Teaching PLL Fundamentals Using MATLAB/Simulink

Daniela Antonova Shehova and Peter Ivanov Yakimov

Abstract – **PLL is employed in a wide array of electronic and communications equipment and understanding its principles is of a great importance. Simulation is an obvious solution for teaching PLL fundamentals. MATLAB/Simulink is a very powerful block simulation environment, most capable for PLL. The paper discusses an approach for teaching enabling the students to obtain sustainable knowledge about PLL.**

Keywords – **PLL, Simulink, MATLAB, simulation, teaching**

I. INTRODUCTION

Phase-locked loop (PLL) is a feedback loop which locks two waveforms with same frequency but shifted in phase [1]. The fundamental use of this loop is in comparing frequencies of two waveforms and then adjusting the frequency of the waveform in the loop to equal the input waveform frequency. Used to synchronize the phase of two signals, the PLL is employed in a wide array of electronic and communications equipment, including microprocessors devices such as radios, televisions, and mobile phones. The basic blocks of the PLL are a phase detector, a low-pass filter, a variable frequency oscillator, and a divider (Fig. 1).

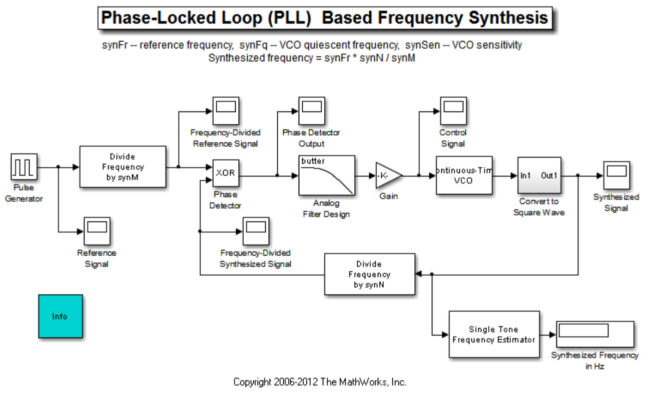

Phase-Locked Loop (PLL) Based Frequency Synthesis
synFr -- reference frequency, synFq -- VCO quiescent frequency, synSen -- VCO sensitivity
Synthesized frequency = synFr * synN / synM
Pulse Generator
Divide Frequency by synM
Reference Signal
Frequency-Divided Reference Signal
XOR
Phase Detector
Frequency-Divided Synthesized Signal
Phase Detector Output
butter
Analog Filter Design
-K-
Gain
Control Signal
Continuous-Time VCO
In1 Out1
Convert to Square Wave
Synthesized Signal
Divide Frequency by synN
Info
Single Tone Frequency Estimator
Synthesized Frequency in Hz
Copyright 2006-2012 The MathWorks, Inc.

Fig. 1. PLL general block diagram.

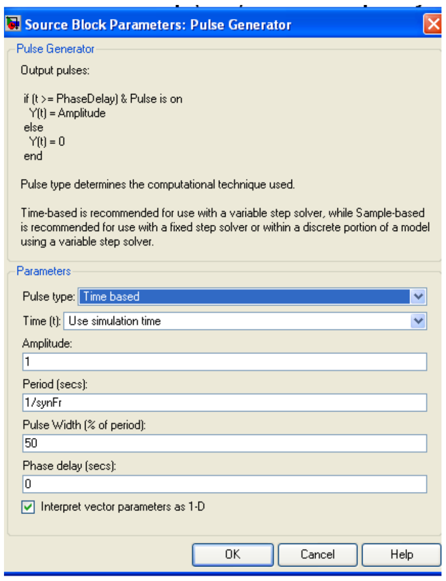
Obtaining sustainable knowledge about PLL requires understanding its fundamentals - loop components, loop response, loop stability, transient response, modulation response. Simulation is an obvious solution. Most often MATLAB will suffice for modeling and simulation. The use of Spice with behavioral level modeling capabilities may also be useful, e.g., XSpice via SIMetrix/SIMPLIS or PSpice. Simulink® is a block diagram environment for multidomain simulation and Model-Based Design [4]. It supports simulation, automatic code generation, and continuous test and verification. Simulink provides a graphical editor, customizable block libraries, and solvers for modeling and simulating dynamic systems. It is integrated with MATLAB®, enabling incorporation MATLAB algorithms into models and exportation simulation results to MATLAB for further analysis. MATLAB Simulink is a very powerful block simulation environment, most capable for PLL. Simulink behavioral simulation is much faster than circuit-level simulation, and as a result, there can be completed many simulations in one day, experimenting with different implementation ideas for the functional blocks. The behavioral simulations are instrumental in determining the block-level specifications that will satisfy a given set of top-level PLL specifications.

D. Shehova is with the Technical College by Plovdiv University "Paisiy Hilendarski", 4700 Smolyan, Bulgaria, e-mail: dani_shehova@abv.bg.
P. Yakimov is with the Department of Electronics, Faculty of Electronic Engineering and Technologies, Technical University of Sofia, 8 Kl. Ohridski Blvd, Sofia 1000, Bulgaria, e-mail: pij@tu-sofia.bg.

II. BASIC BLOCKS MODEL PARAMETERS SETTING

A. Pulse Generator

The Pulse Generator block generates the reference signal. It produces a periodic pulse train. The variable *synFr* denotes the frequency of the pulse train. The period of the pulse train is 1/*synFr*. To change the value of the period, the value of the variable *synFr* has to be changed so that the new value of *synFr* is used in all the blocks whose parameters reference the variable *synFr* (Fig. 2).


Source Block Parameters: Pulse Generator

Pulse Generator

Output pulses:

```
if (t >= PhaseDelay) & Pulse is on
  Y(t) = Amplitude
else
  Y(t) = 0
end
```

Pulse type determines the computational technique used.

Time-based is recommended for use with a variable step solver, while Sample-based is recommended for use with a fixed step solver or within a discrete portion of a model using a variable step solver.

Parameters

Pulse type: Time based

Time (t): Use simulation time

Amplitude: 1

Period (secs): 1/synFr

Pulse Width (% of period): 50

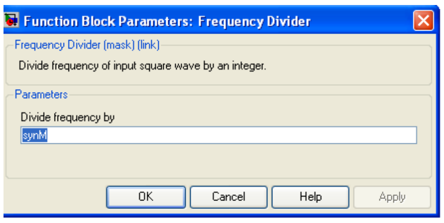
Phase delay (secs): 0

[✓] Interpret vector parameters as 1-D

Fig. 2. Pulse generator model parameter setting.

B. Divide Frequency subsytems

There are two divide frequency subsystems - divide frequency by *synM* and divide frequency by *synN*. The divide frequency by *synM* subsystem divides the frequency of the reference signal by the variable *synM*. The output of the block is a pulse train called the frequency-divided reference signal. This value determines the step of the output frequency setting. The divide frequency by *synN* subsystem divides the frequency of the synthesized signal by the variable *synN*. The output of this subsystem is called the frequency-divided synthesized signal. At steady state its frequency has the same value as the output one of the *synM* subsystem. The value of the divisor in these subsystems can be changed by changing the value of *synM* or *synN* (Fig. 3).


Function Block Parameters: Frequency Divider

Frequency Divider (mask) (link)

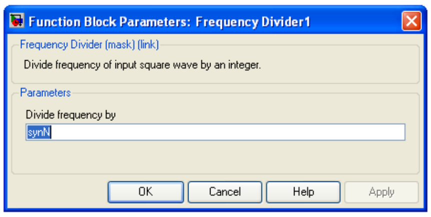
Divide frequency of input square wave by an integer.

Parameters

Divide frequency by

synM

Fig. 3a. *SynM* value setting.


Function Block Parameters: Frequency Divider1

Frequency Divider (mask) (link)

Divide frequency of input square wave by an integer.

Parameters

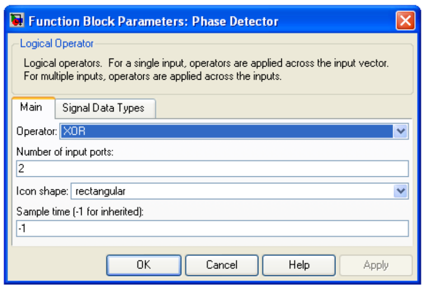
Divide frequency by

synN

Fig. 3b. *SynN* value setting.

C. Phase Detector

The Logical Operator block acts as a phase detector. It uses the XOR operation to compare the frequencies of the frequency-divided reference signal and the frequency-divided synthesized signal. At steady state, the signal is a pulse train with frequency two times higher than the both inputs. The reason for this is that both inputs to the block have equal frequencies, but they are out of phase by 1/4 of their period. As a result, the signal after the XOR operation is a periodic pulse train with double frequency (Fig. 4).


Function Block Parameters: Phase Detector

Logical Operator

Logical operators. For a single input, operators are applied across the input vector. For multiple inputs, operators are applied across the inputs.

Operator: XOR

Number of input ports: 2

Icon shape: rectangular

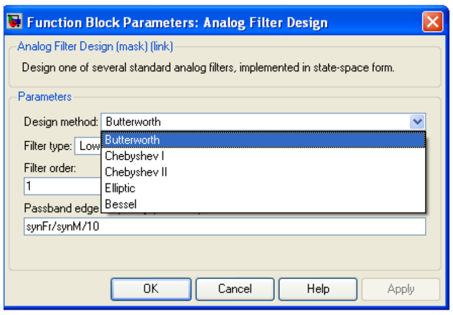
Sample time (-1 for inherited): -1

Fig. 4. Phase detector parameters setting.

D. Analog Filter Design

The Analog Filter Design block filters high frequencies out of the signal coming from the phase detector. The block uses a lowpass Butterworth filter. A higher-order filter or another filter type can be used to improve the stability of the synthesized signal (Fig. 5). In the steady state of the model, the amplitude of the block's output signal is approximately constant, with a value of 0.5. This is the average value of the output from the phase detector.

A Gain block multiplies the output signal from the Analog Filter Design block by a constant to produce the control signal.


Function Block Parameters: Analog Filter Design

Analog Filter Design (mask) (link)

Design one of several standard analog filters, implemented in state-space form.

Parameters

Design method: Butterworth

Filter type: Low

Filter order: 1

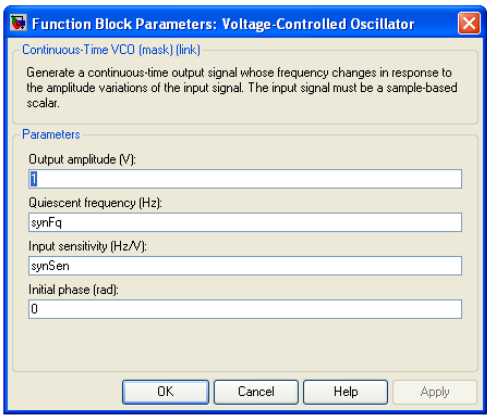
Passband edge

synFr/synM/10

Fig. 5. Analog filter design menu.

E. Voltage-Controlled Oscillator

The Continuous-Time VCO block generates the synthesized signal (along with the Convert to Square Wave subsystem) and adjusts the frequency of the synthesized signal according to the Voltage-Controlled Oscillator input signal. When the control signal is close to its steady-state, the Continuous-Time VCO block generates a signal whose frequency is close to *synFr***synN*/*synM*. If the output frequency drops, the control signal rises, boosting the frequency of the output signal. If the output frequency rises, the control signal falls, lowering the output frequency. The Quiescent frequency parameter is just the oscillation frequency, *synFq*. The difference between the block's output signal frequency and the quiescent frequency is proportional to the input signal, interpreted as voltage. The quiescent frequency is set to the variable *synFq*. This value can be changed in the quiescent frequency field, or by changing the value of *synFq* in the base MATLAB workspace (Fig. 6).


Function Block Parameters: Voltage-Controlled Oscillator

Continuous-Time VCO (mask) (link)

Generate a continuous-time output signal whose frequency changes in response to the amplitude variations of the input signal. The input signal must be a sample-based scalar.

Parameters

Output amplitude (V): 1

Quiescent frequency (Hz): synFq

Input sensitivity (Hz/V): synSen

Initial phase (rad): 0

Fig. 6. VCO parameters setting.

III. Study of Fractional-N Frequency Synthesis

As an example a study of a fractional-N frequency synthesis (Fig. 7) is chosen to explain the students the operation of PLL and to give them skills for work with MATLAB/Simulink [2].

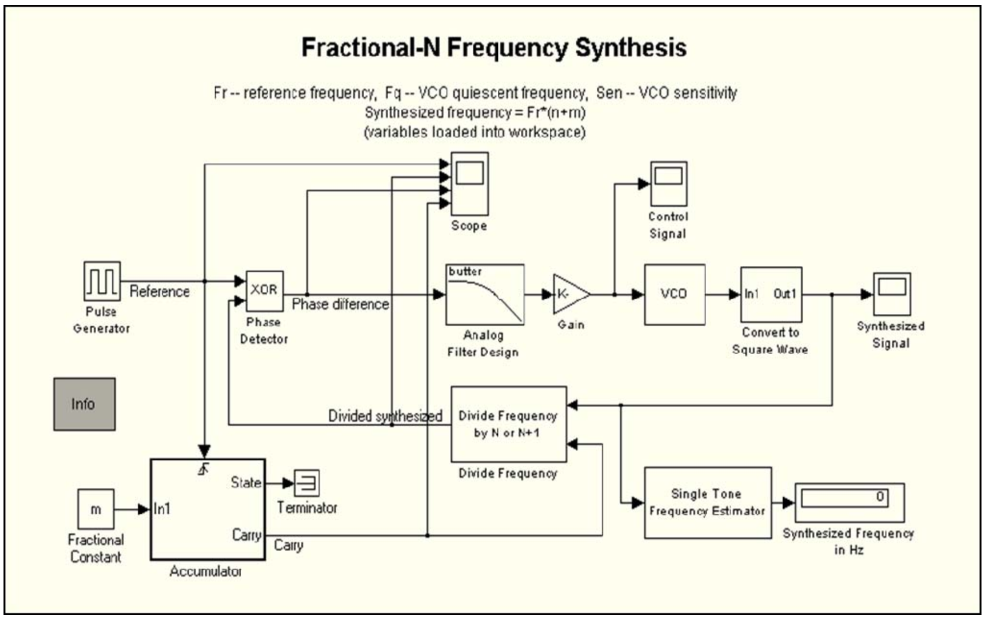

Fractional-N Frequency Synthesis
Fr -- reference frequency, Fq -- VCO quiescent frequency, Sen -- VCO sensitivity
Synthesized frequency = Fr*(n+m)
(variables loaded into workspace)
Scope
Control Signal
Pulse Generator
Reference
XOR
Phase Detector
Phase difference
butter
Analog Filter Design
K-
Gain
VCO
In1 Out1
Convert to Square Wave
Synthesized Signal
Info
Divided synthesized
Divide Frequency by N or N+1
Divide Frequency
m
Fractional Constant
In1
State
Carry
Accumulator
Terminator
Carry
Single Tone Frequency Estimator
0
Synthesized Frequency in Hz

Fig. 7. Fractional-N frequency synthesis model.

The model multiplies the frequency *synFr* of a reference signal by a constant *synN*+*synM*, to produce a synthesized signal of frequency *synFr**(*synN*+*synM*). A feedback loop maintains the frequency of the synthesized signal at this level. *SynN* is an integer and *synM* is a fraction between 0 and 1. Two subsystems in this example are not present in the Phase-Locked Frequency Synthesis model: Accumulator and Divide Frequency.

A. Accumulator

The Accumulator subsystem repeatedly adds the constant *synM* to a cumulative sum. While the sum is less than 1, the output labeled "Carry" is 0. At a time step when the sum becomes greater than or equal to 1, the carry output is 1 and the cumulative sum is reset to its fractional part. The fraction of the time when the carry output is 1 is equal to *synM*, while the fraction of the time when it is 0 is equal to 1-*synM* (Fig. 8).

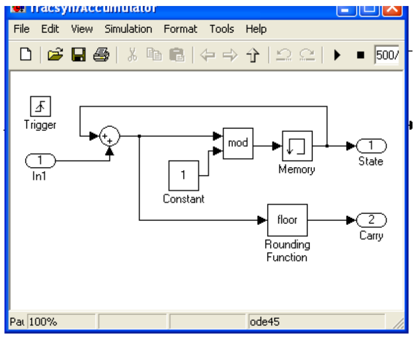
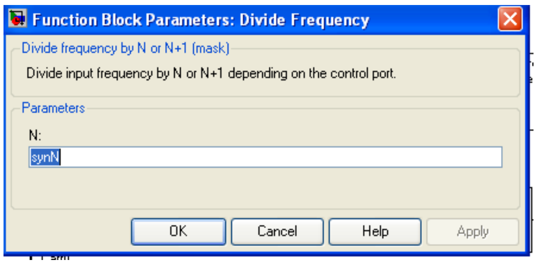


Fig. 8. Accumulator subsystem.

B. Divide Frequency

The Divide Frequency subsystem divides the frequency of the synthesized signal by *synN* when the output of the Accumulator subsystem is 0, and divides it by *synN*+1 when the output is 1 (Fig. 9). As a result, the average amount that frequency is divided by is:

$$(1-synM)*synN + synM*(synN+1) = synN + synM \qquad (1)$$


Function Block Parameters: Divide Frequency

Divide frequency by N or N+1 (mask)

Divide input frequency by N or N+1 depending on the control port.

Parameters

N:

synN

Fig. 9. Divide frequency parameters setting.

The students' task is to investigate the fractional-N frequency synthesis using Simulink according to the following plan:

1. Setting the basic parameters:

Parameter	Value
synM	0,1 ÷ 0,9
synN	10
Fr	10 MHz
Fq	101 ÷ 109 MHz

2. Performing simulations for different filter type and presenting the results for *Fq* [MHz] in table 1:

TABLE 1. SIMULATIONS RESULTS

synM	Butterworth	Chebyshev I	Chebyshev II	Elliptic
0,1	100,8	100,7	94,27	100,7
0,2	101,8	101,8	94,26	101,8
0,3	103,1	103	94,27	103,1
0,4	104,2	103,9	94,26	103,9
0,5	105	104,9	94,26	104,9
0,6	106	106	94,23	106
0,7	107	107,1	94,26	107,1
0,8	108	108	94,26	108
0,9	109	109,4	94,25	109,4

3. The presented in the table above results are visualized graphically using MATLAB. For this purpose the students create the following source code:

```
x=0.1:0.1:0.9;
y1=[100.8 101.8 103.1 104.2 105 106 107 108 109];
y2=[100.7 101.8 103.1 103.9 104.9 106 107.1 108 109.4];
y3=[94.27 94.26 94.27 94.26 94.26 94.23 94.26 94.26 94.25];
y4=[100.7 101.8 103 103.9 104.9 106 107.1 108 109.4];
plot(x, y1, '- g o', x, y2, '- b x', x, y3, '- r o', x, y4, '- m x',....
'MarkerFaceColor', 'k', 'MarkerSize', 5, 'LineWidth', 3)
grid on, xlabel('m'), ylabel('Fq, MHz')
title('Frequensy Synthesis ')
legend('y1(Butterworth)','y2(ChebychevI)','y3(ChebychevII)',
'y4(Elliptic)')
```

where the arrays contain the following data:
y1 - the output frequency values using Butterworth filtration;
y2 - the output frequency values using Chebyshev I filtration;
y3 - the output frequency values using Chebyshev II filtration;
y4 - the output frequency values using Elliptic filtration.

The graphical presentation of the results is depicted on Fig. 10.

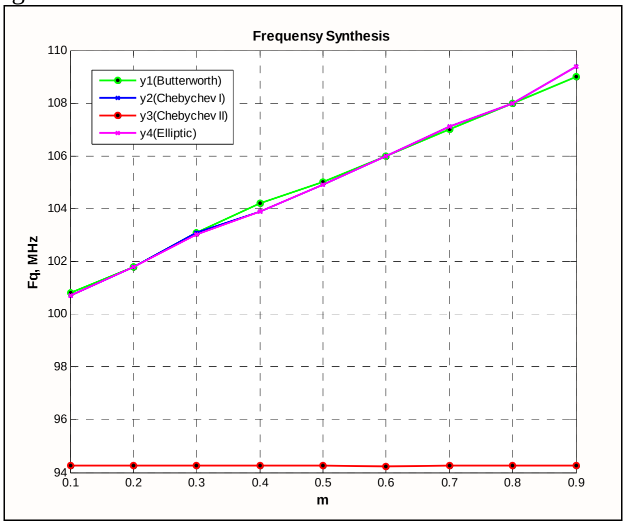


Fig. 10. Output frequency plots.

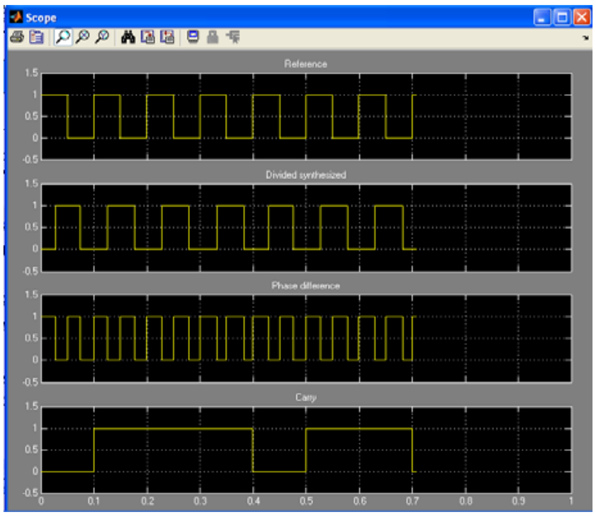


Fig. 11. Pulse signals oscillograms.

The explanation of the simulation results gives the students knowledge about the filter usage. Chebyshev I and Elliptic filtration methods have sharper slopes so the results are the most close to the real values. The inversed Chebyshev approximation is not suitable for the proposed frequency synthesis.

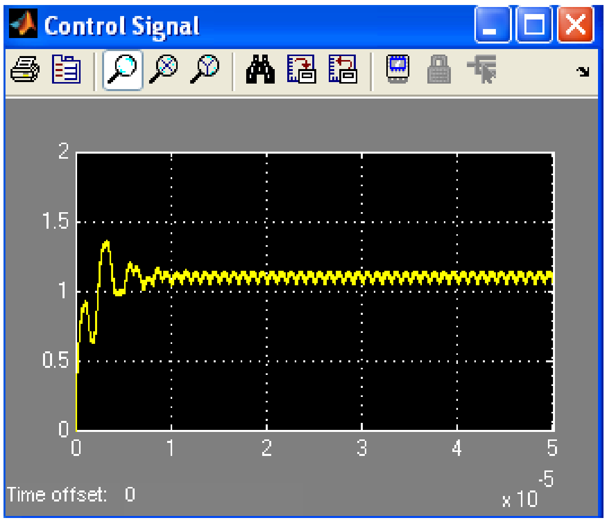


Fig. 12. Control signal oscillogram.

The signals in the important nodes are visualized on Fig. 11 and Fig. 12 using the virtual instrument Scope.

IV. CONCLUSION

Using the proposed approach for PLL fundamentals teaching help the students to understand better the PLL operation and the different blocks impact over the entire performance. They improve their skills for work with MATLAB/Simulink and learn how to analyze the system behavior using simulation investigations. Also they obtain abilities to present the results.

Nowadays Simulink is widely used in the engineering education in many fields so it can be successfully applied in the Electronic Circuits Design teaching.

REFERENCES

[1] http://www.mathworks.com/help/comm/examples/pll-based-frequency-synthesis.html
[2] http://www.mathworks.com/help/comm/examples/fractional-n-frequency-synthesis.html
[3] mathworks.com/newsletters.
[4] http://www.eas.uccs.edu/wickert/ece5675/
[5] Egan, William F. "Fractional-N and Relatives", *Frequency Synthesis by Phase Lock*, (2nd ed., pp. 371-390). N.Y., John Wiley & Sons, 2000.
[6] Jyoti P. Patra and Umesh C. Pati. *Behavioural Modelling and Simulation of PLL Based Integer N Frequency Synthesizer using Simulink*, International Journal of Electronics and Communication Engineering. ISSN 0974-2166 Volume 5, Number 3 (2012), pp. 351-362