Phase Locked Loops

Generate a signal that has a fixed relation to the phase of a “reference” signal

Feedback System
Respond to both the frequency and the phase of the input signals

Analog or Digital Implementations

Common Element of many Communication Systems, Computer and Microprocessor Systems, …

Applications:
Stable Frequency Synthesizers, Stable Clock Generators, Synchronization,…

Available as Integrated Circuits for wide range of frequencies

FM: 88 to 108MHz, 0.1 MHz resolution
XTAL=10MHz R=100, 773≤NS≤973
To 10.7MHz IF frequency

Phase Locked Loops…Phase Locked Loops…

Performance Issues: Phase Noise, Spur, Lock Range, Lock Time

1) Phase(Frequency) Detector (PFD):
Voltage phase detectors: Mixer, XOR Current charge-pump (wider lock range)

2) Loop Divider
CMOS N Counter < 200MHz
Single, Dual, Quadruple Modulus Prescaler ~ 3GHz

Example P/(P+1)=32/33 Prescaler (P+1 is P scaler + pulse swallow circuit)

N = a(P+1) + (b-a)P = bP+a

Phase Locked Loops…Phase Locked Loops…

Double Modulus Prescalers

\[ a = N \mod P \quad b = N \div P \rightarrow b \geq a \text{ to work properly} \]

\[ b < a \text{ : invalid } N \rightarrow \text{ minimum valid continuous set of } N \text{ starts at } P(P-1) \]

Minimum Continuous Divide Ratio

\[ \rightarrow \text{ There are some lower ones but not continuous} \]

Quadruple Modulus Prescaler… 2 pulse swallowers 1 and 4 pulses

Phase Locked Loops…Phase Locked Loops…

Comparison… achieving lower minimum continuous divide ratio

\[ \text{N has big effect on phase noise. The lower } N \text{ the better} \]

ACT3 Justify this
### Phase Locked Loops

**Fractional N**

- Less Phase Noise
- \( N = N_{ini} + \frac{FN}{FD} \)

Example: \( N = 100.25 \) ~ 1MHz Comparison Frequency, spur at 250KHz
- 1st order: 100 x 3 + 101
- 2nd order: 98, 99, 100, 101
- 3rd order: 96..103

#### Loop Filter
- First or Second Order
- Usually 2nd
- LC filters, SAW (to 3GHz) / BAW (to 16GHz) filters in higher frequencies

#### VCO
- Integrator, Silicon, LC, SAW oscillator, …
- Varicap diode + LC tank for high frequencies

### Phase Locked Loops

**Abstraction:**

1. **Physical Model** → **Analytical Model** → **Simulation Model**

2. **Two Modes:**
   - Acquisition (Large phase error)
   - Tracking (Smaller phase errors)

3. **Phase Detector:**
   - Non-linear
   - Linear

4. **VCO Output:** PLL input and VCO output must have quadrature phase → odd function of phase error
Phase Locked Loops...

Example: \( f_a = 5, \quad G = 30, 40 \)

Second Order PLL, \( F(s) = \frac{s + a}{s + \lambda a} \), \( \lambda \ll 1 \) (Perfect \( \lambda = 0 \))

\[
H(s) = \frac{G(s + a)}{s^2 + (G + \lambda a)s + Ga}
\]

\[
s^2 + (G + \lambda a) + Ga = s^2 + 2\zeta\omega_n s + \omega_n^2
\]

Now we have two degrees of freedom, \( G \) and \( a \)

In PLL Design Process:

Natural frequency and damping factor are specified and \( G \) and \( a \) are determined

Phase Locked Loops... Integrator

Impulse Invariant Transform

\[
H_a(s) = \sum_{k=0}^{\infty} \frac{A_k}{s + s_k} \rightarrow H(z) = \sum_{k=0}^{\infty} \frac{A_k}{1 - e^{-\tau_k} z^{-1}}
\]

Bilinear Example

Integrator, Trapezoidal Approximation

\[
y[n] = y[n-1] + T (x[n] + x[n+1]) / 2
\]

\[
H_a(s) = \frac{T}{s} \rightarrow H(z) = \frac{T}{2} \frac{1 + z^{-1}}{1 - \frac{1}{2} z^{-1}}
\]

Phase Locked Loops... Integrator

Phase Locked Loops... Integrator, Rectangular Approximation

\[
y[n] = y[n-1] + T x[n]
\]

\[
H_a(s) = \frac{A_k}{s + s_k} \rightarrow H(z) = \sum_{k=0}^{\infty} \frac{A_k}{1 - e^{-\tau_k} z^{-1}}
\]

Phase Locked Loops... Integrator

Phase Locked Loops... Integrator, Trapezoidal Approximation

\[
y[n] = y[n-1] + T (x[n] + x[n+1]) / 2
\]

\[
H_a(s) = \frac{T}{s} \rightarrow H(z) = \frac{T}{2} \frac{1 + z^{-1}}{1 - \frac{1}{2} z^{-1}}
\]

Case Study: PLL Simulation

Divide the Simulation into separate parts:

- Trivial one can be Preprocessor: Get the system parameters and make the data and elements parameters intrinsic parameters like sampling rate, run length and …
- Engine: Reads the data and executes the tasks
- Postprocessor: Analyze the results, make diagrams and …

Make the elements representable in simulation engine environment:

Everything is almost clear, for instance for \( F(s) \):

\[
F(s) = 1 + \frac{(1 - \lambda) a}{s + \lambda a}
\]

We use Trapezoidal Integration

Sampling Frequency \( \sim 100 f_n \), loop filter BW

Delay needed

Then make the Signal Flow diagram

for \( i = 1 : npts \)

\[
s1 = \text{phin}(i) - \text{phivco};
\]

\[
s2 = \text{sin}(s1);
\]

\[
s3 = G*s2;
\]

\[
s4 = 1*s3;
\]

\[
s4a = 4*s4+2*s5;
\]

\[
w1b = s4+w2b;
\]

\[
w2b = s4+w1b;
\]

\[
s6 = 3+s5;
\]

\[
w1c = s6+w2c;
\]

\[
w2c = s6+w1c;
\]

\[
\text{phivco} = \text{w1c}/\text{w2ofs} ;
\]

\[
\text{phierror}(i) = s1;
\]

\[
\text{fvco}(i) = s6/\text{twopi} ;
\]

end
Phase Locked Loops... Simulation Error Sources

Physical to Analytical
Idealizations: e.g. ignoring the voltage range limitation on different points of the flow graph (related to power supply)

Analytical to Simulation Model
1) Aliasing, caused by moving from analog to digital
   more serious because of non-linearity

2) Feedback inherent discrete delay
   \[ e_d(t) = \sin(\varphi(t) - \theta(t)) \]
   \[ e_d(nT) = \sin(\varphi(nT) - \theta(nT)) \]
   \[ e_d(nT) = \sin(\varphi(nT) - \theta((n-1)T)) \]

One sample delay: not present in physical and analytical models
Causes error in phase correction
Also can reduce the phase margin and cause instability

Solving Differential Equations in Simulation
Any type non-linear, time varying, …

Analog Computers of old times, built based on op-amps
- Drift problem (frequent calibration needed)
- Limited band width
- Multiplication and division was hard to implement …

First Order PLL example
\[ \frac{dy(t)}{dt} - G \sin y(t) = x(t) \]

Constant Coefficient Second order
\[ \frac{d^2y(t)}{dt^2} + a \frac{dy(t)}{dt} + by(t) = x(t) \]
Write highest order derivative in terms of other elements …
**Phase Locked Loops...**

**Second Order PLL example**

$$\Theta(s) = \frac{G(s + a)}{s(s + \lambda a)} E_d(s)$$

$$\frac{d^2 \Theta}{dt^2} + \lambda a \frac{d \Theta}{dt} = \frac{G}{s} \frac{de_v(t)}{dt} + G \xi(t), \quad \xi(t) = \sin \psi(t)$$

$$\dot{\psi} = \lambda \frac{d \psi}{dt} + G \frac{d \psi}{dt} \cos(\psi(t)) + G \sin(\psi(t)) = \frac{d^2 \psi}{dt^2} + \lambda \frac{d \psi}{dt}$$

**Singular inputs can be passed thru the integrators**

For simplification → Equivalent Input

**Signal flow diagram**

**Trapezoidal Integration (or better ones)**

---

**ACT4 Analog Sim in Simulink**

---

**MEX files: MATLAB Executables**

To call custom C, C++, or FORTRAN routines directly from MATLAB

As if they are MATLAB commands

Dynamically linked subroutines

- The ability to call large existing code, avoid rewriting them as M-files
- Rewrite bottleneck computations (like for-loops) as MEX-files for efficiency

**MATLAB can Link to Many Compliers, including MS Visual Studio**

- mex -setup
- mex prog.cpp(/cl.f) generates prog.mexw32

In General:

- **N(F-K)+(N+1)K/F = N**

---

**ACTS Find out How it works**

---

**Phase Locked Loops...**

**General Transceiver Block Diagram**

**Phase Noise Multiplied by N (20logN dB)**

Cellular Transceiver N=30000 (900 MHz), 30 KHz Channel Spacing, 20 logN = 90 dB

Example: with F = 16, and K = 3, F=480 kHz

Log,F=4 bit counter

0, 3, 6, 9, 12, 15, (C2)
5, 8, 11, 14, (C1),
4, 7, 10, 13, (C0)

3 Carry-out, N+3/16
N=AP+(M-A)(P+1)=MP+M-A

In General

(N(F-K)+(N+1)K)/F = N+K/F=N

---

**MEX files: MATLAB Executables**

**General Form**
MEX files: MATLAB Executables

First approach, MATLAB environment / C

Write a function for \( \mathbf{y} = \text{mycmex}(\mathbf{x}, a) \):
\( a, b: \text{scalar}, \ x, y: \text{matrix}, \ function \ y=ax+b=a; \)

```c
#include "math.h" // any library headers
#include "mex.h" // required, contains all the MATLAB related stuff
void mexFunction(int nlhs, mxArray *plhs[], int nrhs, mxArray *prhs[])
{
    // Add the code here
    return;
}
```

**Input Data**

- \( a \) and \( b \) hold the mxArray objects containing \( x \) and \( a \) respectively.

**Output Data**

- \( y \) and \( b \) are the output of the function.

**Calling MATLAB Built in functions**

```c
#include "math.h" #include "mex.h" void mexFunction( int nlhs, mxArray *plhs[],  int nrhs, mxArray*prhs[] ) {
mxArray *result, *arguments[2];
    double *xValues, *yValues, *oValue;
    if (nrhs!=2)
        mexErrMsgTxt("Two inputs are required");
    xData = prhs[0];
    xValues = mxGetPr(xData);
    M = mxGetM(xData);
    N = mxGetN(xData);
    aData = prhs[1];
    a = (double)(mxGetScalar(aData));

    nlhs = 2;
    plhs[0] = mxCreateDoubleMatrix(M, N, mxREAL);
    yValues= mxGetPr(plhs[0]);
    for(i=0;i<N;i++)
    {
        for(j=0;j<M;j++)
        {
            yValues[(i*M)+j] = a*xValues[(i*M)+j];
        }
    }

    plhs[1] = mxCreateDoubleMatrix(1, 1, mxREAL);
    oValue = mxGetPr(plhs[1]);
    oValue[0] = a;
    return;
}
```

**EX4:**

```
5.2, 5.5, 5.7, 5.13, 5.14, 5.15
```

5.6, 5.9, 5.15