Solar Power Anti-Islanding and Control

Problem: When connecting a solar power source to the AC mains it is possible to supply power to the local area in the event of a power outage. While your neighbors might be happy with this behavior, it is a hazard to utility workers attempting to restore power. This affect, called islanding, must be eliminated in the Grid Tie Inverter design. If a phase-locked oscillator synchronizes the inverter output to the line; then a power outage results in the inverter synchronizing to its self. The frequency will then drift out of tolerance, signaling a power outage. This article describes how to achieve this design goal using a digital controller and Intusoft's DSP Designer to simulate the digital design and generate some of the necessary code.

Testing the anti-islanding feature: The U.S. National Electric Code, NEC, defines a test using a resonant circuit at the inverter input, which has a Q of 3 at the inverter maximum power level. The circuit is adjusted so that removal of the mains power will not be detected instantly. The inverter will not be overloaded so that in the absence of special detection circuitry, the solar power will electrify the local grid.

Block Diagram: Figure 1 is a block diagram of the line

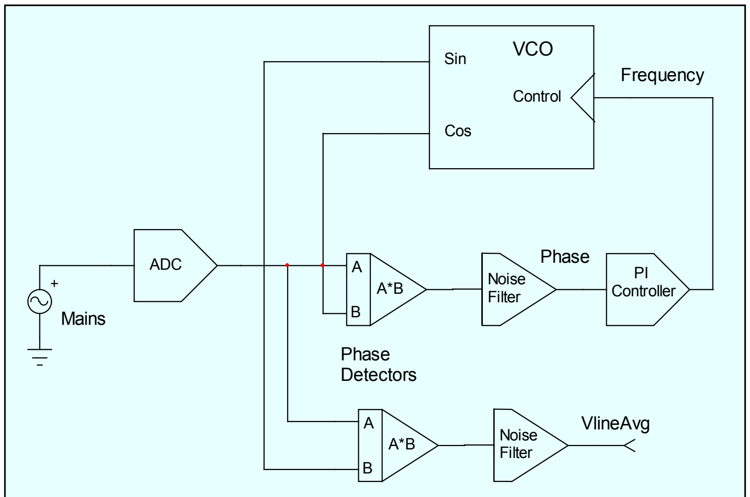

Figure 1, Block Diagram

synchronizing system. Notice that the operation of this system doesn't depend upon the NEC test condition. The idea is to synchronize the inverter using a phase-locked-loop, PLL [1]. If the grid is only supplied by the Inverter, then no error signal is developed at the phase detector and the frequency will drift toward zero. When it gets below some pre-defined limit, the Inverter is shut off until mains power is restored; at which time the PLL syncs up and solar power generation is resumed. The logic gets a bit more complex when a battery backup is included, requiring a switch to isolate the grid from the backup loads. The traditional PLL controller consists of a multiplier to detect the phase error, followed by a PI controller. The large second harmonic signal must be filtered to reduce non-linearity in the voltage-controlled oscillator, VCO, and to allow meaningful measurement of phase error, frequency and line voltage.

Theory: Phase Locked Loops belong to a class of nonlinear circuits that have been studied extensively [1]. When phase detection is accomplished using a multiplier, the second-order nonlinear equation can be solved explicitly. PLLs are used in communications circuits, such as Global Positioning Systems (GPS) to make a narrow band filter centered about the carrier frequency. The noise filter, while not part of the lower frequency phase control loop, is used to eliminate out of band noise. For this problem, the carrier signal is the 60 Hz mains and it is not particularly noisy, although it may have some harmonic distortion. The mains frequency is tightly controlled so that various generators on the grid can be easily synchronized. The main signal to be eliminated by the noise filter is the double frequency, 120 Hz, component that is output by the phase detector. The digital filter used here reduces this unwanted signal by nearly 40 dB. The average value of the mains fundamental is detected using the quadrature signal from the VCO, which is synchronized to the mains. The following logic is used to indicate that synchronization has begun; that is, it sets a software flip-flop:

```
    VlinAvg > 78 Volts (86 VRMS) and
    abs(frequency-60) < 1) and
     (abs)phase < .025 radians (1.4 degrees)
```

The flip-flop is reset when:

```
    abs(frequency-60) > 2) or
    VlinAvg < 72 Volts (79 VRMS)
```

The first reset condition accomplishes the anti-islanding function when the Inverter is supplying power to the grid, and the second condition signals a grid power loss when in standby. The sine output of the VCO is used to switch synchronous rectifiers used to transfer power between the grid and the backup system when a backup battery is present. Backup systems require transferring power in either direction to account for load phase shift (motors) and battery charging, so the use of synchronously switched rectifiers is required.

Selecting Sample Rates: High sample rates give improved accuracy but require longer digital word lengths. Constraining word lengths to 16 bits allows the use of very low cost and low power digital controllers. For this problem, the sine-cosine generator is run at the Inverter switching frequency, which is 25 kHz, a 40 microseconds. That resolves the line voltage to about 2.5 Volts at zero crossing for a 120 Volt RMS line. But the PLL can be sampled at a much lower rate, greater than or equal to 10 times its loop bandwidth. The communication system for this DSP runs at 1 kHz, which meets this criteria.

DSP Equations: The DSP equations are formed using a matrix algebra approach. A SPICE simulator is used to build up the matrix from the simulation schematics net list. Then the matrix is ordered from top to bottom, first using unwanted states, then using the states that need to be computed, and finally using states that are available as inputs. For a linear system, the right-hand side of the matrix is constant. When this matrix is arranged in a triangular form, it can be solved repeatedly at each sample time by using backward substitution.

http://www.intusoft.com/dsp/matrix_solution.pdf

Each row in the backward substitution is formed using a multiply-accumulate series. Scaling is accomplished by scaling the matrix constants up by 2^radix. When the multiply accumulate is finished, the result is multiplied by 2^-radix and saved. These saved results are the desired state solutions and their values are used in subsequent state computations. All of this can be automated so the only thing that must be supplied is the radix parameter. The magnitude of the resulting coefficients lie between 1 and 2^15. If larger than 2^15, the multiply can't be done using 16-bit integers. As the coefficients approach 1, the pole-zero placement accuracy is compromised.

Scaling: Each linear block in the system must be scaled; that is, the radix and sampling frequency need to be selected. Sampling frequencies come from a limited set of real time interrupts. For this problem, they are either 40 kHz or 1 kHz. Here are the blocks to be scaled:

1. ADC, this block is sampled at 40 kHz because that is the Inverter switching rate. It is used both here and for the Inverter control system.
2. PLL noise filter sample rate uses the 1 kHz sample rate. This is because it is the lowest available interrupt rate that is greater than 10 times the PLL bandwidth.
3. Noise filter sampling also uses the 1 kHz interrupt and sets radix=10
4. The VCO is sampled at 40 kHz because its sine output is used as the Inverter set point. Internal scaling uses radix=13.
5. The multiplier is a non-linear block that is computed at the PLL sample rate. Since it is non-linear, its solution does not use the matrix solution technique and is:
   Vmul= ((Vline)*(long)zic)/1571
   =(((Vline*(long)zic)>>10)*5340)>>13

The A to D converter outputs an unsigned 12-bit value. For DSPs using 10 bits, it is shifted left by 2 places before being used. Subtracting 2048 makes it a signed number (Vline), accounting for the Vref/2 offset. Casting zic to a long forces the first product to be a long so that the result doesn't underflow. The result is automatically changed back to a 16-bit integer by the C compiler because that's the data type of Vmul. The 1571 value scales the peak signal used in the noise filter and accounts for the ADC input signal conditioning gain. The equation is re-arranged to eliminate the division as show in 5 above(1/1530 = 5340*2^-23). The noise filter gain is set to prevent saturation within the filter by using simulation results. Similarly, the gain of the shaping network is set to prevent saturation of the internal frequency state and fine-tunes the over-all loop response. Again, the adjustment relies on simulation results. The initial condition of the frequency state is set so the shaping network output, Vfreq, is 377 rad/sec (2*pi*60 Hz). With that scaling, the gain of the VCO integrators is Vfreq*Tsc, where Tsc is the sine-cosine sampling period.

There is no single "right"" answer to the scaling problem. The trick is to avoid the wrong answers. Aggressive scaling will adjust maximum state values close to saturation limits. At the same time, the quantizing noise will be reduced because the dynamic range is maximized. If saturation occurs, the control system can produce unexpected results. That's programmers' language for smoke! It might seem that using saturation limits in intermediate calculations is a good idea. However, it is permissible to have intermediate overflows as long as the resulting state variable doesn't overflow. See:

http://www.intusoft.com/dsp/dsp.pdf pg 16

Phase Detector and Filter: The phase detector uses a multiplier in the classical PLL approach. The filter is a 2nd order elliptic filter with 1dB ripple and 20 dB attenuation. The s-plane roots can be calculated using:

www.intusoft.com/filter.htm.

The cutoff frequency is adjusted so the filter notch is at 120 Hz in order to minimize the 2nd harmonic noise. Figure 2 shows the digital realization converting the s-plane roots to direct programming coefficients using the parameter block shown in Table 1. An extra pole was added to further reduce the noise without impacting the PI controller performance. The circuit was turned into a SPICE subcircuit and added to a user library. Code was generated for the DSP version of the circuit [2].

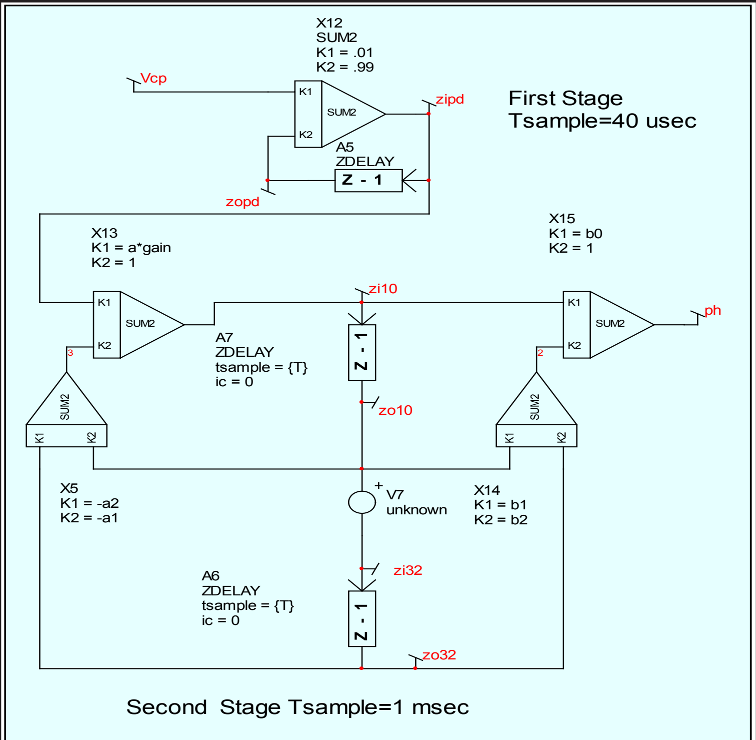

Figure 2, Noise Filter

Table 1, Parameters

Parameters
gain=.1
Radix=13
onee=2^Radix; sets quantizing levels
T=1m
Tsc=40u
Fc=37.1; adjust for minimum at 120 Hz
DR=-.5148;pole real part
DI=.9424;pole imaginary part
NR=0; zero real part
NI=3.205; zero imaginary part
BW=2*3.14159*Fc

```
BW2=BW*BW
D2=1
D1 = -2*DR
D0=DR*DR+DI*DI
N2=1
N1=2*NR
N0=NI*NI+NR*NR
* Quantized Coefficients below
a=floor(onee*D0/N0+.5)/onee
a0=D0*BW2 + D1/T*BW+D2/T/T
a1=-floor(onee*(D1/T*BW+2*D2/T/T)/a0+.5)/onee
a2=floor(onee*D2/T/T/a0+.5)/onee
b0=floor(onee*(N0*BW2 + N1/T*BW+N2/T/T)/a0+.5)/onee
b1=-floor(onee*(N1/T*BW+2*N2/T/T)/a0+.5)/onee
b2=floor(onee*N2/T/T/a0+.5)/onee
```

The PI Controller: A simple PI controller shown in Figure 3 sets up the control loop. Notice that the internal integrator has a signal that is proportional to frequency, and it will be used later on in the anti-islanding logic.

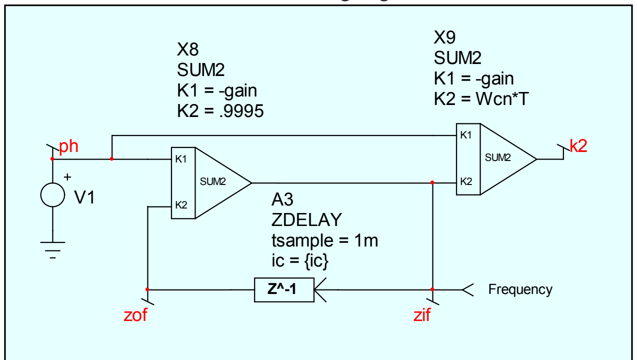

Figure 3, PI Controller with Frequency Test Point

The VCO: The VCO is implemented using an algorithmic sine-cosine oscillator. Two integrators are cascaded with the gain controlled using a non-linear block in series with each integrator. One of the integrators uses the delayed output for 2 reasons:

1. The total phase lag around the loop must exceed 180 degrees to make an oscillator.
2. The equations need to be broken up to avoid solving simultaneous equations, so the gain constants can be used directly.

An automatic gain control, AGC, limiter is used to keep the oscillation at a constant level. A simple peak detector is used, it's contained within the Zdelay model. Figure 4 shows the SPICE model and Table 21 shows the C-language code.

Table 2, VCO source code

```
void vco(void)
{
        si= (long)((((long)freq*zoc)>>13)*2684) >> 13 ;
        zis=si+zos;
        ci = (long)((((long)freq*zis)>>13)*2684) >> 13;
        zic=zoc-ci;
        zoc=zic;
        zos=zis;
        if(zos > 8192) {
                zos = 8192;
                zoc=0;
        }
}
```

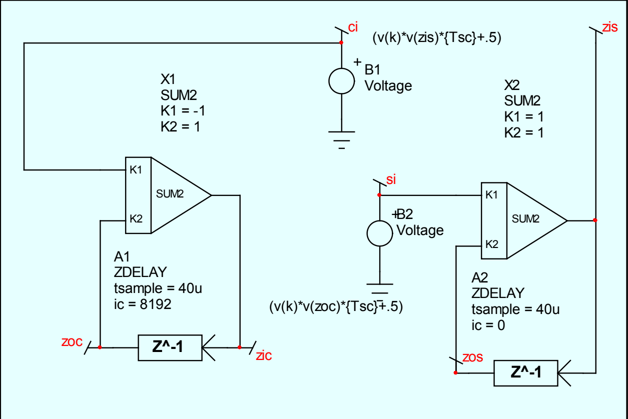

Figure 4, VCO

Notice that B1 and B2 equations were scaled to work with integers.

Simulation Results: Figure 5 shows the startup transient when the VCO phase is 180 degrees out of phase with the line. At 500msec a line failure is simulated by connecting the VCO output to the line input. Notice the frequency drifting out of tolerance in less than .5 second because the integration constant, Figure 3 X8:K2, was set slightly less than 1, forcing the output to decay toward 0 with no error signal.

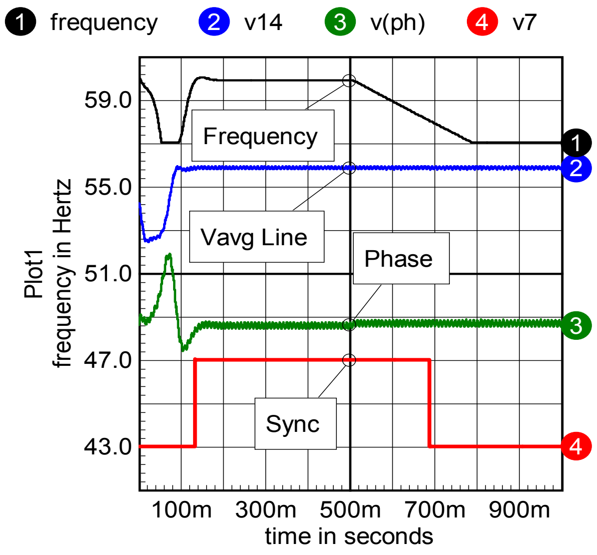

Figure 5, Simulation Results

Test Results: Real time programming is fraught with unexpected behavior. That's why; even with the best tools it takes orders of magnitude more time to program a "line" of code. For this problem, several stumbling blocks needed to be resolved.

1. The 1msec sampled difference equations are interrupted by the 40 µsec interrupt service request (ISR). Both the 40 µsec and 1 msec ISR use the MAC accumulators. (These implement special DSP functions for multiply-accumulate.) Therefore, the 40 µsec ISRs needs to push

these accumulators onto the stack. When this is omitted, the resultant VCO signal is erratic.

2. The 60 Hz sync signal test was originally:

```
if((zis > 0) && (zos < 0)){
LATBbits.LATB8 = 0;
}
else if((zis < 0) && (zos > 0)){
LATBbits.LATB8 = 1;
}
```

and it was changed to:

```
if((zis > 0) && (zos <= 0)){
LATBbits.LATB8 = 0;
}
else if((zis < 0) && (zos >= 0)){
LATBbits.LATB8 = 1;
}
```

because the occasional time when zos was equal to zero caused the sync signal to fail.

3. The 3rd order input filter was originally setup to first do the elliptic filter section followed by the first order pole. Both filters were to be done in the 1ms ISR. However, 1 msec is too granular for synchronization. And the elliptic filter couldn't be solved in the 40usec ISR. The filter was revised to place the pole before the elliptic filter and was solved in the 40 usec ISR. That combination satisfied the accuracy and scaling requirements.
4. The PLL gain is proportional to the magnitude of the input signal. To eliminate gain changes with power line voltage, the line signal was passed through limiting code. The limiter includes a 25-sample lockout to "de-bounce" the output.

All of these seemingly minor problems required 2 days to resolve (6 days real time!) and produced only 40 lines of C code (plus about 100 lines DSP Designer code).

A test signal was generated using the 40 usec ISR and was input to the PLL at 60.0962 Hz. Test code was used to measure peak and RMS noise, using the deviation of the zos (sine signal) when the sync signal switched. Results for 1000 samples were:

Peak deviation: 3.9%
RMS deviation: .75%

The test code remains in the source code[2] and can be activate by including the

```
#define TESTERROR
```

line.

Summary: Converting analog technology into a digital implementation is made easier using Z-Transform models with SPICE simulation. Moreover, automatic code generation using a matrix-based solution relieves the designer of the excruciating job of scaling each multiply-accumulate operation. It uniquely eliminates unused states by ordering the unneeded states at the top of the matrix so that backward substitution ends with the last required state…it simply doesn't solve for unused states.

These features are unique to ICAP/4 DSP Designer; to recap:

1. A new more efficient Z-Transform model that guarantees convergence
2. Decomposition of the SPICE matrix for DSP code generation
3. Generic C-Code generation
4. Assembly code for selected DSPs

References:

[1] Phaselock Techniques, Floyd M. Gardner

[2] Downloads of the detailed design are available at www.intusoft.com/DigitalPLL.zip.