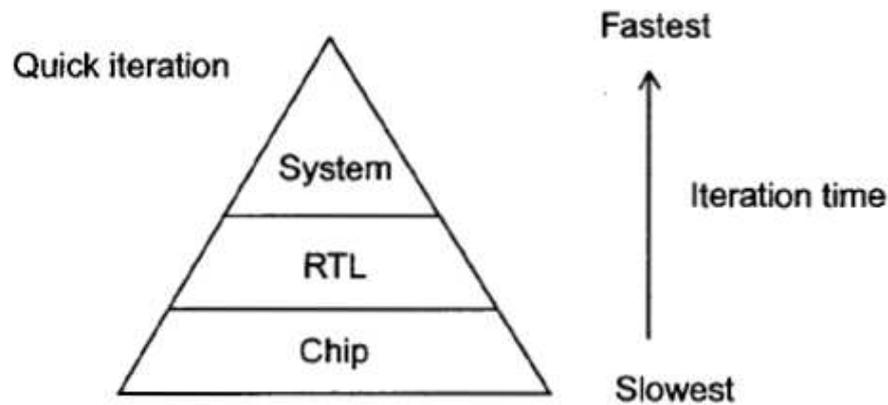


Figure 2 Iteration time of design level [1].

The second advantage of SystemC is the capability of hardware/software co-design. Because of the growing complexity in both the hardware and software domains, partitioning of functions in which domain is desired. This partition could be easily implemented in SystemC since it is able to describe the function either in hardware domain or software domains.

Last but not least, SystemC is an open-source language. There is no license fee to use it.

As discussed above, SystemC is very suitable for modelling the digital circuits. Its main limitation is the disability of simulating analog or mixed signals. For this reason, SystemC-AMS is developed as a complement to deal with this inconvenience.

2.1.2 SystemC-AMS

In order to simulate the analog circuits, AMS extensions based on SystemC are introduced. The SystemC-AMS extensions are built on top of the SystemC language standard IEEE 1666-2005. They define additional language constructs, which introduce new execution semantics and system-level modelling methodologies to design and verify mixed-signal systems [2]. They can be further divided into 3 models -- Timed Data Flow model (TDF), Linear Signal Flow model (LSF), Electrical Linear Networks model (ELN).

2.1.2.1 TDF model

As shown in Figure 3, the TDF model is a discrete-time model. It considers the data as sampled signals in time. Each TDF model is composed of a number of TDF modules which are connected with TDF signals and TDF ports. In each module, C++ methods are used to describe a function that relates the inputs and outputs.

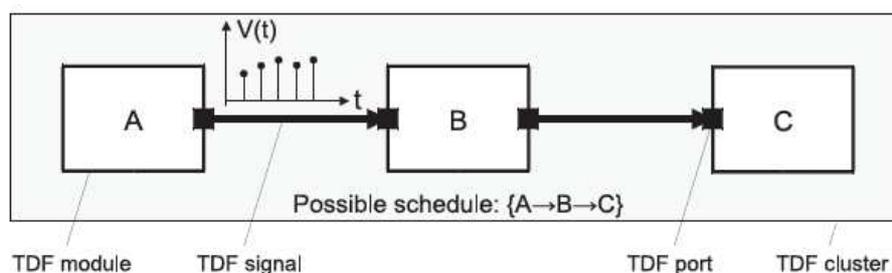


Figure 3 A TDF model with 3 TDF modules [2].

2.1.2.2 LSF model

The LSF model is a continuous-time module. This model provides a finite number of LSF modules to implement calculation such as addition, subtraction and multiplication. The users are not allowed to define their own blocks. In other words, LSF model can only be composed with the provided modules. LSF signals are used to connect the modules. In this way, the equation system will be formed to relate the inputs and outputs. Figure 4 shows a basic LSF model.

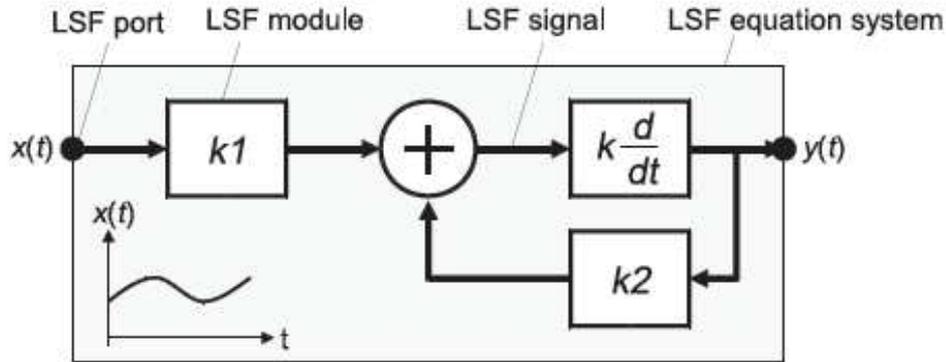


Figure 4 A LSF model with 4 LSF modules [2].

2.1.2.3 ELN model

As shown in Figure 5, in ELN model, a set of electrical primitives such as sources (voltage or current), linear lumped elements (resistors, capacitors, inductors) are provided. Designers model the circuit by connecting these primitives with electrical nodes. The system will follow the Kirchoff's current and voltage laws. ELN terminals are used to communicate with other models.

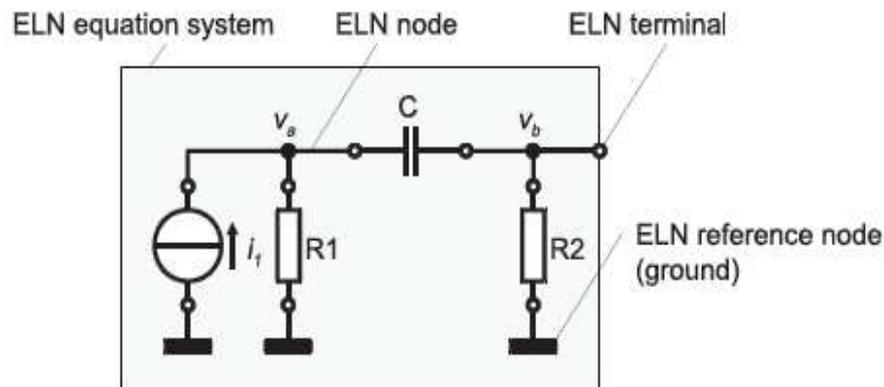


Figure 5 A basic ELN model [2].

2.2 Introduction to super-regenerative radio

The super-regenerative circuit is able to amplify an electrical signal many times by the same vacuum tube or other active component such as a field effect transistor. The circuit has 2 main parts, an amplifier and a feedback loop. The feedback loop connects the output and the input of the amplifier to provide a positive feedback. The idea of the super-regenerative circuit is proposed by Armstrong about 90 years ago. However, it did not attract the attention of engineer for a long period.

In recent years, the super regenerative circuit is re-examined due to its simplicity and power efficiency. It is widely used in short range communication system for robotics, sensor networks and home automation. These applications do not require high data rate but require longer lifetime. As a result, the receiver should be turned off as long as possible. The super-regenerative circuits are very suitable for these applications.

The basic idea of the super-regenerative radio is as follows. The oscillator starts to oscillate when the quenching signal is on and decay when the quenching signal is off. When the quenching signal is on, there are 2 cases. In the first case, there is no input, and the oscillator start to oscillate very slow. As a result, the peak value of the oscillation is small. In the second case, the input signal is not zero, in this case, the oscillator start to oscillate quicker than that with a 0 input and the peak value is big. This difference between the peak values can be used to determine if the data transmitted is 0 or 1. If the transmitter transmits '1', the output of the receiver oscillator will be large because it begins to oscillate earlier. On the contrary, if the data is '0', the output of the receiver oscillator is small.

Figure 6 shows the structure of a super-regenerative receiver. The Quench Oscillator is used to generate the quenching signals to control the rise and die out of the RF oscillator. The RF oscillator can be modelled by a selective network and an amplifier. The amplifier provides a feedback from the output of the selective network to the input of the network. The gain of the amplifier is controlled by the quenching signals which make the system alternatively stable or unstable. The work modes of the receiver can be divided into 2 modes, the first is the linear mode and the second one is logarithmic mode. In the linear mode, the oscillation amplitude is not allowed to reach its maximum value. The peak value of the oscillation is proportional to the input signal. In the logarithmic mode, the peak value of the oscillation is the maximum value and the incremental area under the envelope is proportional to the logarithm of the amplitude of the input signal. The system in this thesis use linear mode.

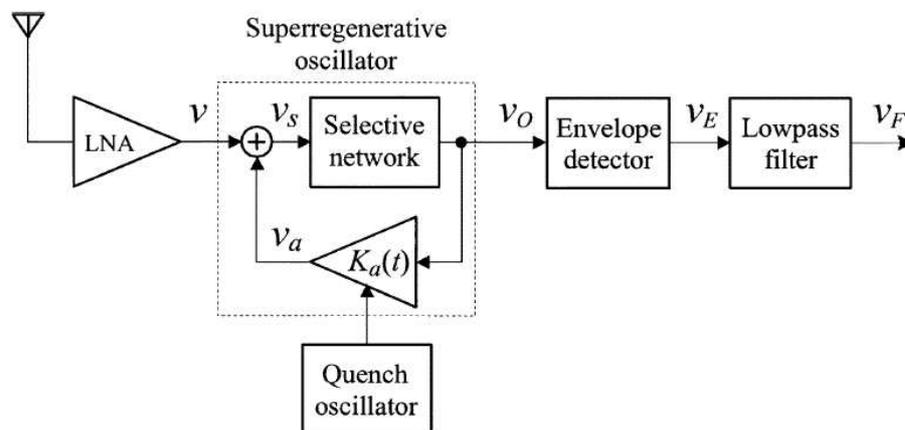


Figure 6 structure of super-regenerative receiver [4].

The super-regenerative oscillator is modelled as a selective network with a feedback amplifier. We will assume that the selective network has two dominant poles providing a band-pass response centred on ω_0 [4]. The transfer function is shown below (in s-domain):

$$G(s) = K_0 \frac{2\zeta_0\omega_0 s}{s^2 + 2\zeta_0\omega_0 s + \omega^2}$$

Or, equivalently:

$$\ddot{v}_O(t) + 2\zeta_0\omega_0 \dot{v}_O(t) + \omega^2 v_O(t) = K_0 2\zeta_0\omega_0 \dot{v}_s(t)$$

Where ζ_0 is quiescent damping factor and K_0 is the maximum amplification. From Figure 6, we can see that:

$$v_s(t) = v(t) + K_a(t)v_O(t)$$

As a result, the general form of the differential equation of the super-regenerative receiver is:

$$\ddot{v}_O(t) + 2\zeta(t)\omega_0 \dot{v}_O(t) + \omega^2 v_O(t) = K_0 2\zeta_0\omega_0 \dot{v}(t)$$

Where $\zeta(t)$ is equal to $\zeta_0 - 2\zeta_0 K_a(t) K_0$. By solving the above equation, we can get the expression of the output of the oscillator.

Another method is to model the LNA and oscillator as a parallel resonant circuit which is shown in Figure 7. In the figure, the L represents the load inductor and C represents the capacitor load, the G is the total conductance that is a sum of the parasitic losses of the resonant LC tank in the oscillator (G_0), and the negative conductance provided by the active devices (transistors) in the oscillator ($-G_a(t)$). The injected signal is the output current of the LNA. It is possible to obtain the equivalence between the parameters of the block diagram (Figure 6) and those of a particular circuit. The equivalence is shown in Figure 8 for parallel RLC circuits.

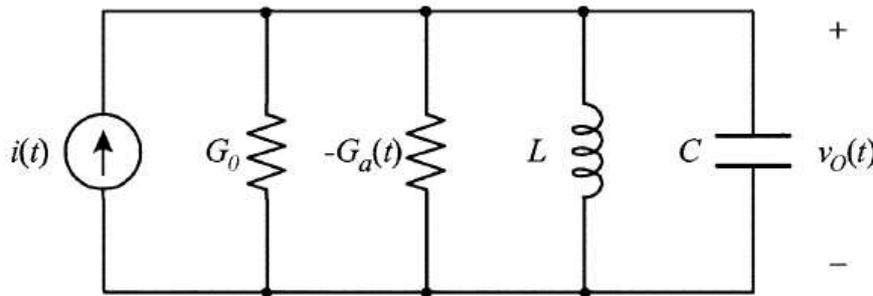


Figure 7 Parallel resonant circuit represented the LNA and oscillator [4].

| Block diagram | v | v_O | K_0 | ζ_0 | ω_0 | $K_a(t)$ |
|---------------|-----|-------|-----------------|--------------------------|-----------------------|----------|
| Circuit | i | v_O | $\frac{1}{G_0}$ | $\frac{G_0}{2C\omega_0}$ | $\frac{1}{\sqrt{LC}}$ | $G_a(t)$ |

Figure 8 The equivalence parameters for parallel RLC circuits [4].

2.3 Introduction to PPM modulation

In this thesis, the PPM modulation is used to modulate the data. Also a new synchronization algorithm for PPM modulation is presented and implemented with VHDL. In this section, a brief introduction to PPM modulation is shown.

The basic idea of PPM modulation is as follow. Every "x" bits data is translated into 2^x chips, and there is only one "1" among these chips, the position of this chip is decided by the value of the "x" bits. Figure 9 shows a 2 to 4 PPM modulation. From the figure, we can see that the position of the "1" is decided by the value of the 2 bits of the source data. In this thesis, the 4 to 16 PPM modulation is used.

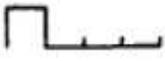
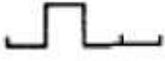
| Source data | 4-PPM |
|-------------|---|
| 00 |  |
| 01 |  |
| 10 |  |
| 11 |  |

Figure 9 2 to 4 PPM modulation

3 Analog parts modelling of the super regenerative radio

In this section, the analog system modelling of the super-regenerative radio system is presented. The structure of the super-regenerative radio is shown in Figure 10. Because the PLL is very important as well as very complex, the discussion of it will be left to the next section, in this part, we will talk about the SystemC/SystemC-AMS model of all the other analog blocks.

As shown in Figure 10. The radio model is composed of twelve blocks, a data block, a mixer block, two PLL blocks, a Power Amplifier (PA) block, a channel block, a super block, a quench block, an Envelope Detector (ED) block, a Low-Pass Filter (LPF) block, a Variance Gain Amplifier (VGA) block and an Analog to Digital Converter (ADC) block. The data block is used to generate the data (replaced by digital baseband later). The mixer block is used to mix the data and the carrier that generate by the PLL in transmitter side. Then the mixed signal is sent to PA to amplify the power. The channel is used to model the transmission channel. The super block is a combination of Low Noise Amplifier (LNA) and receiver oscillator. The oscillator current is provided by Quench block and the oscillation frequency is tuned by the receiver PLL. The Envelop Detector and Low-Pass Filter is used to recover the low frequency baseband signal. The VGA is used to generate the signal that fit the ADC input and the ADC will generate the digital signals that can be processed by the digital baseband. The Quench block is modelled with SystemC and all other blocks (except PLLs) are modelled with SystemC-AMS. Most of them are modelled with TDF model because with TDF model, it is easier to model the non-ideal effect such as harmonies. Also, compare to ELN model, the TDF model is more flexible and can achieve all kinds of behaviour which is especially important for the modelling of some of the blocks such as Super block.

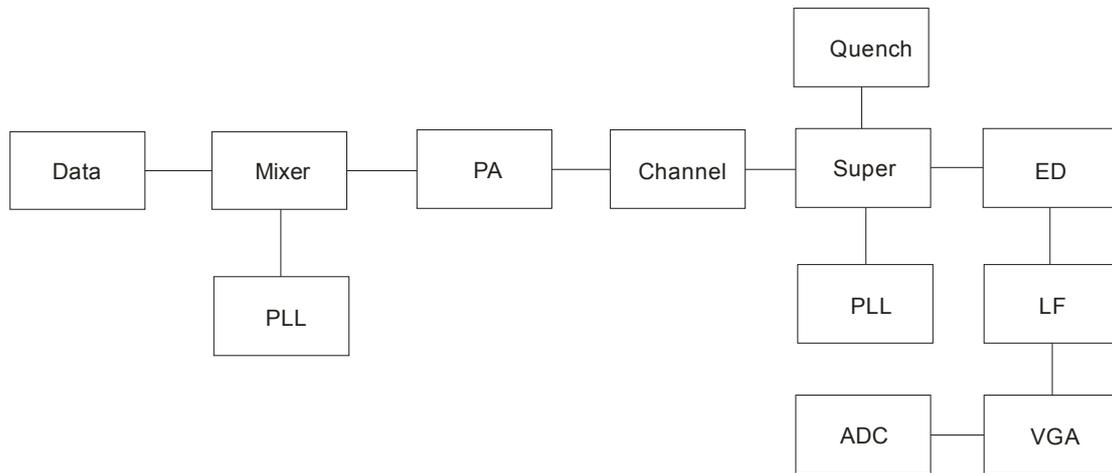


Figure 10 Structure of the super regenerative radio.

3.1 The mixer block

The mixer block is modelled as a multiplier and it is modelled with TDF model. It acts as follows: if the data is equal to 1, then the output of the mixer is equal to the PLL output; if the data is equal to 0, the output of the mixer is 0. Part of the source codes of mixers is shown in Figure 11.

```

SCA_TDF_MODULE(mixer) {
sca_tdf::sca_in<bool> in_bit;
sca_tdf::sca_in<double> in_wave;
sca_tdf::sca_out<double> out;

void processing() {
if (data==1)
{out=carrier;}
else
{out=0;}
}
SCA_CTOR(mixer) {}
};
    
```

Figure 11 Part of source code of mixer.

3.2 The power amplifier

The model of the PA includes the multiplier, and the high-order harmonics. It is modelled with TDF model. Part of the source code of PA is shown in Figure 12. In the code, the variable “in” is the input of the power amplifier and “out” is the output. The variable “a”, “b”, “c” are the gain for each harmonics.

```

out=a*in+b*in*in+c*in*in*in;
    
```

Figure 12 Part of the source code for PA

3.3 The super block

The super block is the main part to perform the function of super regenerative receiver and it is modelled with TDF model. It is the combination of the LNA and the receiver oscillator. The oscillator current is provided by quench block. By solving the equation mentioned in section 2.2, we can get the solution of the output signal which is equally to:

$$V_o(t) = \frac{A \omega_r e^{(-\frac{G \times t}{2 \times C_L})}}{\beta \times |G|} \sin(\beta t) + \frac{A \sin(\omega_r t)}{|G|}$$

in which $\beta = \sqrt{\frac{G^2}{4C_L^2} - \frac{1}{L_L C_L}}$, and $\omega_r = 2\pi f_r$ where “fr” is the resonant frequency [5], the “A” is the amplitude of the input signal, L_L is load inductor, C_L is load capacitor and G is the total conductance.

The solution mentioned above does not take the frequency difference between the transmitter and the receiver into account. As a result, it is impossible to examine the PLL performance. To examine the PLL performance, the distortion caused by the different between the transmitter frequency and receiver frequency is modelled. The method is shown below:

$$s(t) = e^{\omega_0 \int_0^t \zeta(\lambda) d\lambda}$$

$$\psi(\omega) = \int_{-\infty}^{\infty} p_c(t)s(t)e^{j\omega t} dt$$

$$H(\omega) = \frac{\omega \psi(\omega - \omega_0)}{\omega_0 \psi(0)}$$

In the equation, the $\zeta(\lambda)$ is the quenching signals, ω is the transmitter frequency and ω_0 is the receiver centre frequency. “ $p_c(t)$ ” is the pulse shaping signal and here it is assume to be one. The distortion caused by the frequency difference between the transmitter and receiver is the absolute value of $H(\omega)$ and the phase shift is equal to the phase of $H(\omega)$.

3.4 The quench block

The quench block is used to generate the quenching current that is needed by the receiver oscillator and it is modelled with SystemC. The reason why I use SystemC rather than SystemC-AMS is that in the quenching block, it is needed to generate the period signal which is shown in Figure 13. This is convenience to generate with “wait” statement which is available in SystemC rather than SystemC-AMS. Figure 13 shows the quenching current that generated by this block, in the figure, the dashed line is the value of the critical current. In the Q-enhanced mode, the provided current is a little below the critical current, in the super-regenerative mode, the current will increase exponentially with time and during the off mode, the current will be 0. The code to generate the quenching current is shown in Figure 14. In the code, the 0.01ns is the resolution. The time for super-regenerative mode is equal to 0.01ns * 1000 = 10ns.

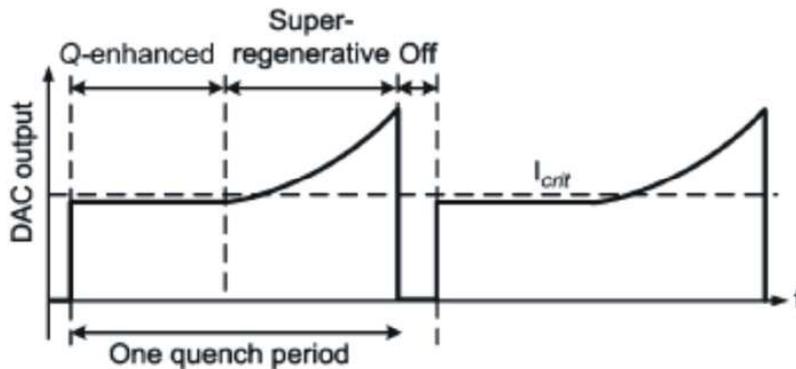


Figure 13 Current provided by the quenching block[5].

```

SC_MODULE (qunch) {
sc_out<double> controls,c,slp;
sc_out<bool> rpllc,adcen,qbegin;
sc_in<double> adco;

void prc_qunch();

SC_CTOR(qunch) {
SC_THREAD(prc_qunch);
}

void qunch::prc_qunch() {
while(1) {
controls=cr; // "cr" is the critical current searched by the system
wait(Q_time,SC_US); //Q-enhanced mode
for(int i=0;i<1000;i++)//super-regenerative mode
{
controls=cr*exp(slope*x);
wait(0.01,SC_NS);
t=t+0.01e-9; //0.01e-9 is the resolution
}
t=0;
controls=0; // Off
wait(off_time,SC_US);
}
}
}

```

Figure 14 Code for quenching current generation.

Besides providing the quenching current, the quenching block also includes the critical current searching and optimum analog slope searching algorithms. In practical, the critical current is not known, as a result, before the operation start, the system needs to find the critical current autonomously. To address this issue, we consider a critical current searching algorithm, which will be described in detail here. In addition, to gain the best performance, the analog slope of the quenching signal need to be optimized. Thus, an analog slope searching algorithm will also be discussed here.

The basic idea of critical current searching algorithm is as follows. A DAC is used to control the quenching current of the VCO, the higher the DAC value, the higher the quenching current. The input of the receiver is set to 0. We change the DAC value, if the VCO oscillate, then decrease DAC value, if not, increase the DAC value. The critical current of the VCO will be found after several iterations (number of DAC bits). Figure 15 shows the steps of this algorithm. In the algorithm, the quench block will first set the highest bit of DAC to 1 and other bits to 0, then, it will estimate the output power of the VCO. If the power is larger than a threshold which means the VCO oscillated, it will set bit n to 0 and bit n-1 to 1. If the power is smaller than the threshold, bit n will remain to 1 and n-1 is set to 1 too. This will iterate for N times (N is the DAC bit number). Then the critical current will be found.

The optimum analog slope searching algorithm is nearly the same with critical current searching except that the DAC is used to control the analog slope rather than the quenching current.

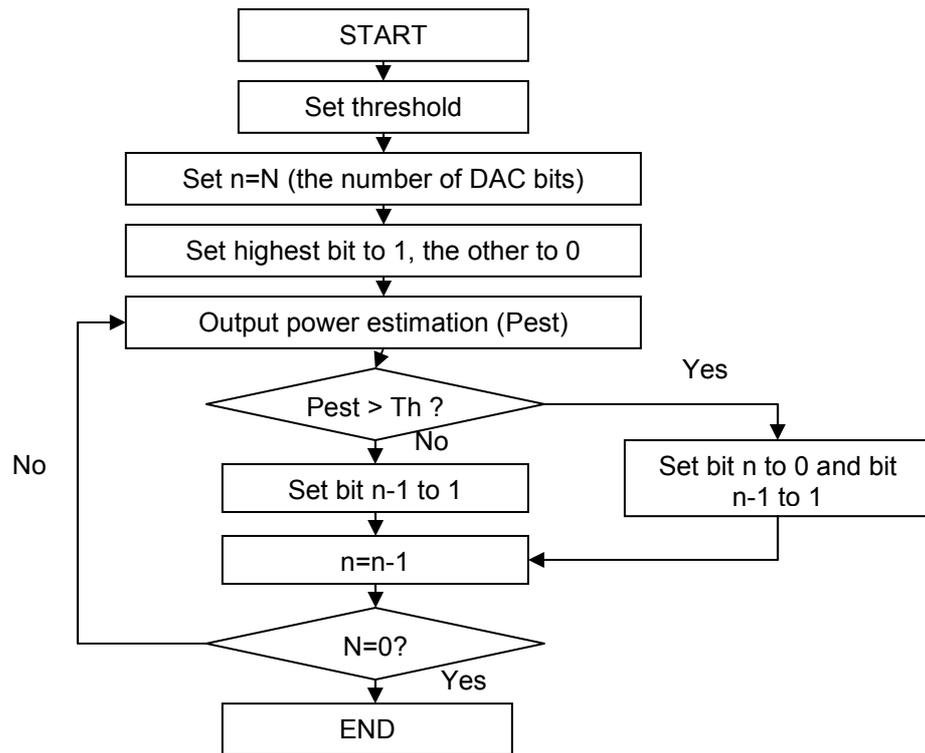


Figure 15 Critical current searching.

3.5 The envelop detector

The structure of an envelope detector is shown in Figure 16. When the input wave's amplitude increases, the capacitor voltage is increased via the rectifying diode. When the input's amplitude falls, the capacitor voltage is reduced by being discharged by a resistor, R. In this way, the "envelope" of the input signal will be detected. The modelling of this block is composed of 2 parts. The first part is the diode which is modelled with TDF model and the second part is the parallel capacitor and resistor which is modelled with ELN model.

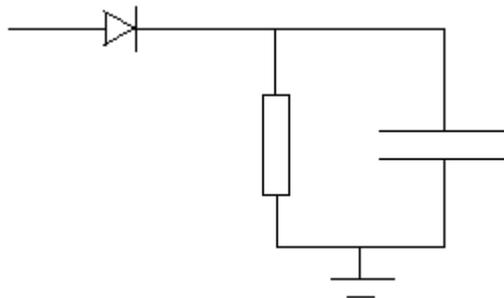


Figure 16 The structure of envelop detector.

3.6 Low-pass filter

The low pass filter is modelled as a basic RC filter with the TDF model and the Laplace transfer function. It is also able to be modelled with ELN model. But the results are same. The structure of the low pass filter is shown in Figure 17. Part of the source code for low pass filter is shown in Figure 18. In the code, the `ltf_1` is the Laplace transfer function which is used to form the low pass characteristic of the filter. The “`freq_cut`” is the cut off frequency.

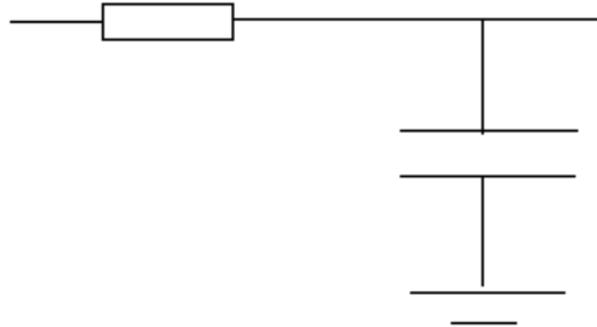


Figure 17 Structure of low-pass filter

```
SCA_TDF_MODULE(lowpass) {
sca_tdf::sca_in<double> in;
sca_tdf::sca_out<double> out;
sca_tdf::sca_ltf_nd ltf_1;
double freq_cutoff;
sca_util::sca_vector<double> Nom, Denom;

void processing() {
out.write(ltf_1(Nom,Denom, in.read()));
}

lowpass(sc_module_name n, double freq_cut){
Nom(0)= 1.0; Denom(0)=1.0;
Denom(1)= 1.0/(2.0*M_PI*freq_cut);
}
};
```

Figure 18 Part of the source code of the low pass filter.

3.7 The VGA

The VGA is modelled as a multiplier that used to adjust the signal amplitude to the ADC requirement. It is modelled with TDF model.

3.8 ADC

The ADC is used to transfer the analog signal to the digital signal that can be processed by the digital baseband. It is modelled with TDF model. The modelling method is shown in Figure 19, it acts as follows: first set the i to the number of ADC bits N , then compare the input value to 2^i , if

the input is larger, bit i will be set to 1, else, it is set to zero. Then subtract $bit(i) \times 2^i$ is subtracted from the input value and goes into the next iteration. The algorithm will iterate N times and then the output will be sampled by the digital baseband.

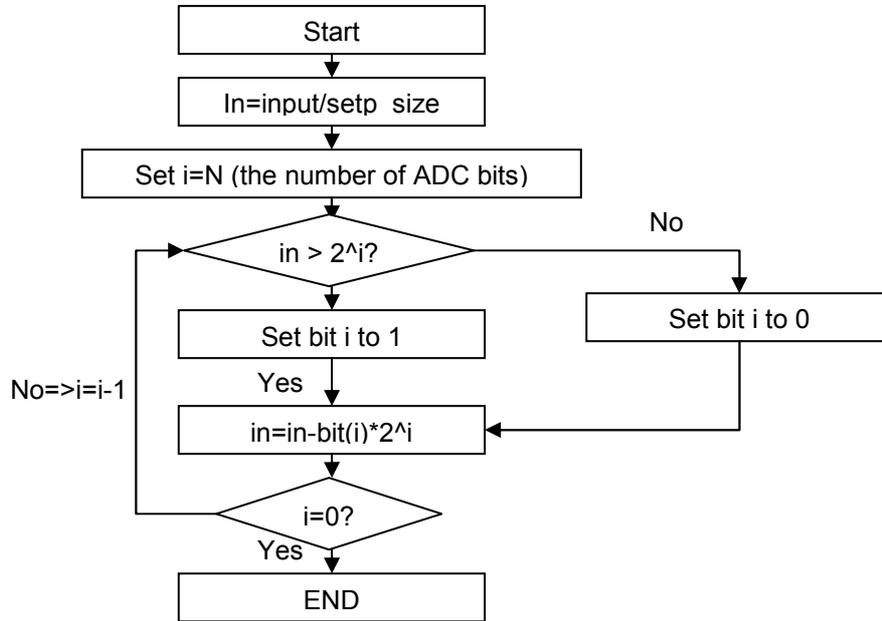


Figure 19 ADC modeling.

4 Phase Locked Loop (PLL) modelling

In the radio system, integer PLL is used to synthesize new frequencies which are integer or fractional times of a reference frequency, with the same stability as the reference frequency. Compare to other blocks of the super-regenerative radio, the PLL is the most complex block which composed of both analog parts and digital parts. In this section the frequency analysis and time model of the PLL are shown.

4.1 Frequency analysis of the PLL

Transient response of phase-locked loops is a nonlinear phenomenon that cannot be formulated easily. However, a linear approximation can be used to gain intuition and understand trade-offs in PLL design. A linear-time continuous PLL model can be used to analyze the PLL in phase domain [6]. The linear model is shown in Figure 20. The output phase of the VCO is Φ_{rf} , then the divider will divide the phase by M and the phase become $\Phi_{div} = \Phi_{rf} / M$. The transfer function of the Loop Filter is $F_{LF}(s)$ and the VCO gain is K_{VCO} . The VCO is modelled as an integrator and the transfer function of which is “ K_{VCO}/s ”, the K_{PFD} is modelled as I_{CP} , where I_{CP} is charge pump current.

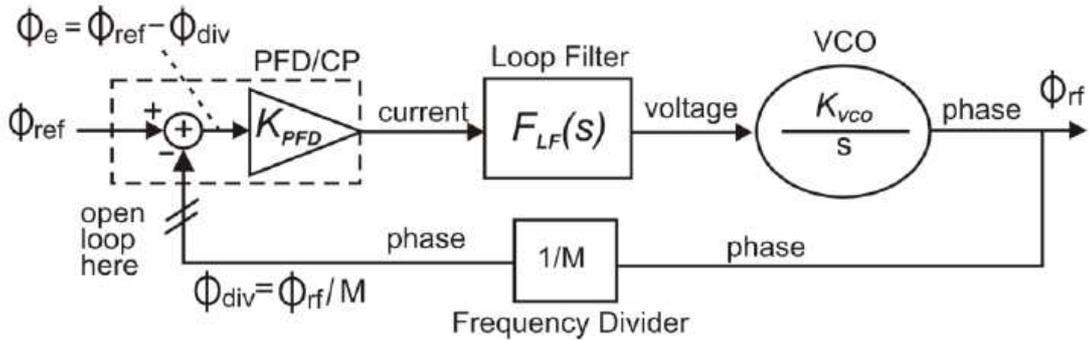


Figure 20 Linear model of the PLL[6].

The open loop transfer function is:

$$H_O(s) = \frac{K_{VCO}}{s} I_{CP} \frac{F_{LF}(s)}{M} \quad [6]$$

As a result, the close loop transfer function is:

$$H_{CL}(s) = \frac{H_O(s)}{1 + H_O(s)} = \frac{K_{VCO} I_{CP} F_{LF}}{sM + K_{VCO} I_{CP} F_{LF}} \quad [6]$$

4.2 The SystemC/SystemC-AMS model

The structure of the integer PLL that modelled in this paper is shown in Figure 21.

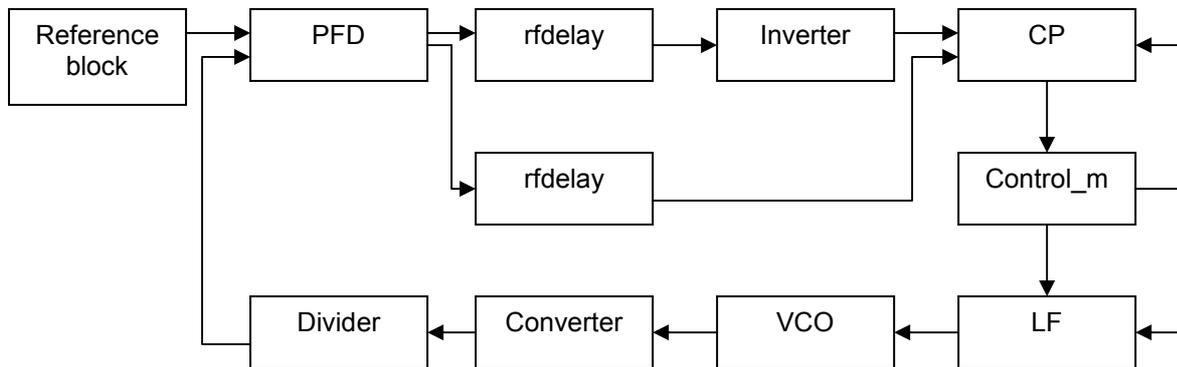


Figure 21 Structure of the PLL.

As shown in Figure 21, the model is composed of eleven blocks, a reference clock block, a phase/frequency detector block, two rfdelay blocks, an inverter block, a charge pump block, a control_m block, a loop filter (LF) block, a voltage control oscillator block, a converter block and a divider block. The reference clock block is used to generate reference signal, the phase/frequency detector block is used to detect the phase/frequency difference between the reference signal and the divider output. After the PFD, The signal goes into rfdelay block and inverter to generate the signals that control the charge pump. Then CP will charge the loop filter to generate the VCO control voltage. The control_m block is used to limit the output voltage of the loop filter. The converter block transfers the VCO analog output signal to a digital input signal that is suitable for the frequency divider.

4.2.1 Reference clock block

The reference block is used to generate the reference clock of the PLL. Part of the code of reference block is shown in Figure 22. It works as follows: assume the period of the reference clock is T, then every T/2, the output will reverse its value. The function “gauss()” is used to generate a gauss random number with 0 mean and variance of 1.0e-9 which is used to model the reference jitter.

```
SC_MODULE (reference){
    sc_out<bool> a;
    sc_in<double> ref;

    void prc_reference();

    SC_CTOR(reference){
        SC_THREAD(prc_reference);
    }
};

void reference::prc_reference(){
    while(1)
    {
        a=!a;
        wait(ref+gauss(1.0e-9),SC_US);
        a=!a;
        wait(ref+gauss(1.0e-9),SC_US);
    }
}
```

Figure 22 Part of source codes of the reference clock block.

4.2.2 The PFD block

The structure of the PFD is shown in Figure 23, it is composed of two D-registers, a delay block and an and-gate. The signals of interest, reference clock and divider output, serve as the clock of the registers. The inputs (D) of the registers are connected to “1”. The registers’ outputs qa and qb will be the output of PFD. If qa and qb are both 0 and there is a rising edge in reference clock, qa will become 1. If this follows by a rising event in divider output the qb will become 1 and reset the registers. A delay block is introduced to model the reset-delay. Also it can be used to eliminate the dead zone. In this way, as shown in Figure 24, the phase and frequency difference will be detected. If the reference frequency is larger than that of divider output or reference phase leads to the divider output phase, there will be a pulse in qa and qb is equal to 0 (maybe with very small pulse because of the reset delay). On the contrary, if the reference frequency is smaller than that of divider output or reference phase delayed from the divider output phase, there will be a pulse in qb and qa is equal to 0.

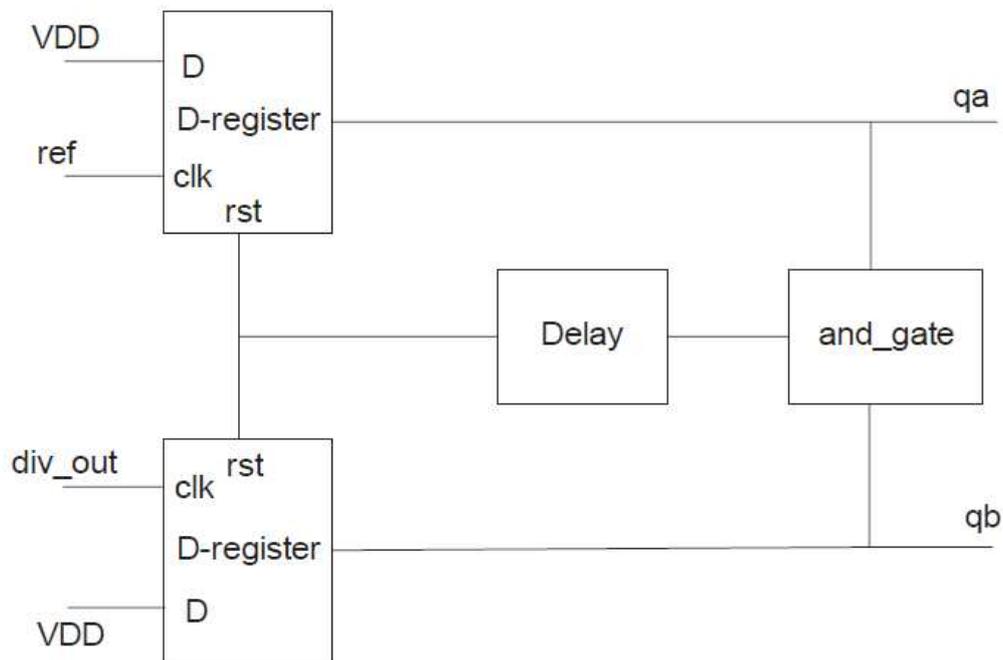


Figure 23 Structure of the PFD.

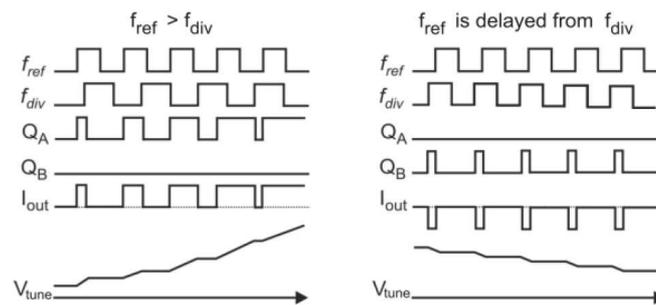


Figure 24 PFD waveform [6].

4.2.3 The rfdelay block

In practice, both the rising edge and the falling edge of a digital signal are non ideal, i.e. there is some delay when the signal changing from "1" to "0" or from "0" to "1". This is the cause of PLL Dead Zone. The rfdelay block is used to model this behaviour. Part of the code of this block is shown 0. It acts as follows: when there is a positive or negative event in the input, the output will increase or decrease gradually until it is larger than "max" V or smaller than "min" V. During the rising or falling behaviour, if the input changes, the output will alter its changing direction.

```

SC_MODULE (rfdelay){
sc_in<bool> clear_in;
sc_out<double> clear_out;

void prc_rfdelay();

SC_CTOR(rfdelay){
SC_THREAD(prc_rfdelay);
}

};

void rfdelay::prc_rfdelay(){
while(1){
wait(clear_in.value_changed_event());
if(clear_in==1)
{
b: while(clear_out<max){
clear_out=clear_out+increase_step; //input=1, increase the output
wait(0.1,SC_NS); //time step size
if(clear_in==0){goto a;} //input changes to 0
}
}
else
{
a: while(clear_out>min){
clear_out=clear_out-decrease_step; //input=0, decrease the output
wait(0.1,SC_NS);
if(clear_in==1){goto b;} //input changes to 1
}
}
}
}
}

```

Figure 25 Part of source codes of the rfdelay block.

4.2.4 Inverter

The inverter is modelled as a simple inverter, which will reverse the value of its input. It should be noticed that the inverter includes the rising and falling delay, which is modelled as the “rfdelay” block. The delay will contribute to the delay difference of the charge pump control signals which will cause reference spurs.

4.2.5 Charge pump and loop filter

The structure of the charge pump and loop filter are shown in Figure 26, the output of the PFD (after the rfdelay and inverter) are connected to gate Mp1 and Mp2, if the frequency of reference clock is larger, the Mp1 will open and the up current source will charge the loop filter. As a result, the tuning voltage “Vtune” will increase, the VCO output frequency will increase. If the reference frequency is smaller, it will go to the reverse way. Figure 27 shows part of the code of the charge pump. The current source of the charge pump is transistors working in saturation region. When the voltage Vtune (vc here) is larger than “min” and smaller than “max”, both pumps works in saturation region, the charge current and discharge current will be equal to the saturation current currentup, currentdw respectively. The “oneone” indicates the current when both switches are closed which is the charge pump current mismatch. When Vtune is smaller than “min”, the down pump goes to the triode region and the current will be smaller. In the same way, when Vtune is larger than “max”, the up pump goes to the triode region, the up pump current will become smaller.

The other imperfection modelled in this charge pump model is the leakage current, when both switches are open ($a > \text{thres} \&\& b < \text{thres}$), there is still a small negative current exist (leakage).

In summary, there are three imperfections modelled in the Charge Pump model, current mismatch, transistor triode region and leakage current. The current mismatch and leakage current will cause the reference spurs. The transistor triode region will result in the increasing of the settling time.

The loop filter is modelled as a second order loop filter with two capacitors and one resistor using the ELN model.

The calculation of the current in the triode region is as follow, we assume the current source is a single transistor current source. So in the saturation region, the current is equal to:

$$I_D = \frac{\mu_n C_{ox} W}{2L} (V_{GS} - V_{TH})^2 \dots (1).$$

Where μ_n is the electron mobility, C_{ox} is the dioxide capacitor, W is the width of transistor, L is the length of the capacitor V_{GS} is gate source voltage and V_{TH} is the threshold voltage.

The current in triode region is:

$$I = \frac{\mu_n C_{ox} W}{L} [(V_{GS} - V_{TH})V_{DS} - \frac{V_{DS}^2}{2}] \dots (2),$$

solve $\frac{\mu_n C_{ox} W}{2L}$ in (1) and put it into (2), and assume the $V_{GS} - V_{TH}$ equal to 0.9 (which means V_{TH} is 0.3V). We can get the triode region current which is $\frac{20}{9} I_D V_{DS} - \frac{100}{81} I_D V_{DS}^2$ (V_{DS} is the drain source voltage).

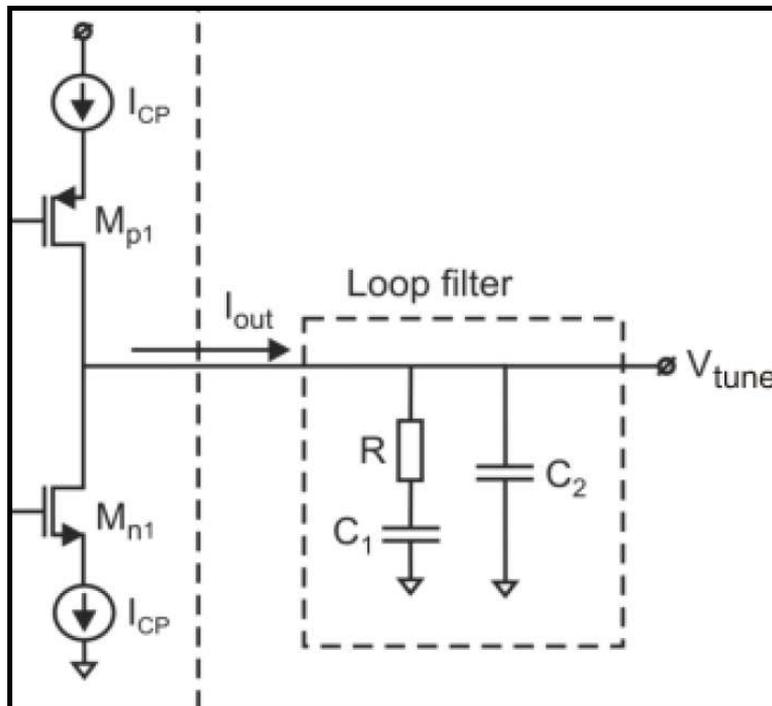


Figure 26 Structure of charge pump and loop filter [6]

```

SCA_TDF_MODULE(charge_pump) {
sc_in<double> a,b;
sca_tdf::sca_in<double> vc;
sca_tdf::sca_out<double> out;
.
.
.
void processing(){

if(vc>min&&vc<max)//transistors of both current source are in saturation
{
    charge=currentup; //up current source saturation current
    discharge=-currentdw; // down current source saturation current
    oneone=charge+discharge;
}
else if(vc<min) // down current source goes into triode region
{
    vds=vc;
    charge=currentup;
    discharge=-20/9.0*currentdw*vds+100/81.0*currentdw*vds*vds;
    oneone=charge+discharge;
}
else if(vc>max) // up current source goes into triode region
{
    vds=1.2-vc;
    charge=20/9.0*currentup*vds-100/81.0*currentup*vds*vds;
    discharge=-currentdw;
    oneone=charge+discharge;
}

if(a<thres&&b<thres) //up current source charge loop filter
{out=charge;}
else if(a>thres&&b>thres) //down current source discharge loop filter
{out=discharge;}
else if(a>thres&&b<thres) //leakage current modeling
{out=-leakage;}
else if(a<thres&&b>thres) //up down current source mismatch
{out=oneone;}
}
};

```

Figure 27 Part of source codes of the charge pump block.

4.2.6 Control_m block

Because the TDF module is based on C++, the voltage has no limitation. In other words, the voltage can go to infinite. The control_m block is used to limit the voltage. When the “Vtune” becomes larger than the supply voltage (1.2V here) or smaller than 0, control_m will cut off the current from charge pump which prevent the “Vtune” to keep on changing. In this way, the Vtune will be limited to the range 0-1.2V.

4.2.7 VCO

The output of VCO is given by:

$$\cos(\omega_0 \times t + 2 \times \pi \times K_{vco} \times \int vcdt)$$

where the ω_0 is free running VCO frequency, K_{vco} is the VCO gain, vc is the control voltage of the VCO (Vtune). Also, VCO noise is included in this model, the way to model VCO noise is

shown in Figure 28, first, a gauss random number generator will create a sequence of random number with 0 mean and variance 1. Then it is multiplied with a gain “ $\sqrt{C_{vco}}$ ” which can be derived from the data sheet of the VCO [7]. After the multiplication, an integrator will integrate the result to change it to phase domain. Finally, add the noise to the output phase of the VCO. Part of the VCO code is shown in Figure 29. In the code, “0.01e-9” is the time step. “in” is the VCO control voltage. “integrationrn” represented the noise phase.

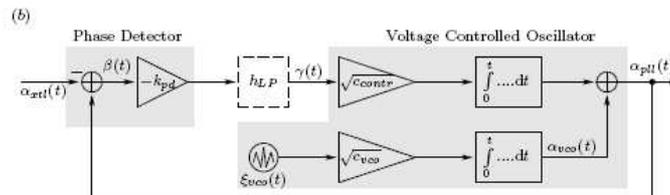


Figure 28 VCO noise model [7].

```

integrationr=integrationr+0.01e-9*gain*2.*M_PI*in;
integrationrn=integrationrn+sqrt(0.01e-9)*gain_noise*2.*M_PI *noise;
out.write( cos( sc_time_stamp().to_seconds()*(control*2.*M_PI)+integratio
nr+integrationrn) );
    
```

Figure 29 Part of source codes of the VCO block.

4.2.8 Converter

The PLL model is a mixed signal model and is needed to convert the signals from analog to digital. A converter block is used to achieve this function. In the feedback loop, it converts the VCO sine wave output to a square wave which is suitable for the digital divider. It acts as follows, when the sine wave’s value is larger than 0, the output is equal to 1, if it is smaller than 0, the output is equal to 0.

4.2.9 Divider

The divider is modelled as a counter. If there is an event in the input, a signal “inter” will add one. If “inter” is larger than a certain number N (“count” shown in the source code), the output will be reversed. Part of the divider code is shown in Figure 30.

```

while(1){
wait(in.value_changed_event());
inter = inter.read() + 1;
if(inter.read() > count){
out=!out.read();
inter.write(0);}
    
```

Figure 30 Part of source codes of the divider block.

4.2.10 Sigma-Delta Modulator

Sigma-delta modulators have been widely used over the last few decades in various signal processing applications. In the PLL, it is mainly used for eliminate the fractional spurs. In this section, the model of a MASH 1-1-1 sigma-delta modulator will be introduced.

4.2.10.1 The structure of a MASH 1-1-1 sigma-delta modulator

In a PLL, sigma delta modulator is used to eliminate the fractional spurs (which is a big problem in fractional-N PLL) by random the divider ratio. It also shifts the low frequency noise to higher frequency. In this section, there is a brief discussion about the structure of sigma-delta modulator.

(1) First order sigma delta modulator

To understand the structure of MASH 1-1-1 sigma delta modulator, let us first look at first order sigma delta modulator. Figure 31 shows the structure of the first order modulator.

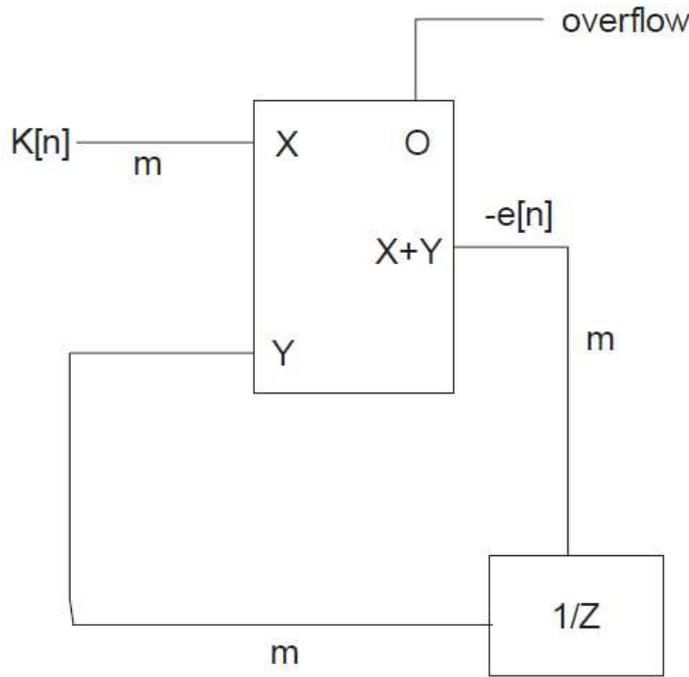


Figure 31 First order sigma delta modulator.

In the figure, we can see that in each cycle, the sum of the m bits input $K[n]$ and $-e[n-1]$ will be calculated and generate a $m+1$ bits signal which includes $-e[n]$ (m bits) and overflow (1 bit). The $-e[n]$ will go into the latch and wait for the calculation of next cycle. As a result: $Overflow(n) = K(n) - (-e[n]) - e[n - 1]$, doing z transform to both sides, we can get:

$$Overflow(z) = K(z) + (1 - \frac{1}{z})E(z) \text{ (E(z) is the z transform of e[n])}$$

(2) Third order sigma delta modulator (MASH 1-1-1)

Figure 32 shows the structure of a MASH 1-1-1 sigma delta modulator. The structure is composed of 3 cascade first order modulators. The output of the modulator Y is:

$$Y(z) = .f + (1 - \frac{1}{z})E1 + [-E1 + (1 - \frac{1}{z})E2](1 - \frac{1}{z}) + [-E2 + (1 - \frac{1}{z})E3](1 - \frac{1}{z})^2 = .f + (1 - \frac{1}{z})^3 E3$$

$N_{div} = N.f(k) + (1 - \frac{1}{z})^3 E3$, where $.f(k)$ is equal to $\frac{input}{2^b}$ (b is the adder bits, N is the divider ratio)

When the PLL is locked, the average output frequency is equal to N_{div} multiplied by reference frequency

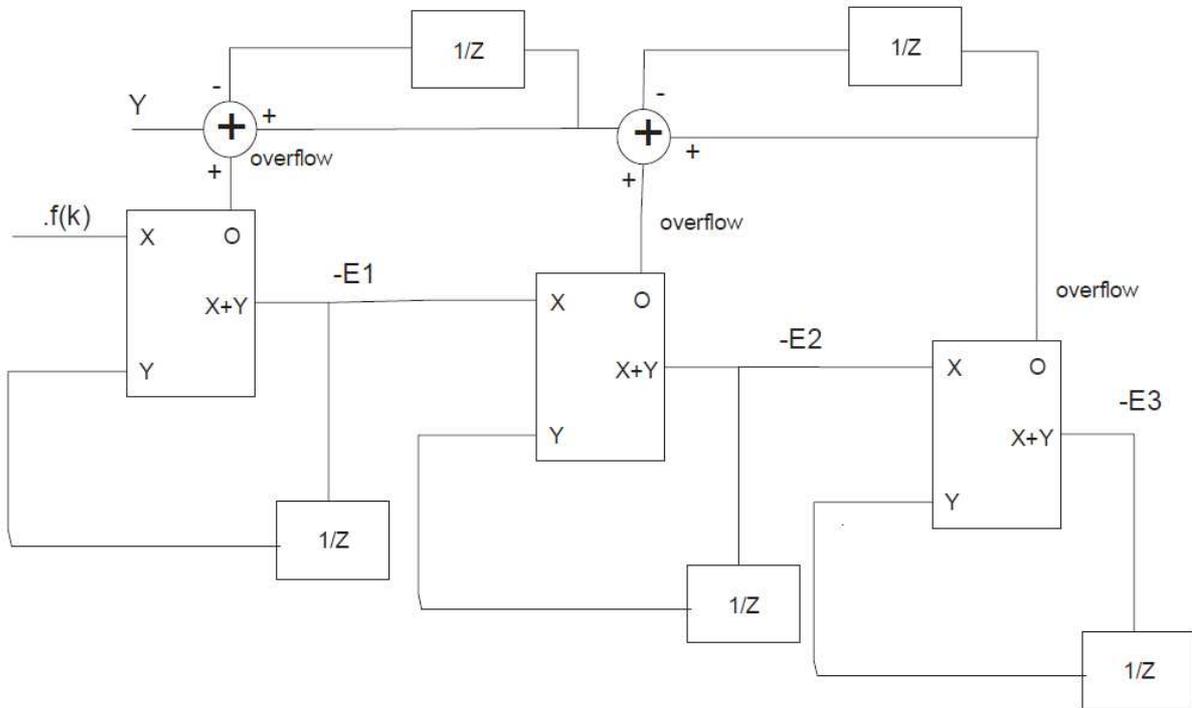


Figure 32 MASH 1-1-1 sigma delta modulator.

5 Frequency Locked Loop (FLL) modelling

Compared with a PLL, a FLL can only locked the frequency but not the phase. However, the structure of a FLL is simpler than a PLL and if the reference frequency is high. The settle time of a FLL is much shorter than a PLL. As a result, in some application, a FLL is much suitable than a PLL. So, it is also worth investigating the FLL model. In this chapter, the SystemC/SystemC-AMS models for two types of FLLs are presented.

5.1 Phase-Aliened FLL (PA-FLL)

5.1.1 Structure of PA-FLL

To reduce the power consumption, most of the RF blocks need to be shut down as long as possible and switched on when needed. As a result, a short settle down time is needed. Compare to a PLL, the PA-FLL has a much faster settle down time. The structure of PA-FLL is shown in Figure 33. In the figure, the red parts are digital circuits which are modelled with SystemC. The blue parts are analog circuits which are modelled with SystemC-AMS TDF model.

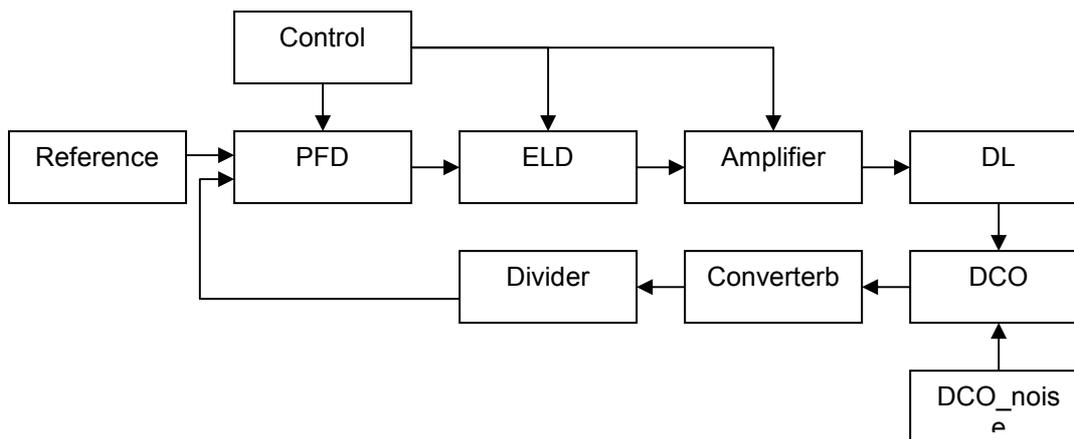


Figure 33 FLL structure.

This section will focus on the brief discussion of the modelling of each part of the PA-FLL.

5.1.1.1 Reference

The reference block is used to generate the reference clock of the FLL.

5.1.1.2 Phase/Frequency Detector

The PFD block is used to detect the phase difference between the 2 inputs, it acts as follows. First it will set the output to 0 and when there is an event in either 2 inputs, the output will reverse. The timing sequence is shown in Figure 34.

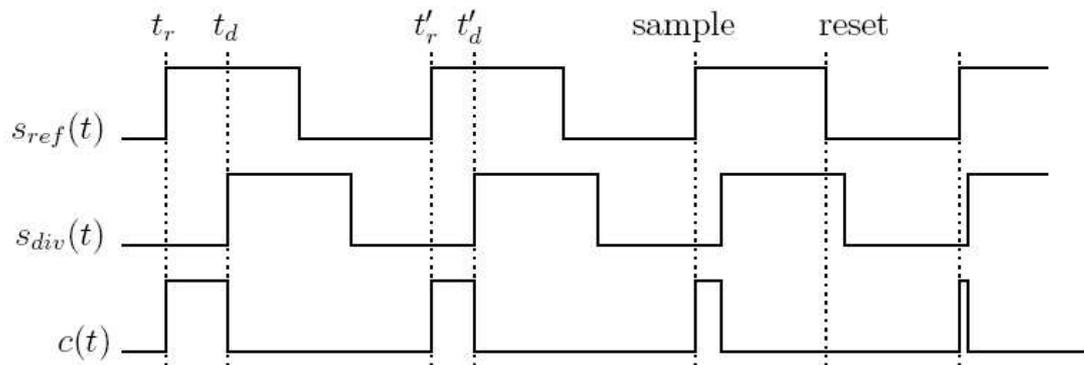


Figure 34 Waveform of the PFD [8].

5.1.1.3 Early Late Detector

The structure of early late detector is shown in Figure 35. The timing sequence of the ELD is shown in Figure 36. There are 3 states for this block. During state P1, S_1 will close and other switch will open. C_1 will be charged and V_{c1} will increase. During stage P2, S_2 will close, C_2 will be charged and V_{c2} will increase. During stage 3, V_{c1} and V_{c2} will be sampled. If V_{c1} is larger, the output will be -1. If V_{c2} is larger the output will be 1. When reset signal is set, S_3 will close and V_{c1} and V_{c2} will be set to 0.

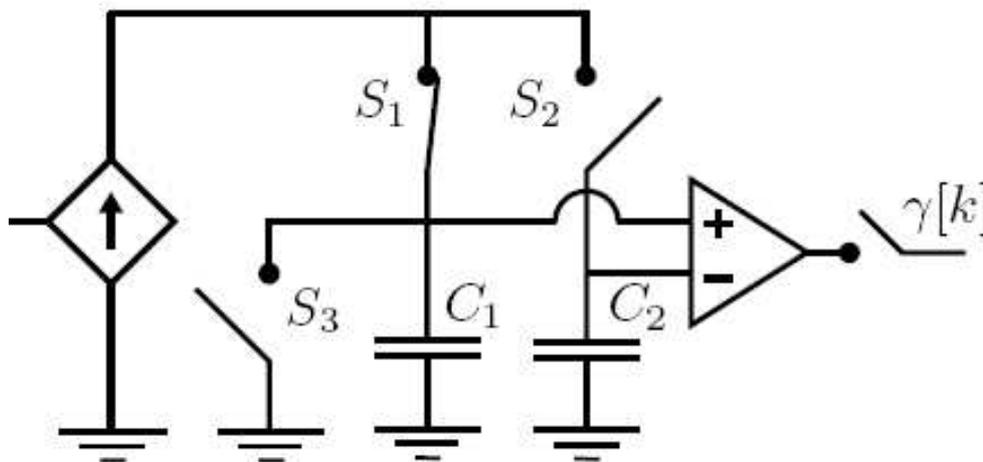


Figure 35 Structure of the ELD [8].

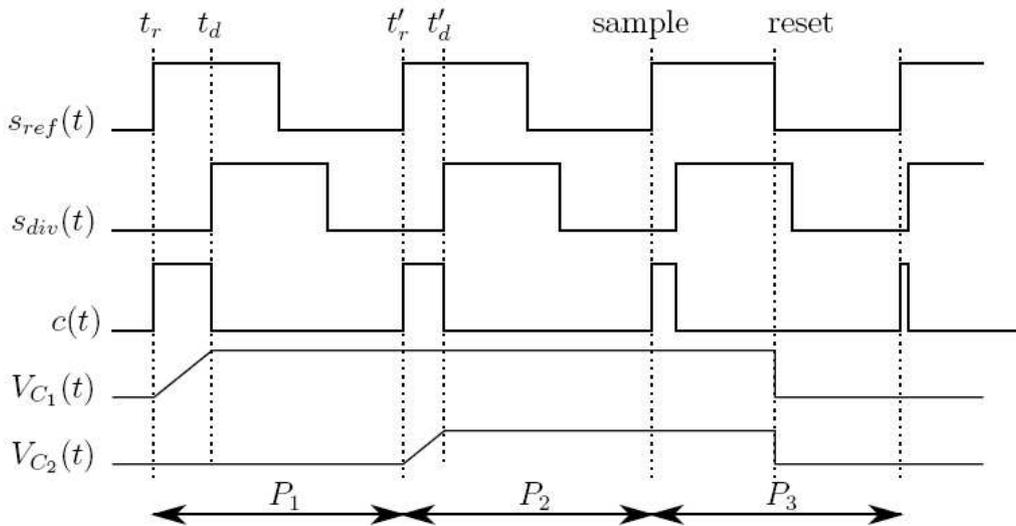


Figure 36 Figure 4 waveform pf the ELD [8].

5.1.1.4 Control block

The control block is used to generate the control signals of the system. The signals include: reset signal, sample signal, state signal. The state signal changes when there is a rising edge in $c(t)$ ($P1 \Rightarrow P2 \Rightarrow P3 \Rightarrow P1$), sample signal changes every 3 rising edge of $c(t)$ which means every 3 cycles. When sample signal changes, the voltage of the 2 capacitors will be sampled. At the first falling edge of $c(t)$ after the sample signal, the reset signal will change and the system will reset the 2 capacitors.

5.1.1.5 Decision Logic

This block is just an integrator (integrate the ELD output) and is used to generate the control bits of the DCO, every time the sample signal change, the DL will add its input to its output.

5.1.1.6 DCO

The DCO is modelled in behaviour level. The output of DCO is equal to $\cos(\omega_0 \times t + 2 \times \pi \times Kdco \times \int vcdt)$ where the ω_0 is the free running DCO frequency, $Kdco$ is the frequency represent by 1 bit, vc is the control bits of the DCO.

5.1.1.7 Converter

The FLL model is a mixed signal model and it is needed to convert the signals from analog to digital, the converter block is used to achieve this behaviour. In the feedback loop, it converts the DCO sine wave output to a square wave which is suitable for the digital divider.

5.1.1.8 Divider

The divider is modelled just as a counter. If there is an event in the input, a signal “inter” will add one. After “inter” reach a certain number N , the output will be reversed. In this way, an “ $N+1$ ” divider is achieved. It is same with the divider that used in the PLL model.

5.1.1.9 Imperfections

I add three imperfections to the PA-FLL:

DCO noise

Reference jitter

Leakage current in ELD

5.1.1.10 DCO noise

Figure 37 shows the way to model the DCO noise. The noise is generated with a Gauss white noise generator. Then it is multiplied with a gain which indicates the quality of the DCO. After the multiplication, integrate the result to change it to phase domain and then add it to the output phase of the DCO.

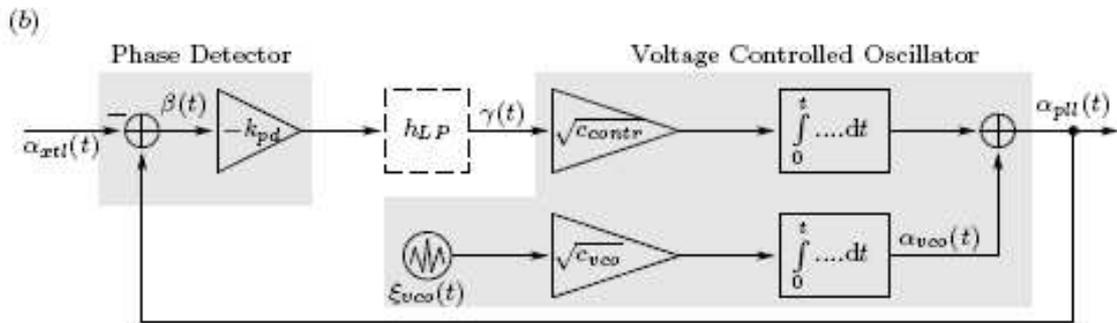


Figure 37 DCO noise [7].

5.1.1.11 Reference jitter

The reference clock will never be an ideal clock in practice, there is always a small jitter exist which affects the performance of the FLL. The way to add the reference jitter is the same with that of the PLL.

5.1.1.12 Leakage current in ELD

When S1 is open, there should not be any current in that path in ideal case. However, in practice, there is a small leakage current. This may cause the judging error. I modelled the leakage by add a small current in S1 path during phase 2.

5.2 Type-2 FLL

5.2.1 The structure of the FLL

The structure of the FLL is shown in Figure 38. In the figure, FVC is Frequency to Voltage Converter, which converts the input frequency to an output voltage. The higher the input frequency is, the lower the output voltage is.

An opamp is used to compare the different of Vin1 and Vin2, and then add the result to the control voltage (Vctr). Depending on the voltage difference between Vin1 and Vin2, the Vctr will adjust the output frequency of the VCO until the voltage Vin2 is equal to the voltage Vin1. Once Vin2 reaches Vin1, the output voltage Vcfr becomes constant and forces the output frequency of VCO Fout to stay also constant.

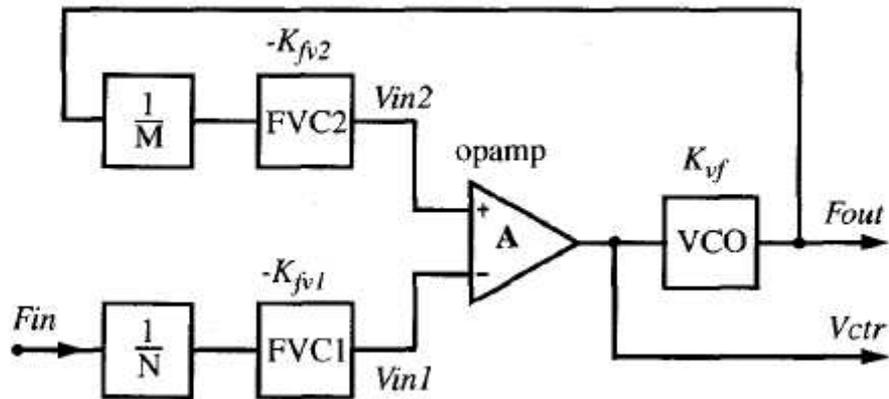


Figure 38 The structure of pll based on frequency to voltage converter [9].

5.2.1.1 The FVC

The structure of FVC is shown in Figure 39, and the waveform of the FVC is shown in Figure 40. The FVC has 3 phases, during the first phase, S1 will close and the current source will charge capacitor C1. During the second phase, S2 will close and S1, S3 will open, and there will be a charge sharing between C1 and C2. During phase 3, the S3 will close, and the C1 will be discharged.

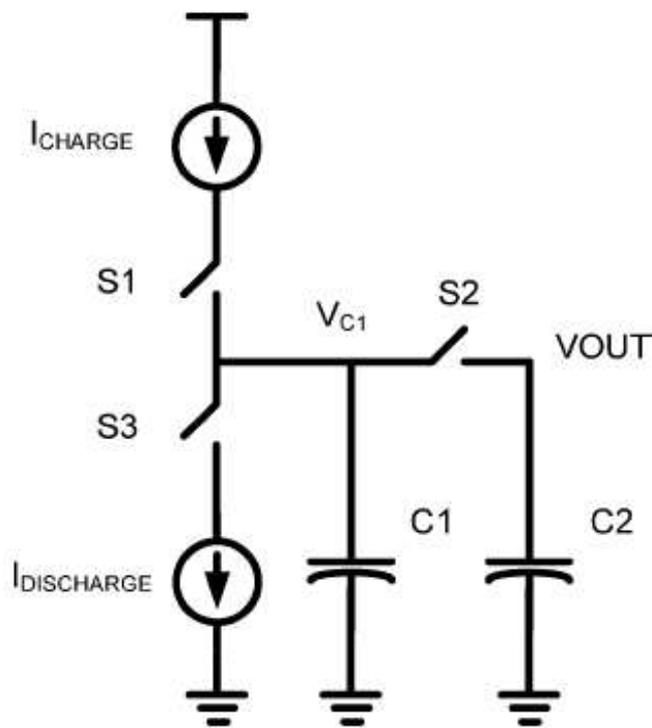


Figure 39 The structure of FVC [10].

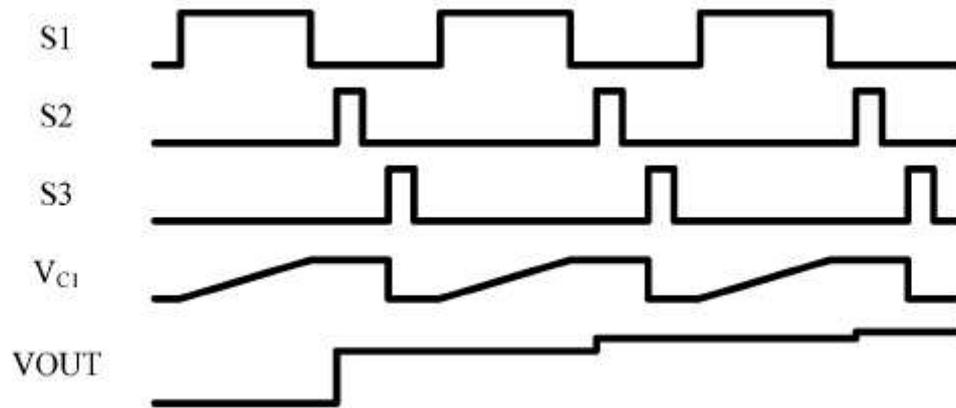


Figure 40 Waveform of the FVC [10].

5.2.1.2 The opamp

The opamp will calculate the different between V_{in1} and V_{in2} and integrate the result to generate the VCO control voltage (V_{ctr}). It is modelled with SystemC-AMS TDF model.

5.2.1.3 The VCO and divider

The VCO and divider is the same with which used in the PLL model.

6 The digital baseband of the model

The analog system has been discussed in previous chapters. In this chapter, we focus on the digital baseband model of the radio. Section 6.1 describes the model of the transmitter part and Section 6.2 shows the receiver part. It should be noticed that the PPM synchronization algorithm is first represented here and it is also implemented with VHDL.

6.1 The transmitter SystemC/SystemC-AMS model

The transmitter part is used for packet generation and PPM modulation. The structure of the transmitter part is shown in Figure 41. It is composed of two layers, the transmitter data link layer and transmitter physical layer. The data link layer is used to generate the data structure. The physical layer is used to realize the PPM modulation and transfer the encoded data to the analog environment.

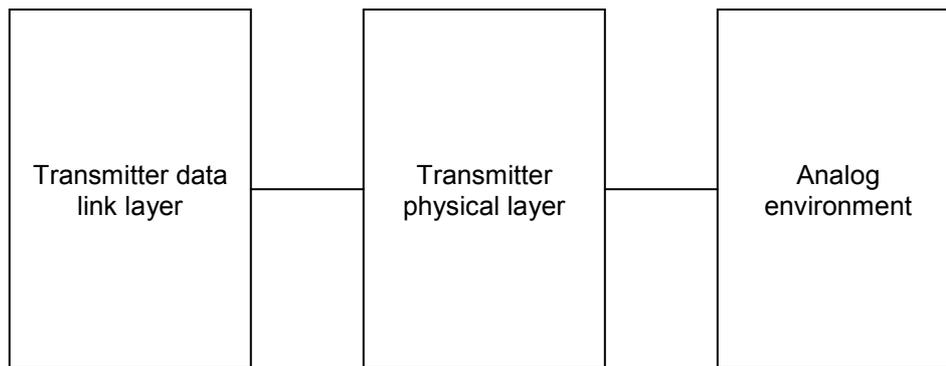


Figure 41 The structure of the transmitter part.

6.1.1 Transmitter data link layer

Part of the source code of the transmitter data link layer is shown in Figure 42. The function of the data link layer is to generate the data structure and pass the data to the transmitter physical layer. The data is composed of 2 parts – the preamble part and the data part, the preamble part is a known data that could enable the synchronization algorithm in the receiver and the data part is the real data to be transferred. In our system, the preamble part is composed of 128 bits “1” (32 pulses after PPM modulation) and a 8 bits SFD, the code for SFD is “10011010”. In Figure 42, the first “for” loop is used to generate the 128 “1”, the second “for” loop is used to generate the data. The codes between the two loops are used to generate SFD. It should be noticed that every time there is an output bit in the data link layer, a small pulse of 10ns is generate in the port “doa” to inform the physical layer that one bit is transfer. In the data part, the “guass” function is used to generate a Gaussian random number with 0 mean and variance 1. If the generated number is larger than 0, the output is 1, or the output is 0. In this way, the random data are generated.

6.1.2 Transmitter physical layer

The function of the physical layer is to realize the PPM modulation and transfer the data to the analog environment. At every rising edge of the "dor" (come from the analog environment), a variable "count" will add one and the lowest bit of output buffer vector "outvec" will be sent out. If the remaining number of count divided by 4 is 3, the "dir" is set to "1" to read a number from the data link layer. At the same time, the input buffer vector "invec" will shift 1 bit left and the read data become the last bit of "invec". When "count" is equal to "15", the "outvec" will be reset according to the number of "invec". The rules is as follows, if the value of the "invec" is "x", the bit x of "outvec" is set to 1 and the other bits are set to "0". This is our implementation of the 4 to 16 PPM modulation.

```

for(int i=0; i<128; i++)                // preamble generate
{
    wait(dor.posedge_event() );
    a = 1;
    doa = 1;
    wait(10,SC_NS);
    doa = 0;
}
wait(dor.posedge_event() );            // SFD generate
a = 1;
doa = 1;
wait(10,SC_NS);
doa = 0;
wait(dor.posedge_event() );
a = 0;
    doa = 1;
    wait(10,SC_NS);
    doa = 0;
wait(dor.posedge_event() );
a = 0;
    doa = 1;
    wait(10,SC_NS);
    doa = 0;
wait(dor.posedge_event() );
a = 1;
    doa = 1;
    wait(10,SC_NS);
    doa = 0;
wait(dor.posedge_event() );
a = 1;
    doa = 1;
    wait(10,SC_NS);
    doa = 0;
wait(dor.posedge_event() );
a = 0;
    doa = 1;
    wait(10,SC_NS);
    doa = 0;
wait(dor.posedge_event() );
a = 1;
    doa = 1;
    wait(10,SC_NS);
    doa = 0;
wait(dor.posedge_event() );
a = 0;
    doa = 1;
    wait(10,SC_NS);
    doa = 0;
for(int j=0; j<80; j++)                // data generate
{
    wait(dor.posedge_event() );
    temp = gauss(1);
    if(temp>0)
    {a = 1;}
    else
    {a = 0;}
    doa = 1;
    wait(10,SC_NS);
    doa = 0;
}

```

Figure 42 Part of the source code of transmitter data link layer.

6.2 The receiver SystemC/SystemC-AMS model

The receiver part is used to receive the data from the analog environment, realize the synchronization algorithm and recover the data from the 4 to 16 PPM modulation. The system will collect 16 chips, then translate them into a 4 bits data and send the data out. The structure of the receiver algorithm is shown in Figure 43. At beginning, the data sampled by ADC will be buffered. Because the sample rate is three times higher than the data rate, each chip has three samples. So, to collect 16 chips, 48 samples are collected. Then, the threshold will be set. After that, the system will go into the “detection” stage. If the data value is larger than the threshold, the system will go into the “confirmation” stage. In the “confirmation” stage, the system will confirm whether the data is coming. If this is confirmed, the system will go to “timing” stage. If not, it will go back to the “detection” stage. In the “timing” stage, the offset of the timing will be decided and in “calibrate” stage, the offset will be compensated. After that, the system will detect the SFD signal. Once the SFD is detected, the system will enter the “data” stage for data detection.

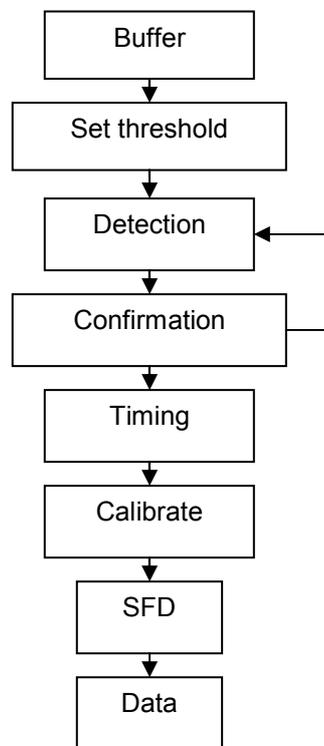


Figure 43 The structure of receiver.

6.2.1 The buffer

As mentioned at the beginning, every period, the system will collect 16 chips and each chip has 3 samples. So the buffer will collect 48 samples, the first sample of each chip are stored in array data0, the second sample of each bit are stored in data1 and the third sample are stored in data2. Part of the source code of the buffer is shown in Figure 44, each time there is a sample comes, there will be a rising edge in the “dia”, at that time, the variable “counta” will add 1, the “counts” indicate which array should the sample be stored, so when it is “0” the data is stored in data0, when it is “1” the data is stored in data1 and so on. The “count” indicate the chip of the data, it will increase 1 every 3 samples. If “counta” is equal to 16, 32 or 48, data0, data1 or data2 is assigned to the output. The code is shown in Figure 45. The “doa” indicates that there is an output from buffer.

```

wait(dia.posedge_event() );
counta = counta + 1;
    if(counts == 0)
    {
        data0[count] = data;
        counts = 1;
    }
    else if(counts == 1)
    {
        data1[count] = data;
        counts = 2;
    }
    else if(counts == 2)
    {
        data2[count] = data;
        counts = 0;
        if(count == 15)
        {count = 0;}
        else
        {count = count + 1;}
    }
}
}

```

Figure 44 Part of the source code of buffer.

```

if(counta == 16)
{
    doa = 1;
    wait(5, SC_NS);
    datao0 = data00[0];datao1 = data00[1];datao2 = data00[2];datao3 =
    data00[3];datao4 = data00[4];datao5 = data00[5];datao6 = data00[6];datao7
    = data00[7];datao8 = data00[8];datao9 = data00[9];datao10 =
    data00[10];datao11 = data00[11];datao12 = data00[12];datao13 =
    data00[13];datao14 = data00[14];datao15 = data00[15];
    position = 0;
}
else if(counta == 32)
{
    doa = 1;
    wait(5, SC_NS);
    datao0 = data11[0];datao1 = data11[1];datao2 = data11[2];datao3 =
    data11[3];datao4 = data11[4];datao5 = data11[5];datao6 = data11[6];datao7
    = data11[7];datao8 = data11[8];datao9 = data11[9];datao10 =
    data11[10];datao11 = data11[11];datao12 = data11[12];datao13 =
    data11[13];datao14 = data11[14];datao15 = data11[15];
    position = 1;
}
else if(counta == 48)
{
    doa = 1;
    wait(5, SC_NS);
    datao0 = data22[0];datao1 = data22[1];datao2 = data22[2];datao3 =
    data22[3];datao4 = data22[4];datao5 = data22[5];datao6 = data22[6];
    datao7 = data22[7];datao8 = data22[8];datao9 = data22[9];datao10 =
    data22[10];datao11 = data22[11];datao12 = data22[12];datao13 =
    data22[13];datao14 = data22[14];datao15 = data22[15];
    position = 2;
    counta = 0;
}
}

```

Figure 45 Part of the source code of buffer.

6.2.2 The threshold estimation

Before the synchronization algorithm, a threshold used in detection and confirmation should be estimated. In this step, the input of the receiver is set to 0. So the estimation is based on the mean and variance of the noise, the steps for threshold estimation is shown in Figure 46. At first, “num” chips are collected from the buffer. Then absolute values of the difference between the continuous data are calculated (cal_d). After that, the mean and variances of the “num” absolute values are calculated. To make the calculation easy, the variance is defined as the mean of the absolute value of the differences between the cal_d and the mean. Finally, the threshold is set to the sum of the mean, “a” times of the variance (“a” is the parameter that indicate how much effect the variance give) and a constant “x”. The constant “x” is user defined.

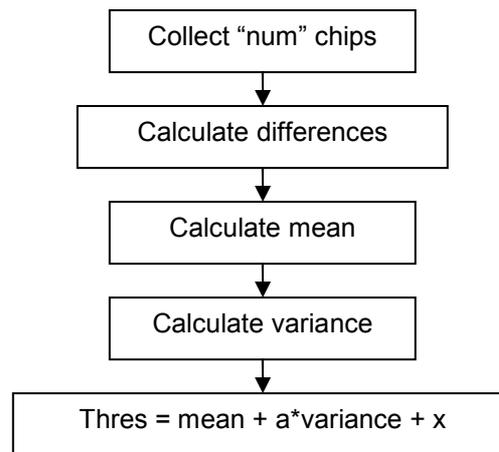


Figure 46 The steps of threshold estimation.

6.2.3 The Detection

The detection step is used to detect whether the packet has arrived. The steps for detection are shown in Figure 47. In 4 to 16 PPM modulation, each 4 bits data is represented by 16 chips. So, we assume each 16 chips is a group and the algorithm is based on group. First, one group (16 chips) is collected from the buffer. Then the system will calculate the differences between the adjacent values. After that, the max value of these differences is selected and compared with the threshold estimated in the threshold estimation stage. If the “max” is larger than the threshold, the data is detected and the system will go into the “confirmation” stage. If not, the system will do the whole steps again.

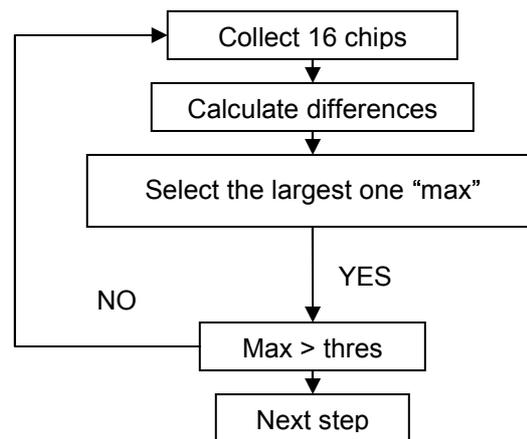


Figure 47 The steps of detection.

6.2.4 The confirmation stage

The confirmation stage is used to confirm the data packet is coming, the steps of the confirmation is nearly the same with that of detection except that it will repeat detection step for several times and average the max value in each iteration. Then, the final max number will compare with the threshold, if it is larger, the system will go to the “timing” stage and if not, the system will go back to detection stage.

6.2.5 The timing stage

In PPM modulation, every 4 bits data is represented by 16 chips. As a result, to restore the data correctly, we need to know the beginning of the 16 chips. The timing step is used to do this. It will determine the “timing offset” of the data which means how many samples exist to the beginning of next 16 chips. The steps of timing are shown in Figure 48. First collect 48 samples and divided them into 3 groups (each 4 bits are represented with 16 chips and each chip has 3 samples, so we need 48 samples for each 4 bits), then select the largest value of each group and store the position of them in max1, max2, max3. The “pos1” is equal to the position of the second group (max2). Then “pos2” is decided, if the maximum positions of each group are equal, “pos2” is equal to 3. If the maximum position of the first group is equal to the second one but not the third one, “pos2” is equal to 2. If the maximum position of second group is equal to the third one but not the first one, “pos2” is equal to “4”. Finally, the offset is set to “3*pos1 + pos2”.

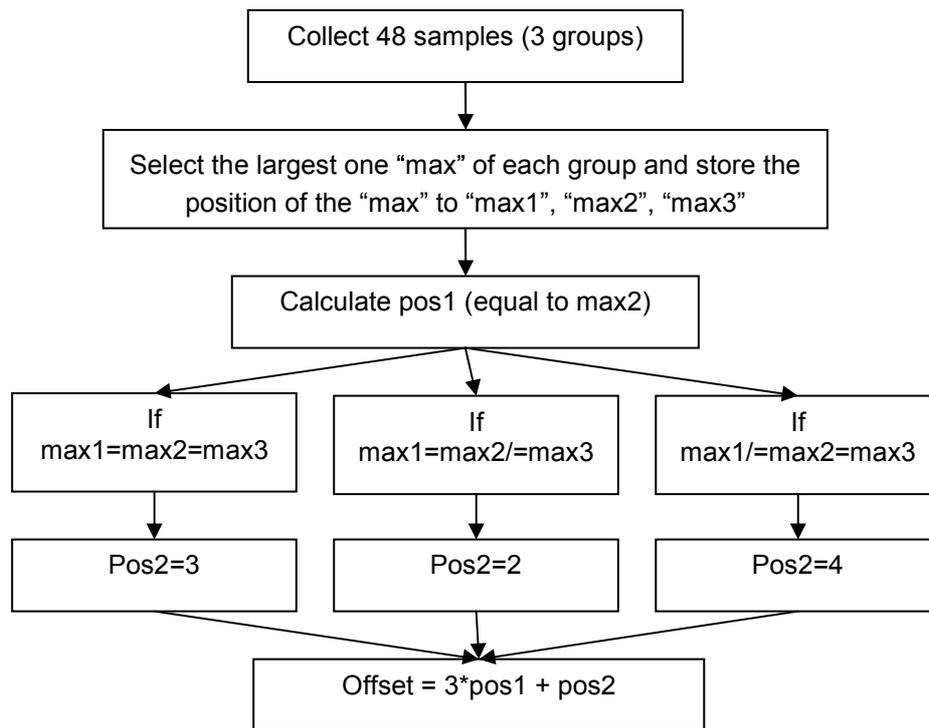


Figure 48 Steps for timing.

6.2.6 The SFD detection

The SFD detection is the last step of the synchronization algorithm. After the 8 bits “SFD”, the data is coming, as a result, when the “SFD” signal is detected, the data collection begins. The steps for SFD detection are shown in Figure 49. First, collect 16 chips (4 bits). Then, select the maximum value of this group and store the position of the max value in max0. Then collect another 16 chips (4 bits) and store the position of max value in max1. If max0 is equal to 9 and max1 is equal to 10 or max0 is equal to 10 and max1 is equal to 9 (the SFD value is 10011010), the SFD signal is detected and the system will go to the data collection. If not, the system will continuously repeat the steps mentioned above until the SFD is detected.

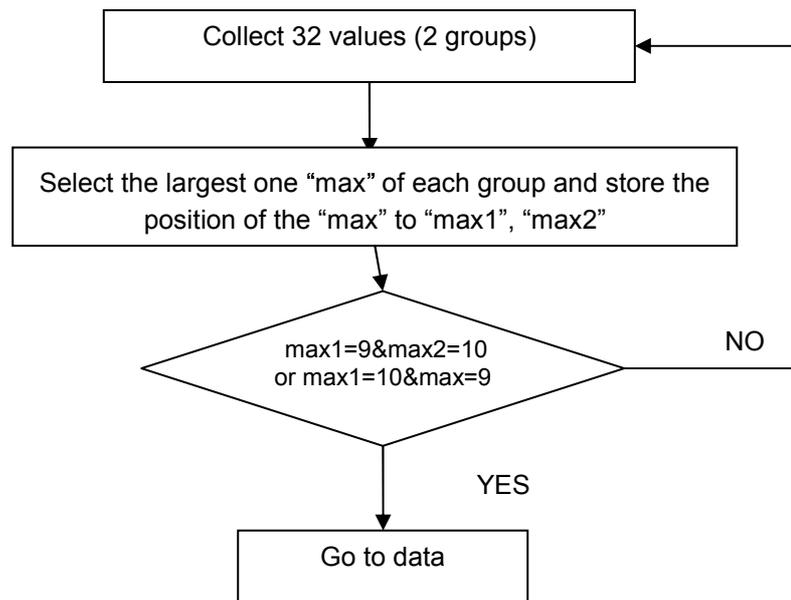


Figure 49 The steps for SFD detection.

6.2.7 Data collection

The data collection is used to restore the data from the encoded PPM signal. The steps for data collection are shown in Figure 50. First, collect 16 chips (4 bits). Then find the maximum value among the 16 chips. Finally, the system will decode the data according to the position of the maximum value.

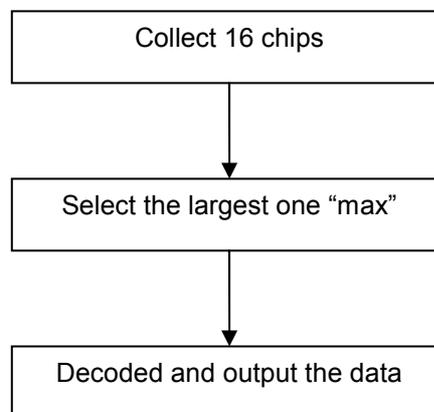


Figure 50 Steps for data collection

7 Simulation results

The model presented in this thesis has been used to simulate the radio represented in [5]. In this section the simulation results of the system will be represented. Also, to ensure the correctness of the system, the results are compared to that of some simulation tools as well as measurement results. This section is organized as follows. In the first part, the PLL and FLL simulation results are represented. In the second part, the simulation results of the analog parts are shown. In the third part, the simulation results of the whole system are represented.

7.1 The simulation results of PLL and FLL

The model introduced in previous chapters has been used to simulate the integer PLL represented in [6]. The PLL has a reference clock of 500kHz, an output frequency of 2.4GHz, a charge pump current of 50uA. The leakage current is set to 25pA and the charge pump current mismatch is 5uA. The divider ratio is 4800 and the VCO gain is 160MHz/V. The threshold for charge pump transistor is 0.6V and the supply voltage is 1.2V. The range for the Vds to keep the charge pump transistor saturated is 0.3V. The simulation time for 1200us transient analysis with a 0.01ns resolution is about 15 minutes. To ensure the correctness of the model, the simulation results are compared with the Matlab simulation results, Analog Device’s ADI SimPLL simulation results, Cadence simulation results, and the measurement results. The parameter values in ADI SimPLL, Cadence and real circuits are equal to the values mentioned above. When comparing with the Matlab model, the parameters are set as follows: a reference clock of 31.2MHz, an output frequency of 124.8MHz, an up pump current of 25uA and a down pump current of 15uA. The free running VCO frequency is 124MHz.

7.1.1 Comparison with Matlab simulation results

The simulation result of the SystemC/SystemC-AMS model is first compared with the simulation results of a Matlab model. The Matlab model is a sample-driven simulator for charge-pump PLLs. To shorten simulation-time, the VCO output signal is modelled in the phase domain, such that the frequency division becomes a mere scaling of the phase. The phase noise is modelled as a Gaussian random process and the current-source imbalance can be simulated.

The two models have the same parameter values and include the same imperfections. Figure 51 shows the VCO output power spectrum density (PSD) of the two models, in which the red line is the PSD of the SystemC/SystemC-AMS model while the blue one is that of the Matlab model. As shown, the two results are very similar.

Figure 52 and Figure 53 show the time domain simulation results of VCO control voltage. The settling time of the Matlab model is about 1.5 us whereas that of the SystemC/SystemC-AMS model is about 1.5 us too. It is clear that our model and the Matlab have comparable simulation results. The simulation time of SystemC/SystemC-AMS model (about 2 minutes for 650 us simulation with a resolution of 0.1 ns) is about 10 times smaller than that of the Matlab model (about 20 minutes for 639 us simulation with a resolution of 0.1 ns).

Another advantage of SystemC/SystemC-AMS over Matlab is its capability to simulate very high frequency signal in a high resolution. For example, for a 2.4GHz signal with resolution of 0.01ns, SystemC/SystemC-AMS is able to simulate it while Matlab fails (Matlab will run out of memory).

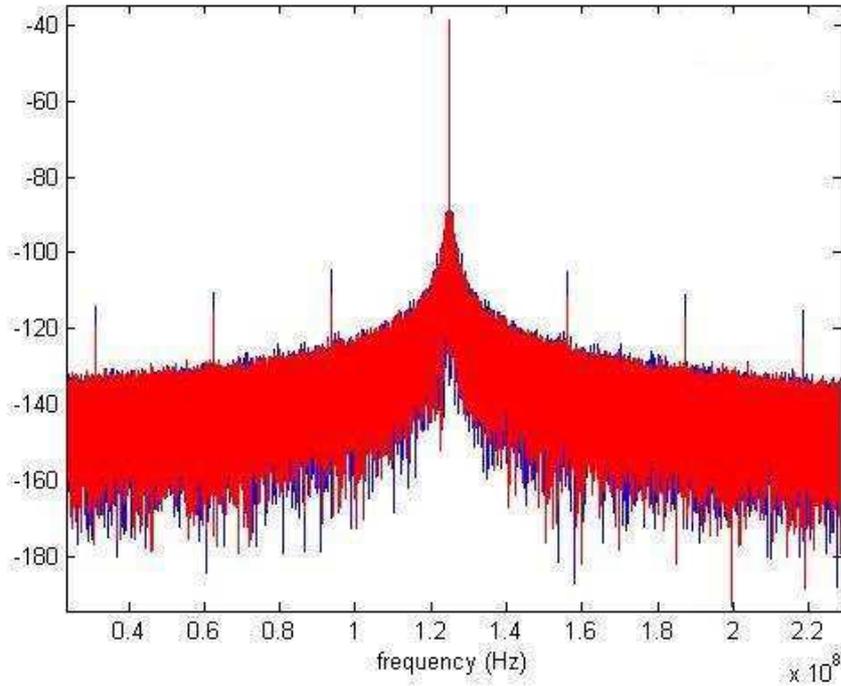


Figure 51 PSD of VCO output of our and Matlab models.

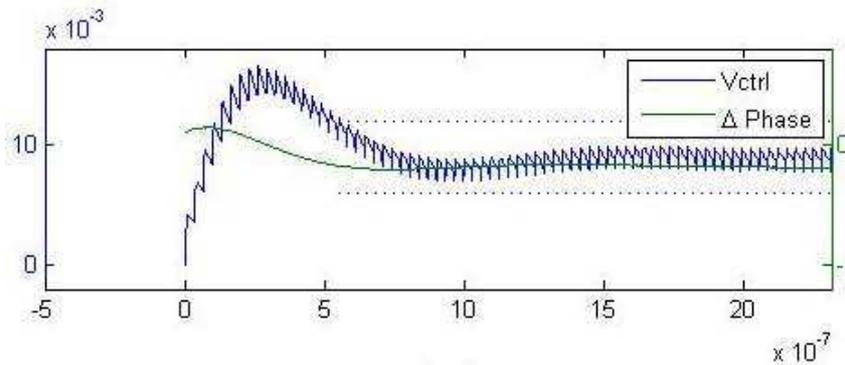


Figure 52 VCO control voltage of the Matlab model.

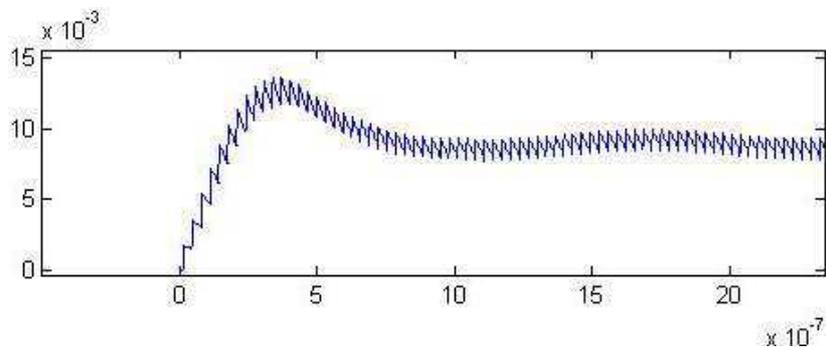


Figure 53 VCO control voltage of our model.

7.1.2 Comparison with ADI SimPLL simulation results

The simulation tool ADI SimPLL published by Analog Devices is used to verify the close loop simulation results of the SystemC/SystemC-AMS model. Figure 54 shows the VCO output

frequency of SystemC/SystemC-AMS model and Figure 55 shows the frequency of the ADI tool’s simulation result. The parameter values of both are the same. The settling time of both results is about 60 us and the waveforms are comparable. Thus, the closed loop behaviour is correctly modelled with SystemC/SystemC-AMS.

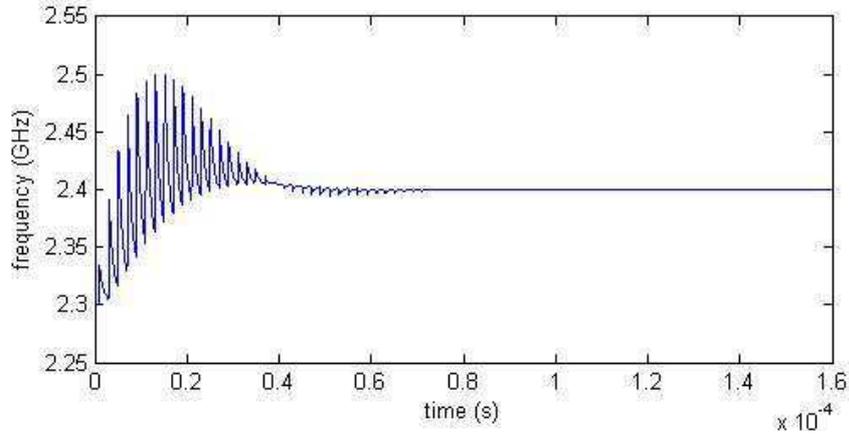


Figure 54 VCO output frequency of our model.

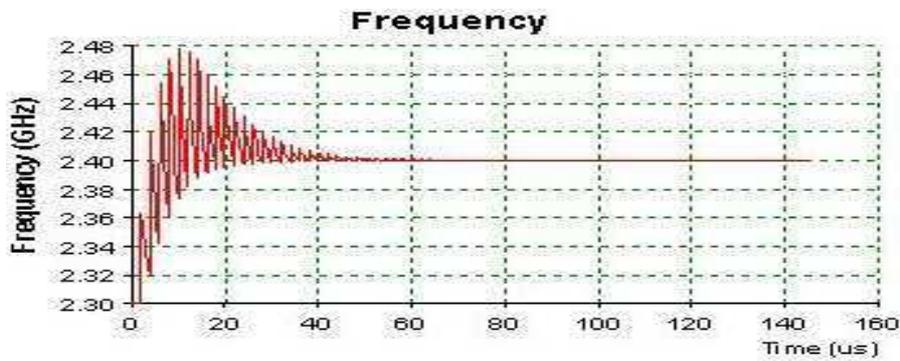


Figure 55 VCO output frequency of the ADI SimPLL tool.

7.1.3 Comparison with Cadence simulation results

The simulation results of the PLL closed loop behaviour were verified in the previous subsection. In this section, the open loop simulation results are verified by Cadence simulation results. Figure 56 and Figure 57 show the open loop simulation results of Cadence and the SystemC/SystemCAMS model respectively. The input signals of the two models are same. In the figure, the first two signals are the outputs of PFD, the third signal is VCO control voltage V_{tune} , the last signal is the charge pump current. As can be seen from the figure, in the 18th cycle, the value of V_{tune} of Cadence simulation results is 403.9 mV and that of SystemC/SystemC-AMS model is 417.0 mV, which are similar. The other signals are also similar, except that in the Cadence results, there are some current spurs when the current source goes into the triode region. Normally the current sources will not enter the triode region. Thus these spurs are not modelled here in order to further decrease simulation time.

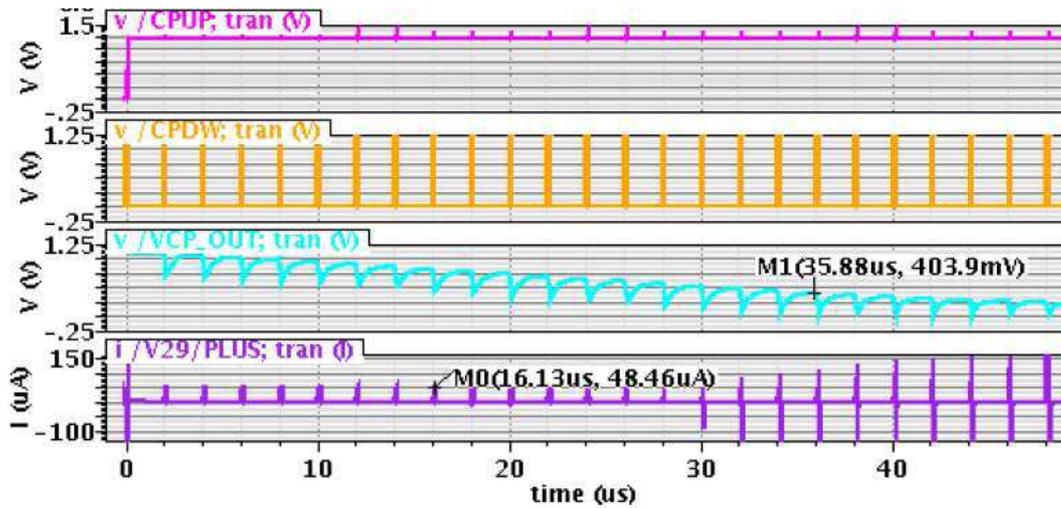


Figure 56 Open loop simulation results of the Cadence.

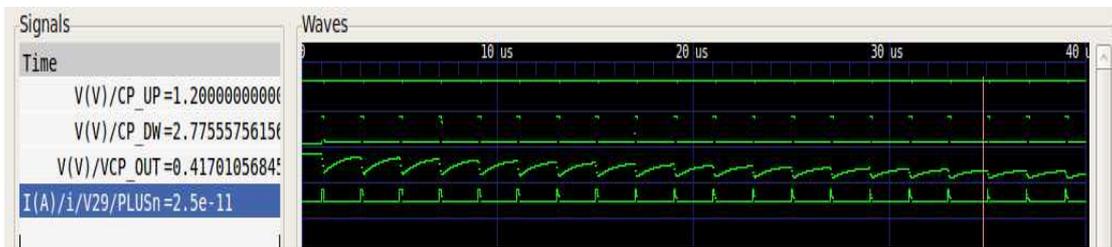


Figure 57 Open loop simulation results of our model.

7.1.4 Comparison with measurement results

Figure 58 shows the measurement PSD of the Pre-Scaler output (VCO output signals divided by 30) of the PLL in [6] while Figure 59 shows that of the SystemC/SystemC-AMS model. From these two figures, it is clear that the model can correctly reflect the real PLL noise behaviour.

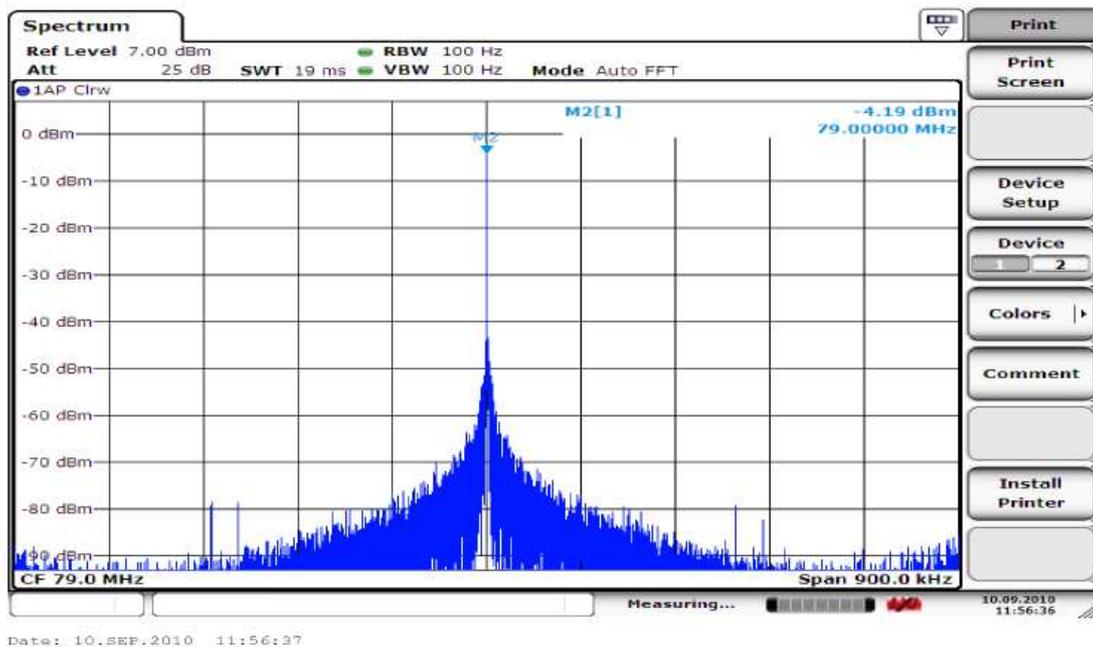


Figure 58 Measurement result of the prescaler output PSD [6].

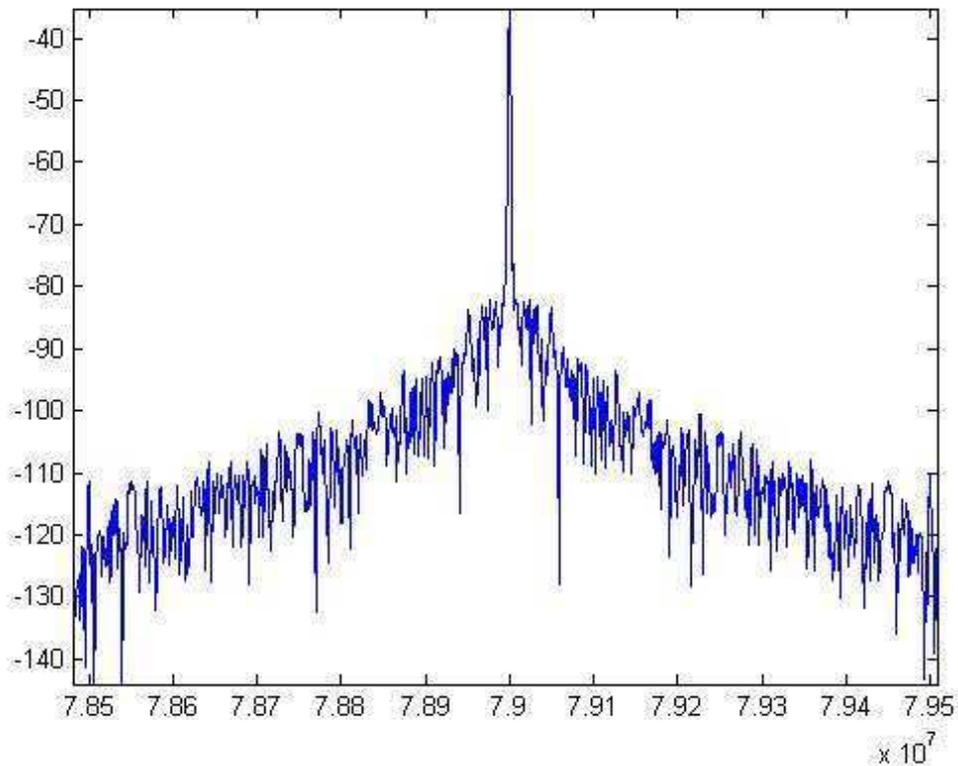


Figure 59 Simulation result of the prescaler output PSD of our model.

7.1.5 Simulation results for sigma-delta modulator

7.1.5.1 20MHz PLL

To test the behaviour of the sigma delta modulator, let us connect it to a PLL model with a reference frequency of 1 MHz and divider ratio of 20 ($N(k)=20$). Figure 60 and Figure 61 shows the change of VCO control voltage with time. The ripple caused by the sigma delta modulator is clear.

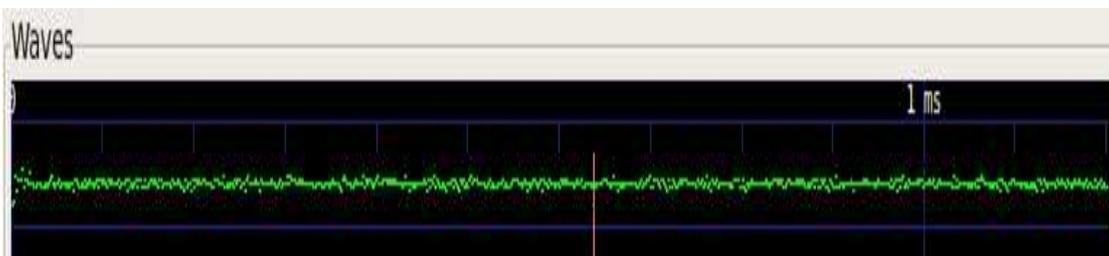


Figure 60 Vco control voltage of a 20MHz sigma-delta pll (zoom out).

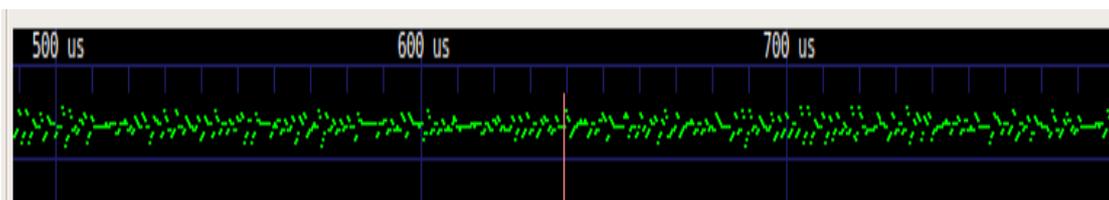


Figure 61 Vco control voltage of a 20MHz sigma-delta pll (zoom in).

Figure 62 shows the power spectrum density of the output of the PLL, the red one is the PSD of ideal sigma delta PLL output and the blue one is the spectrum of the same PLL with current mismatch added to the charge pump. The output frequency is 20.93 MHz which is the same with the theory calculation result. The low frequency noise is shifted to higher frequency and no fractional spurs exist. Also, it is clear that when add current mismatch to the charge pump, the low frequency spectrum goes up. This is because the modulator shifts the noise to higher frequency by random the divider ratio. When there is a current mismatch, the random divider ratio will cause an asymmetry charge pump output current which will affect the VCO tuning voltage and VCO output.

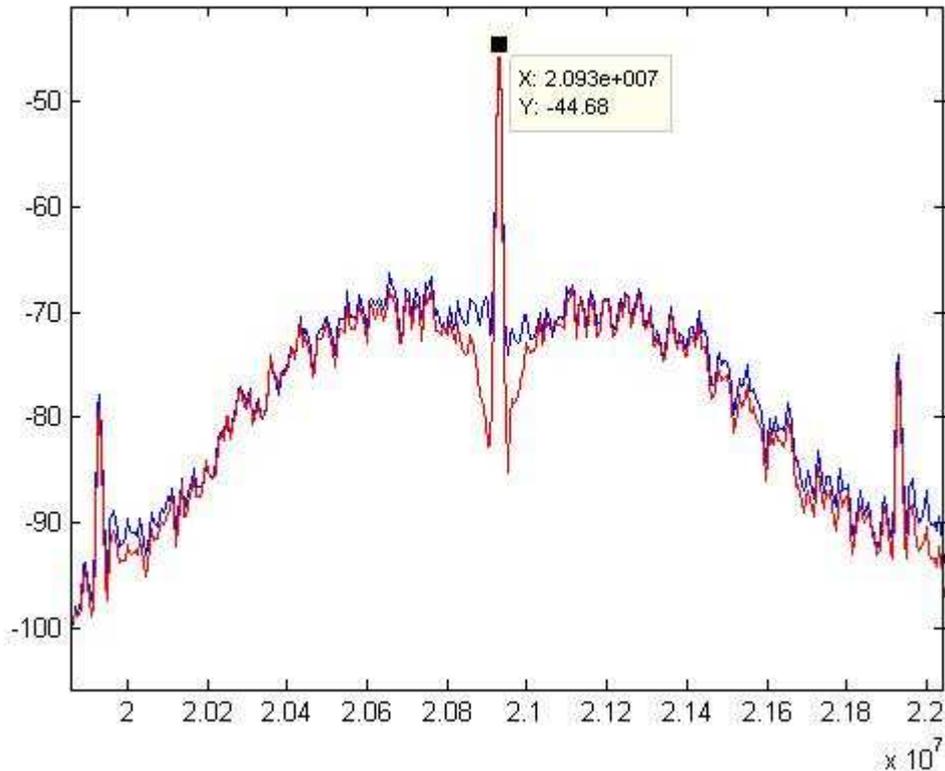


Figure 62 Power spectrum density of the output of the sigma-delta pll.

7.1.5.2 2.4GHz PLL

I also connect the sigma delta modulator to the 2.4 GHz PLL that includes all kinds of noise, the simulation result of the control voltage and PSD of the Prescaler output are shown in Figure 63, Figure 64 and Figure 65 respectively. Figure 66 shows the PSD of Prescaler output of a fractional PLL without sigma delta modulator. Compare Figure 65 and Figure 66, it is clear that the fractional spurs are eliminated by the sigma delta modulator. The noise performance has a 20 dB improvement.

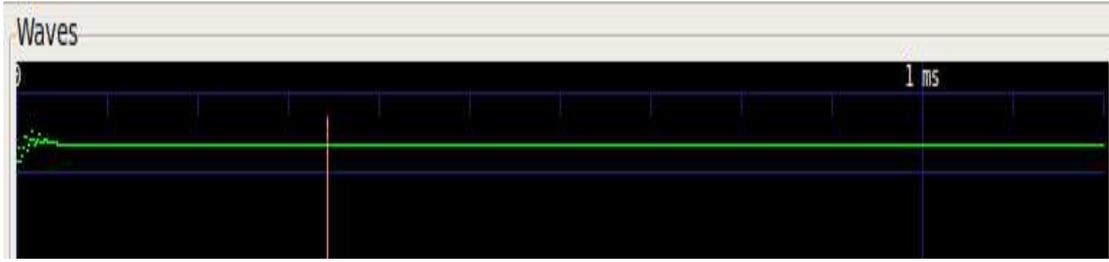


Figure 63 Control voltage of a 2.4GHz sigma delta pll (zoom out).

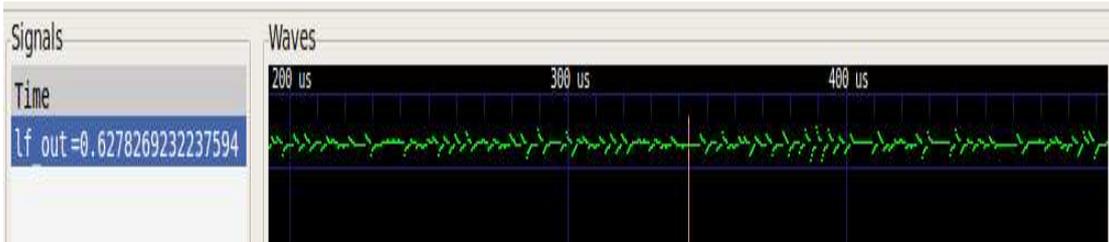


Figure 64 Control voltage of a 2.4GHz sigma delta pll (zoom in).

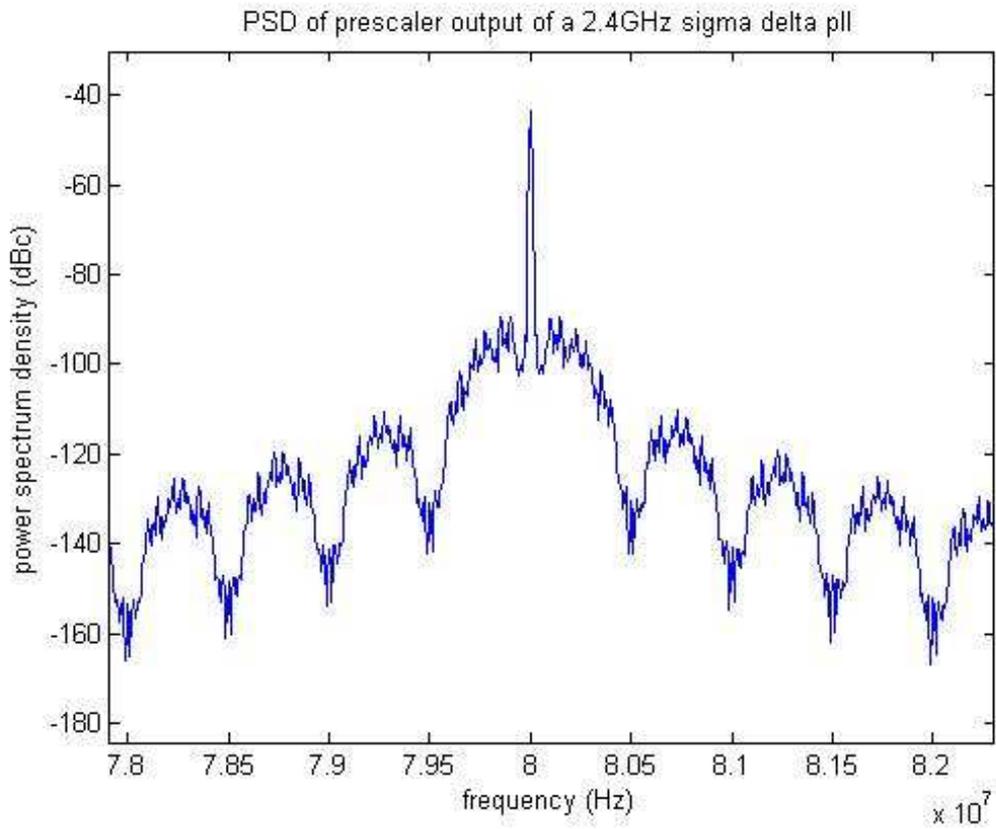


Figure 65 PSD of the prescaler output.

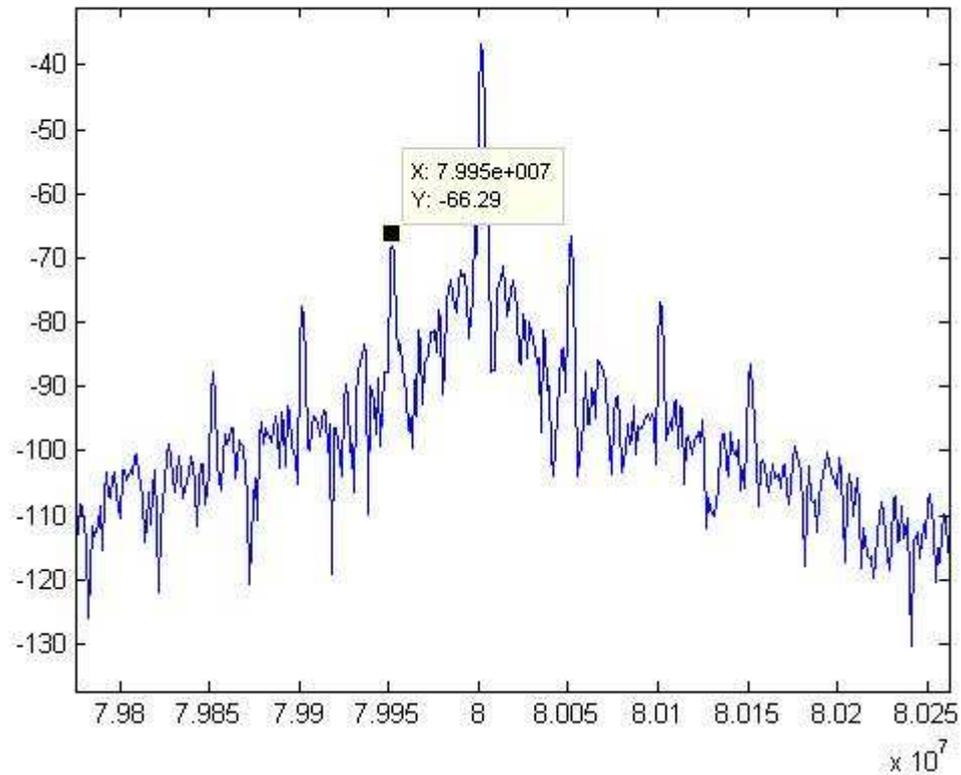


Figure 66 PSD of a fractional pll without sigma delta modulator.

7.1.6 Simulation results for the AC domain analysis of the PLL

The AC domain simulation results are shown in this part.

7.1.6.1 AC-domain open loop simulation result

The ac-domain open loop gain and phase simulation results are shown in Figure 67 and Figure 68 respectively. From the figures we can see that the phase margin is about $180-123.8=56.2$.

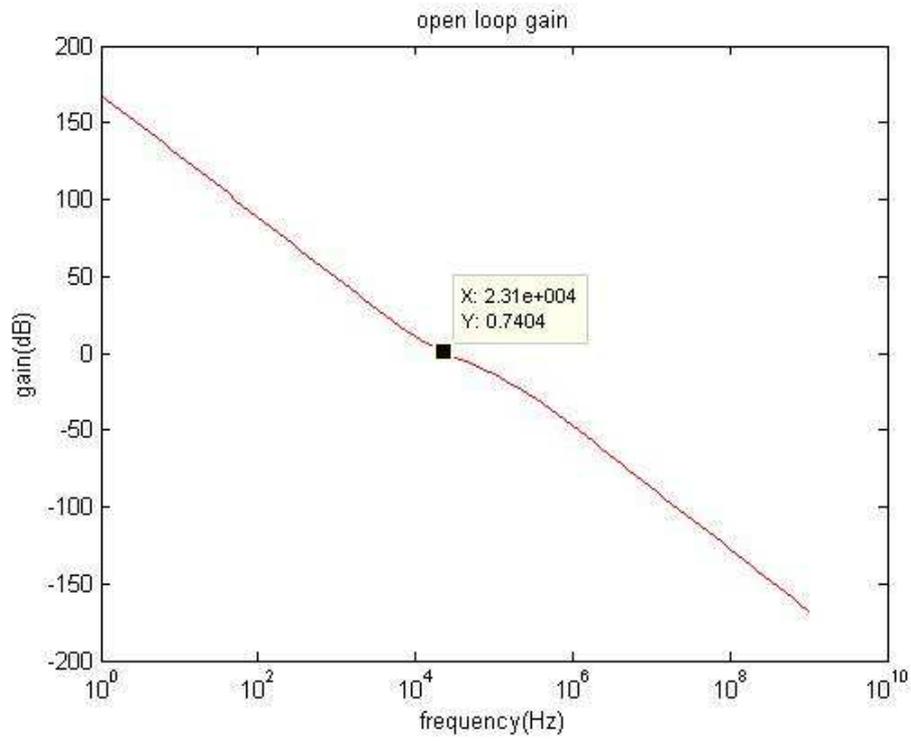


Figure 67 PLL open loop gain.

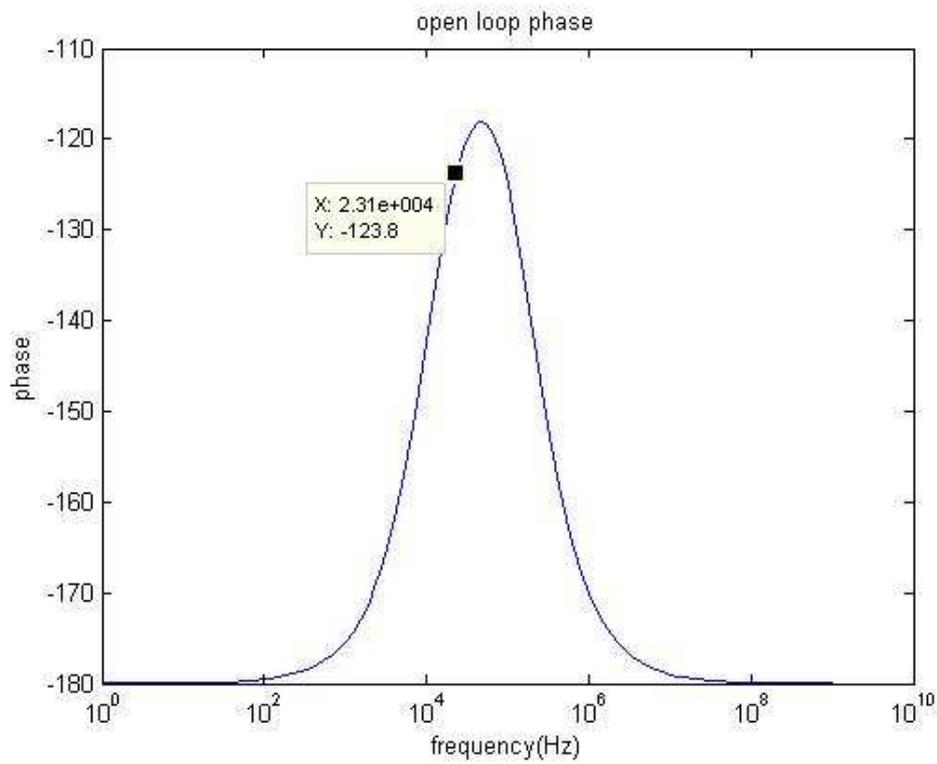


Figure 68 PLL open loop phase.

7.1.6.2 AC-domain close loop simulation result

The ac-domain close loop gain and phase simulation results are shown in Figure 69 and Figure 70 respectively. From the figure we can see that the loop bandwidth is about 35.11 KHz.

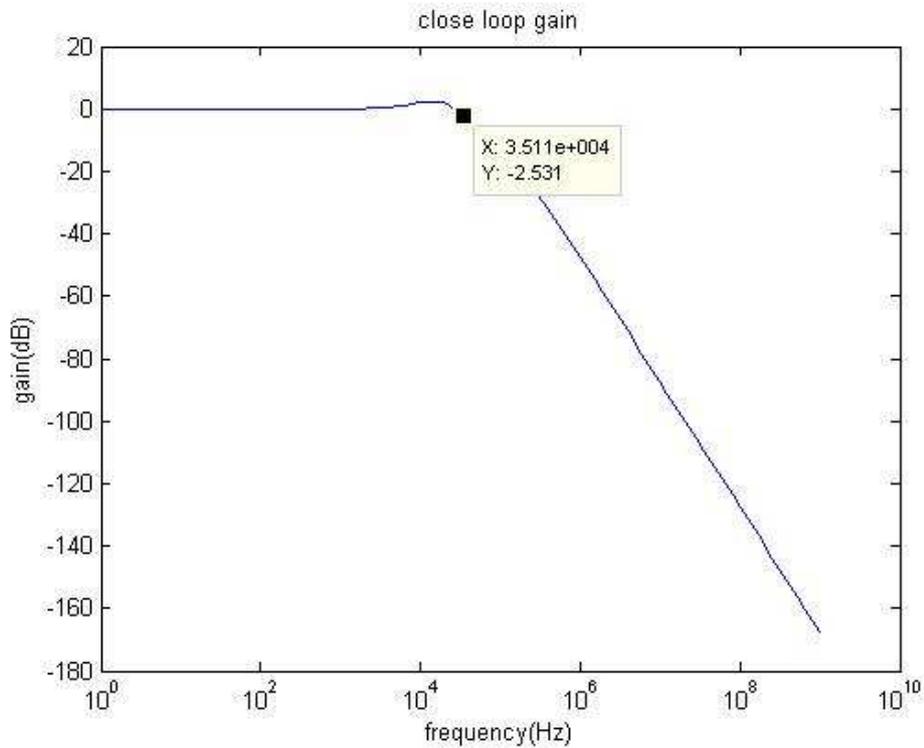


Figure 69 PLL close loop gain.

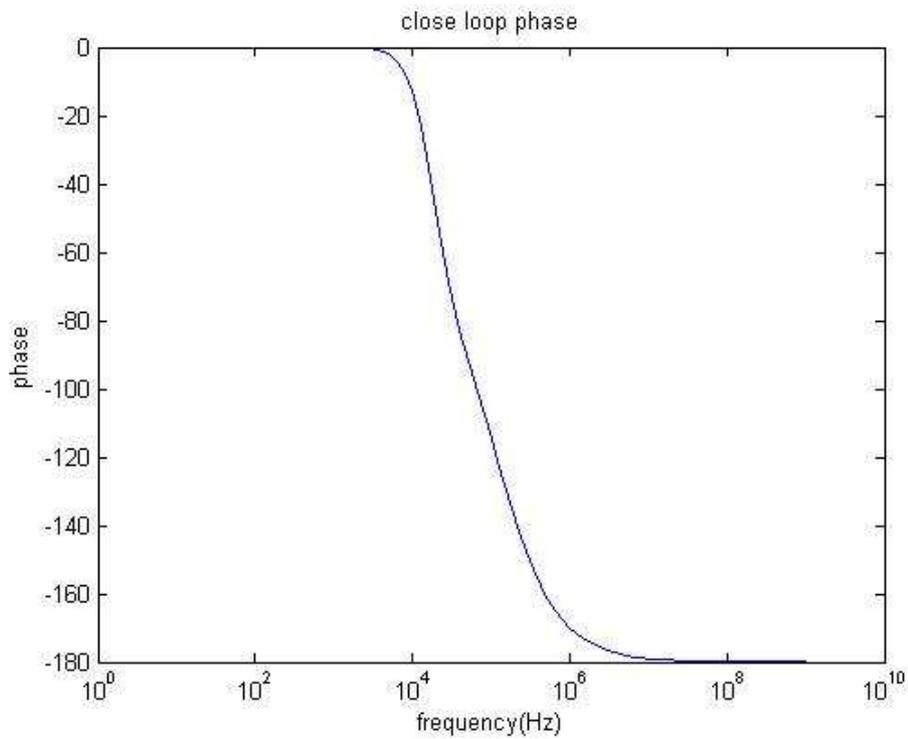


Figure 70 PLL close loop phase.

7.1.6.3 VCO noise reduction

Figure 71 is the PSD of the PLL that locked at 20 MHz and Figure 72 is the close loop gain of the PLL. It is clear that the low frequency noise of VCO is reduced and the width of the "skirt" is same with the calculation bandwidth of the PLL (about 35 KHz). This proves that the PLL works well.

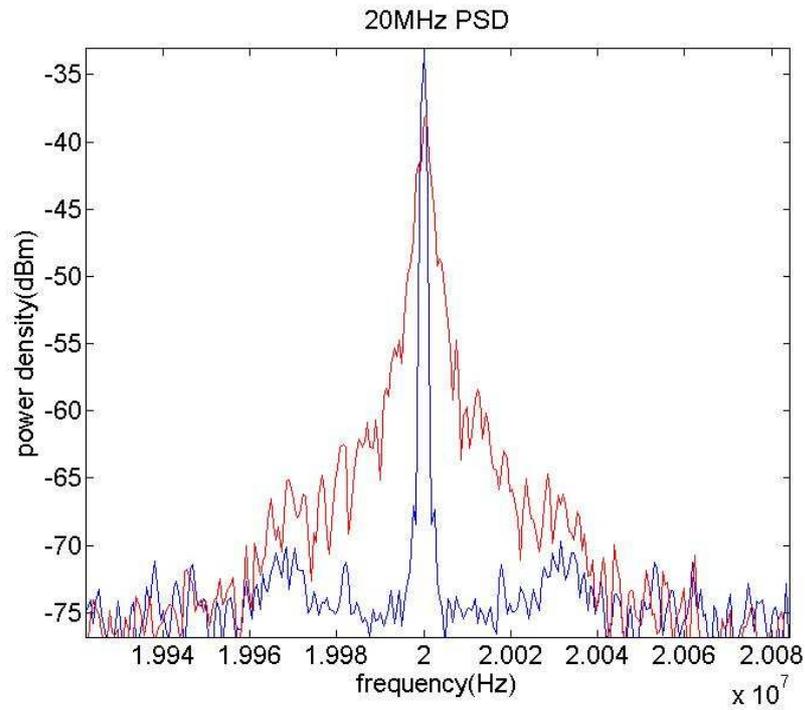


Figure 71 Low frequency noise distortion.

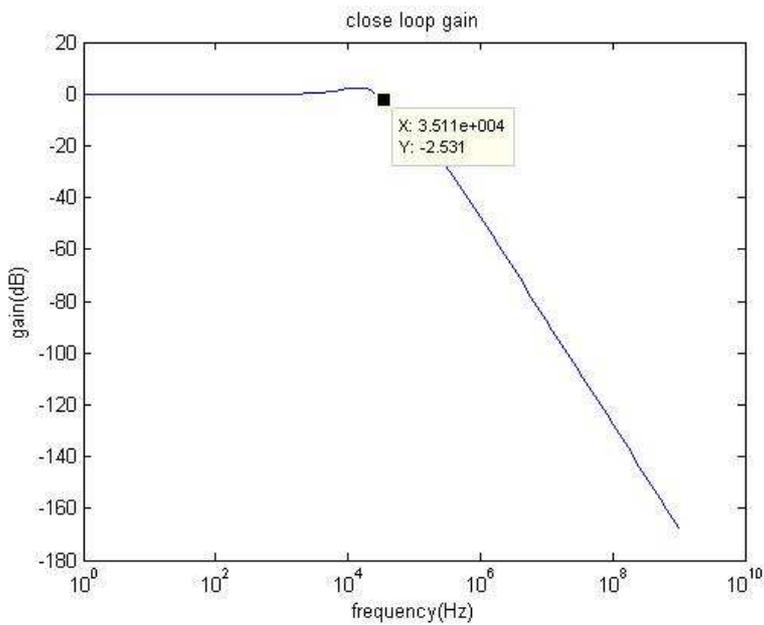


Figure 72 Bandwidth of the PLL.

7.1.7 Simulation results for PA-FLL

Figure 73 shows the simulation results of PA-FLL model. In the figure, C1 is the output of PFD, vc is the control bits value for DCO, vc1 and vc2 are the voltage of C1 and C2, state indicates the working phase of FLL (0=P1, 1=P2, 10=P3), rst and sample are reset and sample signals respectively, when sample signal changes, the system will sample the voltage in C1 and C2. We can see from the figure that the waveform is right according to the analysis in section 5.1.

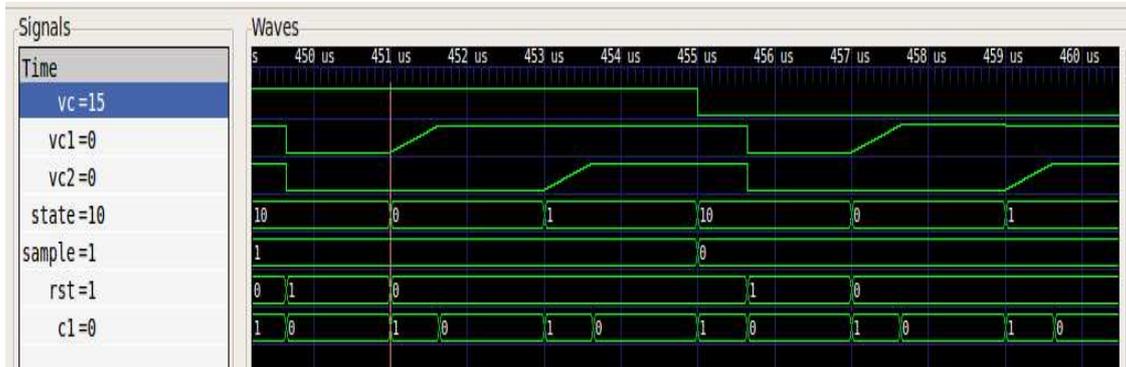


Figure 73 Simulation results for PA-FLL.

Figure 74 shows the wave form of reference (a) and the output of divider (b), we can see that the FLL can only lock the frequency but not the phase.



Figure 74 Simulation results for reference (a) and the output of divider (b).

7.1.8 Simulation results for FLL type-2

The FLL waveform is shown in Figure 75. In the figure, a, sig22, sig32 are the control signals of FVC1. b, sig2, sig3 are the control signals of FVC2. Temp1 and temp2 are the voltage of C1 in 2 FVCs. Vc1 and vc2 are the output of 2 FVCs. vc is the VCO control voltage. From the figure, we can see that the waveform is correct according to the analysis in section 5.2.

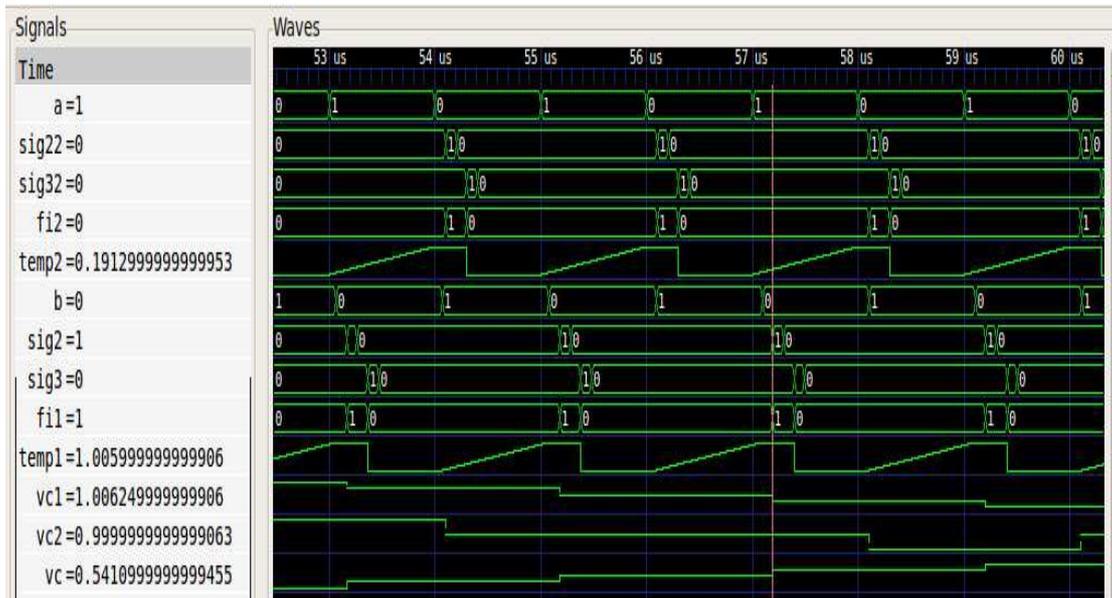


Figure 75 Waveform of the FVC (systemic simulation result).

Figure 76 shows the waveform of reference (a) and the output of divider (b), we can see that the FLL can only lock the frequency but not the phase.

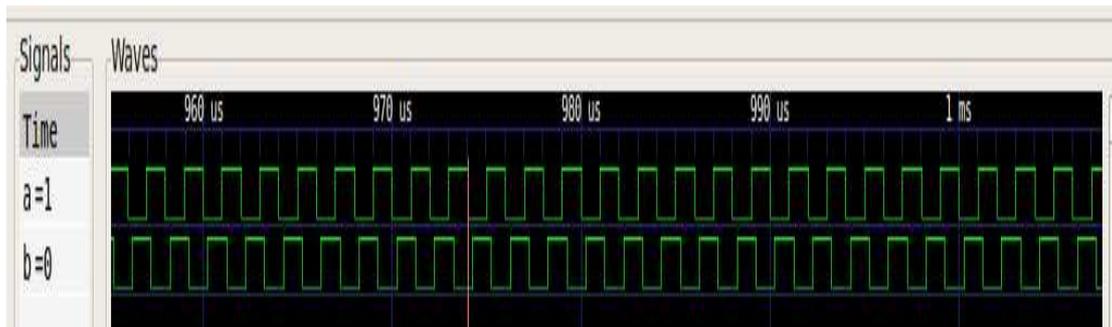


Figure 76 Waveform of reference (a) and the output of divider (b).

Figure 77 shows the waveform of control voltage, it can be seen from the figure that the settling time is about 100 us.

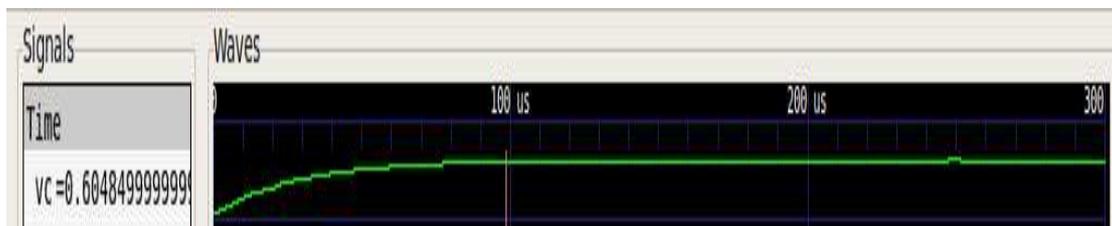


Figure 77 Waveform of control voltage.

7.2 Simulation results for the Analog System

In this section, the simulation results of the analog system will be shown. The parameters for PLL are the same as the previous section (2.4GHz). At the beginning, the PLLs will take 70us to tune the transmitter and receiver frequency. After that, the PLL of the receiver part will be switch off. Then

the algorithm for critical current searching and optimal slope searching will be executed. After that, the data start to transfer.

7.2.1 The simulation results for PLLs

Figure 78 shows the tuning voltage of the transmitter and receiver PLLs. The first signal is the transmitter PLL tuning voltage and the second one is that of the receiver. We can see that the settling time for both PLL is about 70us. As a result, the first 70us is used by PLLs to tune the frequency of the transmitter and receiver. Figure 79 is the zoom in figure of the two signals. We can see that after 70 us, the receiver PLL is switched off and the tuning voltage of the receiver part going down continuously because of the leakage current.

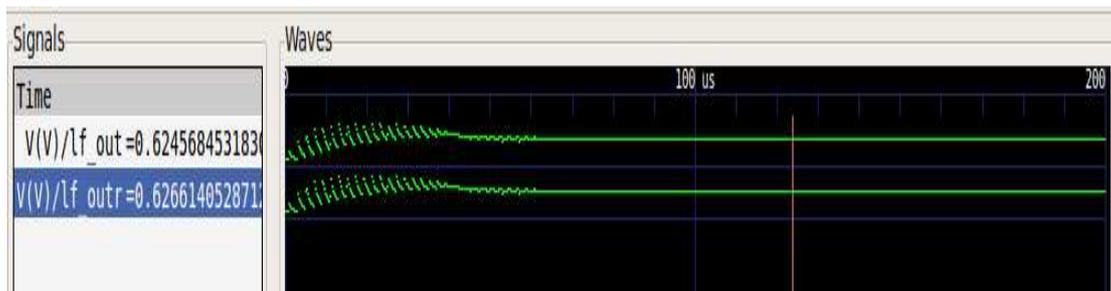


Figure 78 Tuning voltage of the PLLs.



Figure 79 Tuning voltage of the PLLs.

7.2.2 The simulation results of the transmitter

Figure 80 shows the simulation results of the transmitter. The first signal is the transmitted data, the second signal is the mixer output and the last signal is the output of the power amplifier. We can see that when the data is one, the output of the mixer is the PLL output. When the data is 0, the output of the mixer is 0. The PA just amplifies the signal from the mixer.

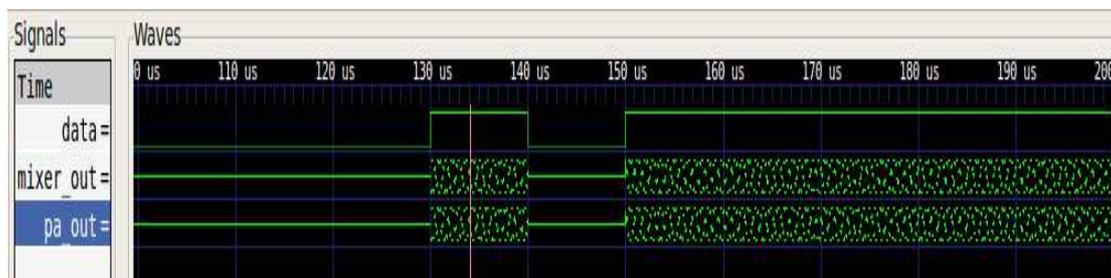


Figure 80 Simulation results of transmitter.

7.2.3 The simulation results of the receiver

Figure 81 shows the simulation results of the receiver. In the figure, the first signal is the output of the receiver VCO, the second one is the envelop detector output, the third one is the output of the low pass filter, the forth one is the VGA output and the last one is the output of ADC. Figure 82 is

the zoom in Figure 81. From the figure, we can see that the output of the receiver VCO is a high frequency signal. After the envelop detector and the low pass filter. The envelope of the signal is carried out. The VGA is just a multiplier which amplifies the output signal of the low pass filter. And the ADC changes the analog signal to digital signal. Figure 83 is the simulation results from Cadence. The first signal is the quenching voltage, the second one is the output voltage from envelop detector, the third one is the receiver oscillator output voltage and the last one is the input of the LNA. From the figure, we can see that the simulation results are similar between the Cadence and the SystemC/SystemC-AMS model.

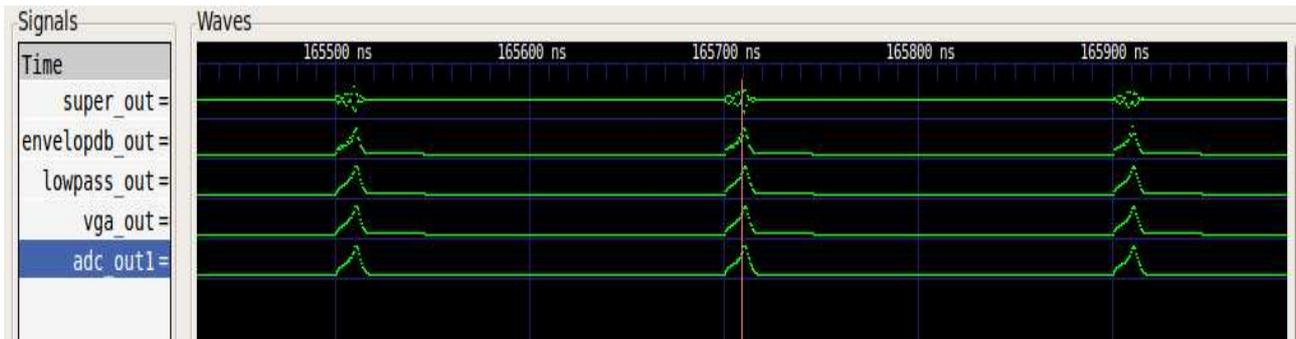


Figure 81 Simulation results of receiver.

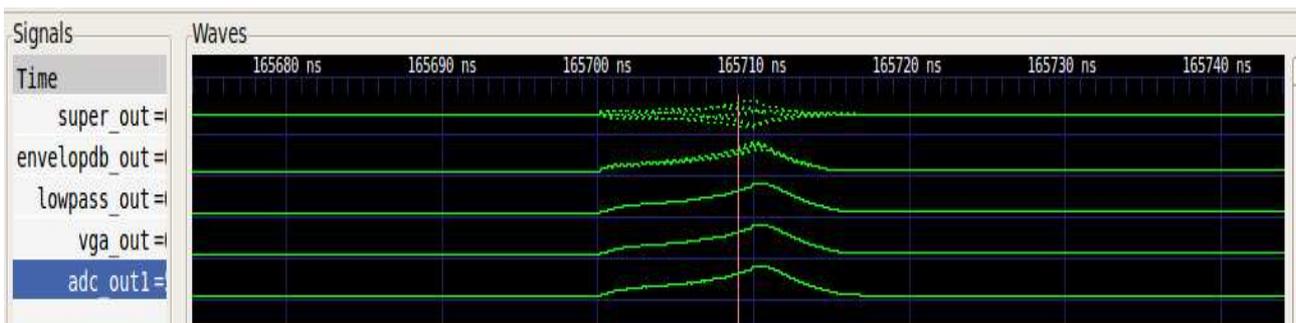


Figure 82 Zoom in figure of Figure 81.

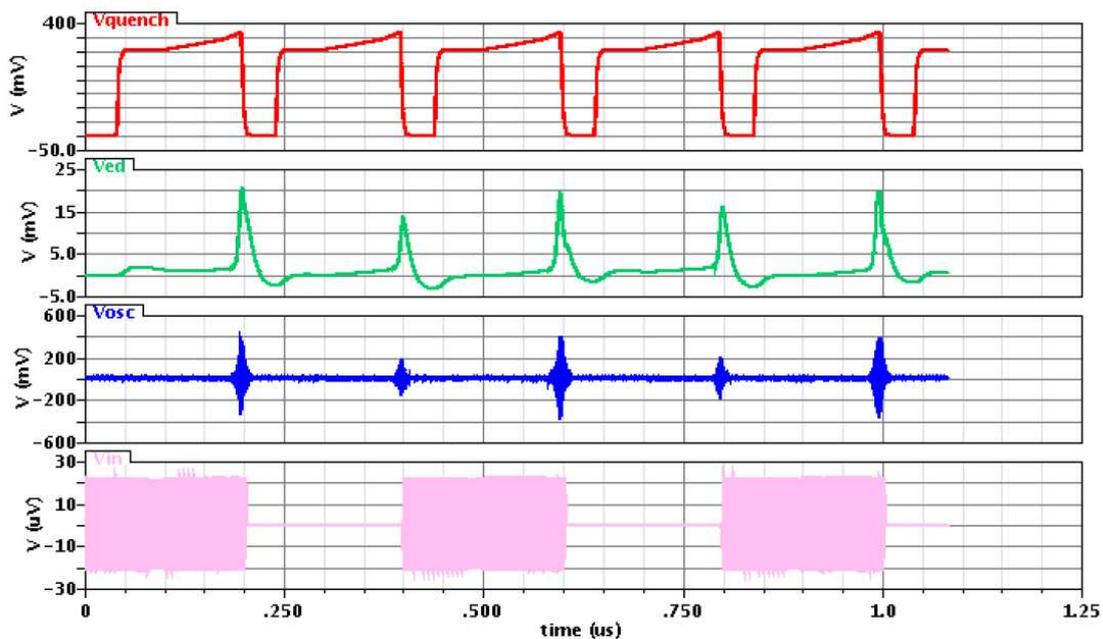


Figure 83 Simulation result from Cadence[5].

7.2.4 Simulation results for critical current searching

Figure 84 shows the simulation results for critical current searching. In the figure, the first signal is the quenching current and the second one is the receiver VCO output. We can see from the figure that at first, the VCO does not oscillate so the quenching current increase at the second cycles, and the VCO oscillate. So the quenching current decreased. After several iterations, the critical current is found.

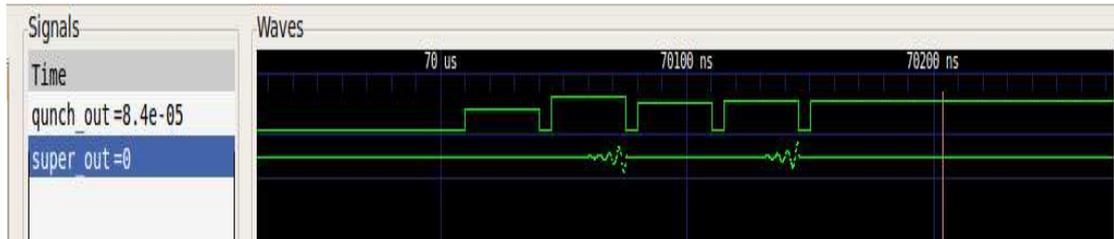


Figure 84 Simulation results of critical current searching.

7.2.5 Simulation results for optimal slope searching

Figure 85 shows the simulation results of the optimal slope searching. From the figure, we can see that the slope of the quenching current change every cycle according to the VCO output. Finally the optimal slope is found. The algorithm is nearly the same with critical current searching.

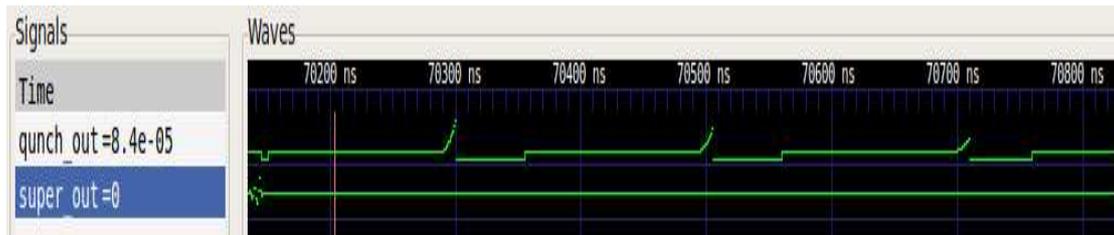


Figure 85 Simulation results of optimal slope searching.

7.3 Simulation results for digital baseband

7.3.1 The transmitter simulation results

Figure 86 shows the simulation results of the transmitter digital baseband, the upper signal is the data before the PPM coding and the lower signal is the data after coding. From the figure, it is clear that every 4 bits data are represented by 16 chips with only one “1”. The position of “1” is related to the value of the 4 bits.

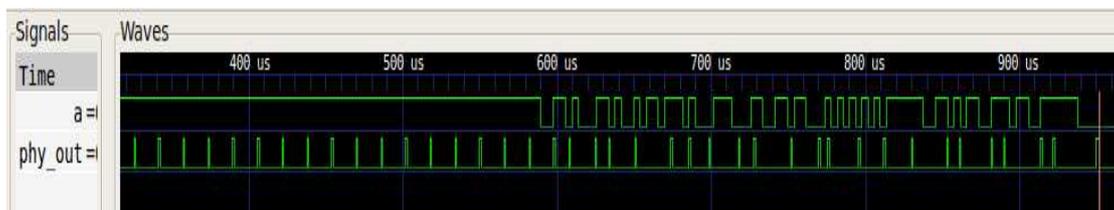


Figure 86 Simulation results for transmitter digital baseband.

7.3.2 Threshold estimation

Figure 87 shows the simulation results for the threshold estimation. From the figure, we can see that the “mean” is 7.875 and the variance is 4.71875. As a result, the final threshold is set to $7.875 + 4.71875 + 16 = 28.59375$.

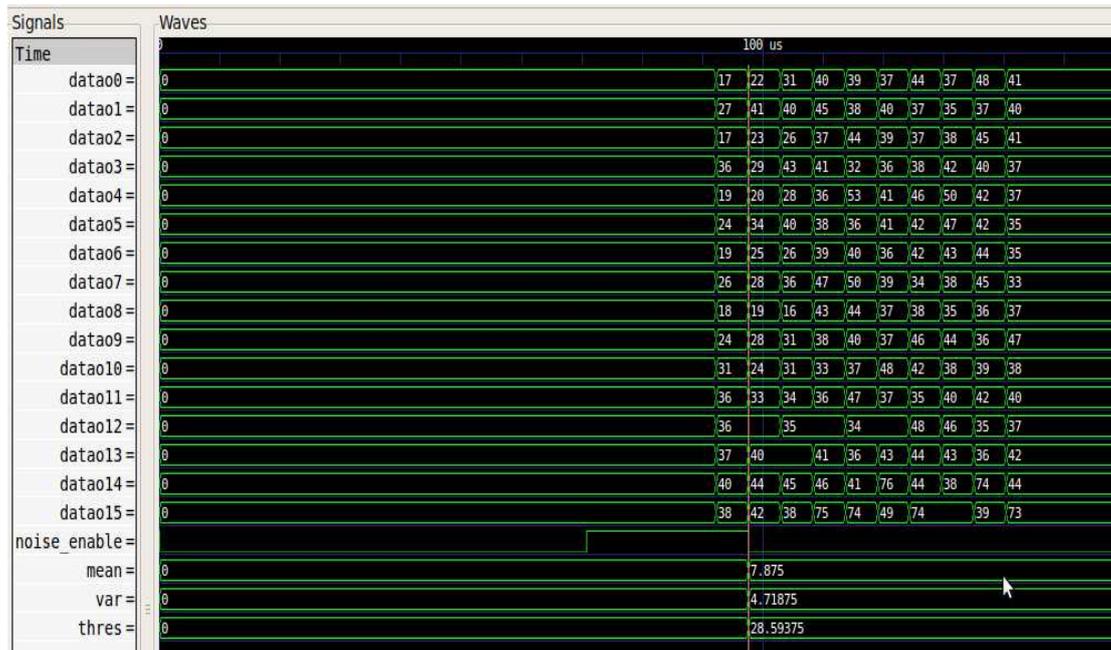


Figure 87 Simulation results for noise estimation.

7.3.3 Simulation results for detection

Figure 88 shows the simulation results for detection. From the figure, we can see that when the data signal is coming, the signal “detection_confirm” will become one and then the signal “detection_enable” will become 0. This indicated that the detection step works well and the data is detected.

7.3.4 Simulation results for confirmation

Figure 89 shows the simulation results for confirmation. The confirmation step will collect 2 groups of samples from the buffer and calculate the mean of the maximum difference of the two groups. If the mean is larger than the threshold, the data is confirmed. From the figure, we see that the data is confirmed.

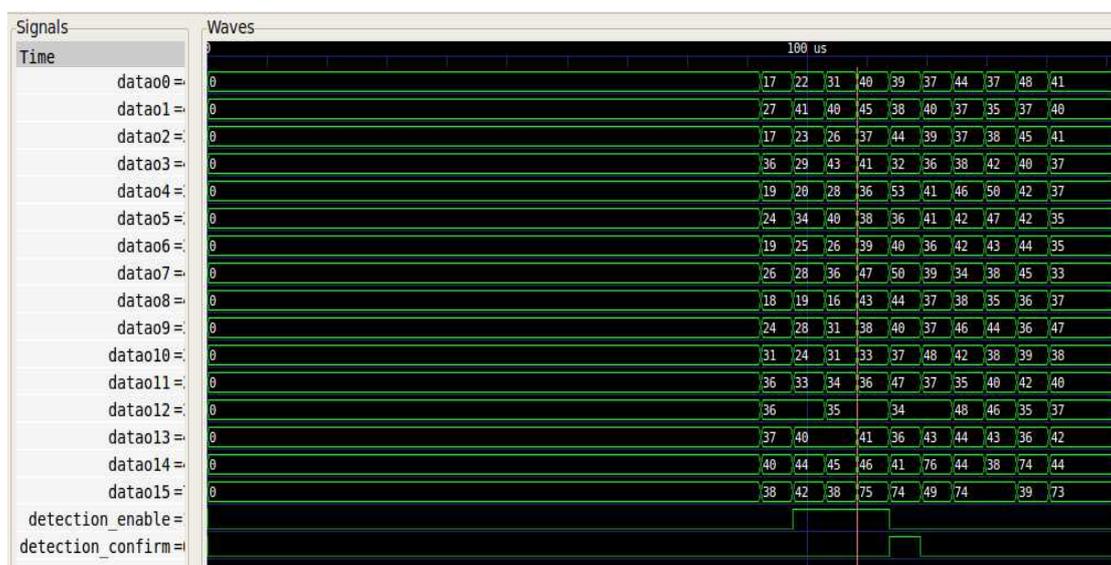


Figure 88 Simulation results for detection.

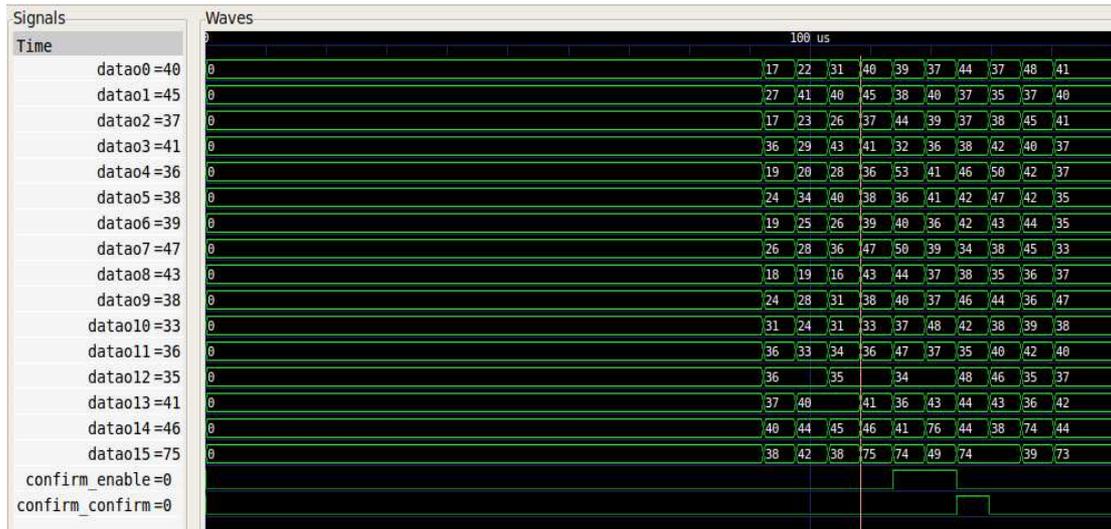


Figure 89 Simulation results for confirmation.

7.3.5 Simulation results for timing

Figure 90 shows the simulation results for timing. From the figure, we can see that the position of maximum value of each group is store in max0, max1, max2 and according to the value of these three numbers, the value of offset can be decided. In this case, the offset is set to “2F”. The calculation of the offset please refers to 6.2.5.



Figure 90 Simulation results for timing.

7.3.6 Simulation results for SFD detection

Figure 91 shows the simulation results for SFD detection. From the figure, we can see that the detector will continuously collect data from buffer and decide the maximum position of the data. If the positions of the maximum values of 2 continuous groups are “9” and “10” or “10” and “9”, the SFD is detected.

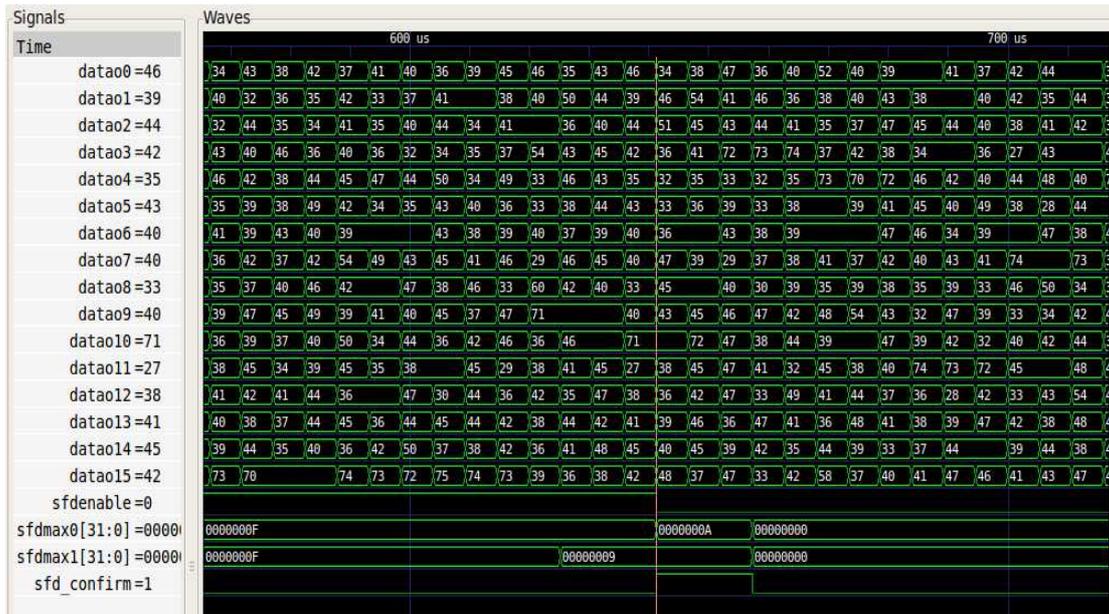


Figure 91 Simulation results for sfd detection.

7.3.7 Simulation results of data collection

The data collection is used to restore the data from the encoded PPM signal. The simulation results for data collection are shown in Figure 92. The decoding system will collect the data from buffer (16 for 1 group). Then it will decode according to the position of the maximum value of the group. Then the decoded data will be sent out. Every time there is a data being sent out. There will be a rising edge in the signal “datadoa”.



Figure 92 Simulation results for data collection.

7.3.8 Data verified

Figure 93 and Figure 94 show the value of transmitted data and received data. From the figure, we can see that the data is correct received.

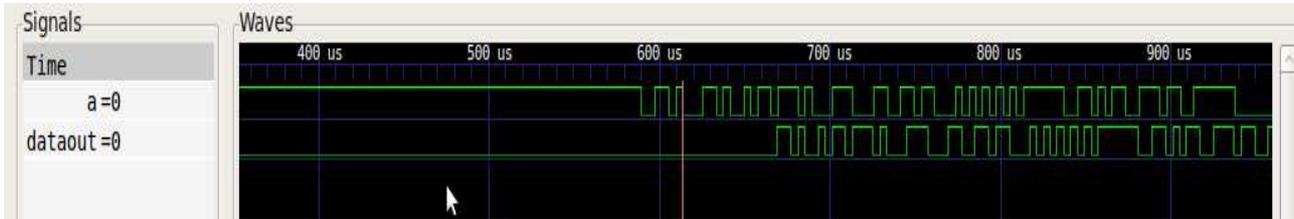


Figure 93 Transmitted data and received data.

```

transmitted data is: 00100111    received data is: 00100111
correct!
total: 34    error: 0
transmitted data is: 11110010    received data is: 11110010
correct!
total: 35    error: 0
transmitted data is: 00011001    received data is: 00011001
correct!
total: 36    error: 0
transmitted data is: 11111000    received data is: 11111000
correct!
total: 37    error: 0
transmitted data is: 00111000    received data is: 00111000
correct!
total: 38    error: 0
transmitted data is: 00110001    received data is: 00110001
correct!
total: 39    error: 0
transmitted data is: 10000011    received data is: 10000011
correct!
total: 40    error: 0
one package done!!!
transmitted data is: 00111100    received data is: 00111100
correct!
total: 41    error: 0
transmitted data is: 01011100    received data is: 01011100
correct!
total: 42    error: 0
transmitted data is: 01100110    received data is: 01100110
correct!
total: 43    error: 0
transmitted data is: 10110100    received data is: 10110100
correct!
total: 44    error: 0
transmitted data is: 00000010    received data is: 00000010
correct!
total: 45    error: 0
transmitted data is: 01010010    received data is: 01010010
correct!
total: 46    error: 0
transmitted data is: 01000101    received data is: 01000101
correct!
total: 47    error: 0
transmitted data is: 11010111    received data is: 11010111
correct!
total: 48    error: 0
transmitted data is: 10000001    received data is: 10000001
correct!
total: 49    error: 0
transmitted data is: 10110111    received data is: 10110111
    
```

Figure 94 Transmitted data and received data.

7.3.9 Compare with measurement results

The PPM modulation is also implemented on VIRTEX-5 FPGA platform which is shown in Figure 95. Figure 96 and Figure 97 show the measurement results of the packet error rate of the OOK and

PPM. Figure 98 shows that of the simulation results. From the figure, we can see that the simulation results are similar to the measurement results and the PPM modulation has less packet error rate than the OOK modulation. Table 1 is the summary of the results, the difference between the simulation results and measurement results maybe caused by too little packages are used (1000) to calculate the error rate. The simulation time for 1000 packages (about 100ms) is about 14 hours.

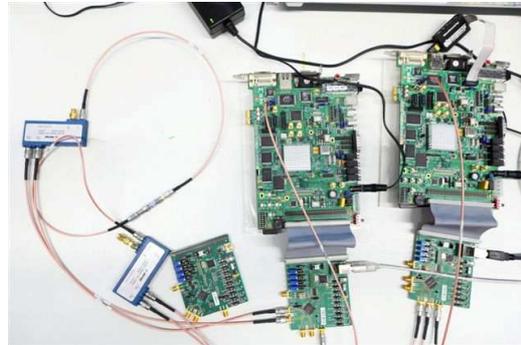


Figure 95 VIRTEX-5 FPGA platform

```
miss detection rate : 0.003
false alarm rate   : 0.268
Packet error rate  : 0.325
Bit error rate     : 0.012738
number of bers for cal: 729
```

Figure 96 Measurement results of package error rate of the OOK

```
miss detection rate : 0
false alarm rate    : 0.067
Packet error rate   : 0.101
Bit error rate      : 0.00031102
number of bers for cal: 933
```

Figure 97 Measurement results of package error rate of the PPM

```
total: 1000 error: 144
one package done!!!
```

Figure 98 Simulation results of package error rate of the PPM

| Measurement/simulation status | Packet error rate |
|----------------------------------|-------------------|
| OOK measurement (1000 packages) | 32.5% |
| OOK measurement (10000 packages) | 34.4% |
| PPM measurement (1000 packages) | 10.1% |
| PPM measurement (10000 packages) | 11.89% |
| PPM simulation (1000 packages) | 14.4% |

Table 1 Summary of the simulation / measurement packet error rate

7.3.10 Improvement of the PPM modulation

The PPM modulation represented in this thesis is implemented with VHDL and the packet errors dramatically reduced. The table below shows the measured packet error rate with attenuation of 60dB, 55dB and 50dB. From the table, we can see that the PPM modulation have a better performance under all three attenuation.

| | 60dB | 55dB | 50dB |
|-----|-------|-------|-------|
| OOK | 34.4% | 27.3% | 26.2% |
| PPM | 11.9% | 9.0% | 3.1% |

Table 2 The improvement of PPM

8 Conclusion

A new and precise mixed-signal model for a super-regenerative radio system based on the recent published language SystemC/SystemC-AMS is presented in this thesis. The simulation time of the model is dramatically reduced so that the PLL is able to be included into the model to simulate the frequency difference between the transmitter and receiver. Then the distortion caused by this difference can be simulated.

Also, the digital baseband is first included into the model, the digital baseband can be divided into two parts – the transmitter part and the receiver part, the transmitter part realizes the PPM modulation and the receiver part realizes the PPM synchronization algorithm and the PPM decoding. Also it is responsible for the verification of the received data.

Moreover, the effects of various practical imperfections are included in this model. The imperfections include high order harmonics, VCO noise, Charge Pump current mismatch, leakage current and so on.

With this model, we can not only see the performance of each block, but also the performance of the entire system. The model is verified by a number of simulation tools as well as the measurement results. These comparison results demonstrate that the model is able to precisely reflect the real system behaviour with fast simulation time. They also validate that the SystemC/SystemC-AMS is an efficient and powerful language for mixed-signal modelling.

Paper published:

I have published a paper “A Fast and Accurate SystemC-AMS model for PLL” in conference MIXDES 2011.

9 References

- [1] J. Bhasker, A SystemC Primer, Second Edition
 - [2] Open SystemC Initiative (10-02-2010), “SystemC AMS extensions User’s Guide” [online]. Available: <http://www.systemc.org/downloads/standards/>
 - [3] J.Y. Chen, M.P. Flynn and P. Hayes, “A fully integrated auto-calibrated super regenerative receiver in 0.13- μ m CMOS”, IEEE J. of Solid-State Circuits, vol. 42, no. 9, September 2007.
 - [4] F. X. Moncunill-Geniz, P. Palá-Schönwälder, and O. Mas-Casals, “A generic approach to the theory of superregenerative reception,” IEEE Trans. Circuits Syst. I, vol. 52, no. 1, pp. 54–70, Jan. 2005.
 - [5] M. Vidojkovic, “Design and implementation of a super regenerative RF receiver front-end for the WBAN IEEE 802.15.6 main radio: Tape out February 2009”, Holst Centre, Netherlands, Tech. Rep. TN-09-WATS-TP2-033, Oct. 2009.
 - [6] P. Harpe, C. Huang, S. Rampu, and M. Vidojkovic, “Analog BAN Radio Blocks – Designs for Oct09 and April10 Tapeouts”, Holst Centre, Netherlands, Tech. Rep. TN-10-WATS-TP2-050, Jun. 2010.
 - [7] S. Bittner, S. Krone, and G. Fettweis, “Tutorial on Discrete Time Phase Noise Modeling for Phase Locked Loops” [online]. Available: <http://www.vodafone-chair.com/staff/bittner/>.
 - [8] Dr. J. Romme, “Phase Noise in Phase-Aligned Frequency Locked Loops”, Holst Centre, Netherlands, Oct. 2010.
 - [9] A. Djemouai, M. Sawan and M. Slamani*, “A 200 MHZ FREQUENCY-LOCKED LOOP BASED ON NEW FREQUENCY-TO-VOLTAGE CONVERTERS AE’PROACIH”
 - [10] Hung Tien Bui, Member, IEEE, and Yvon Savaria, Fellow, IEEE, “Design of a High-Speed Differential Frequency-to-Voltage Converter and Its Application in a 5-GHz Frequency-Locked Loop”
-