Si5351 VFO

Introduction:

A central thread in my recent projects (last 40 years) has been the job of putting a radio “on frequency”. Most receivers and transmitters (Superhet, DC & some SDR’s) require a VFO. Traditionally, this has been a free running LC oscillator, with its inherent “problems” - drift, FM, noise, frequency calibration.

More recently, PLLs and DDS have been used to solve the “frequency problems”, but introduce others eg. spurious sidebands &/or a host of low level birdies.

Until recently (for some definitions of recent ;-), the “digital” approach has involved a lot of components - dividers, phase comparators loop filters & maybe a microprocessor.

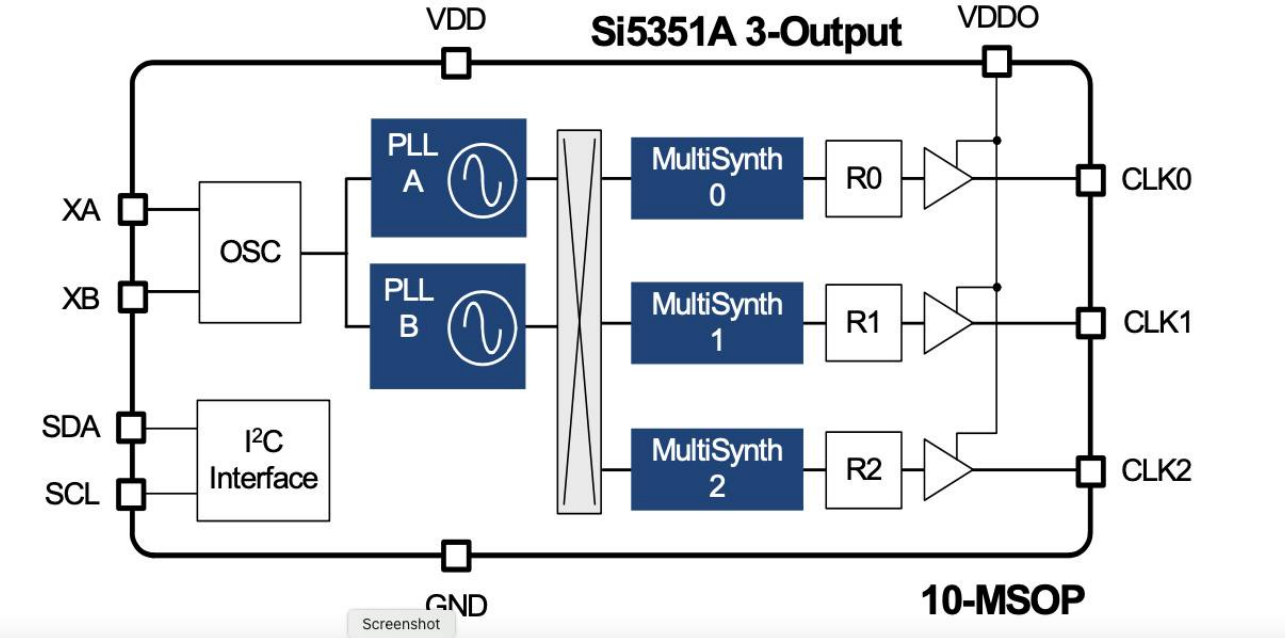
Now a “synthesised VFO” can be built with two cheap chips. A Si5351 programmable clock chip & a microprocessor.

Introducing the Si5351:

This chip is a programmable clock generator, intended for digital applications. It offers modest performance in the way of timing jitter (which leads to spurious birdies & FM), but turns out to be entirely adequate for use as a VFO in HF SSB transceivers. Better jitter performance is available from the venerable Si570 & spectacular performance from the “top of the line” Si549, but these cost a lot more.

All these devices use a combination of a PLL controlled VHF/UHF oscillator and “fractional N” dividers to produce the final output frequency.

So, let’s have a closer look at the Si5351.

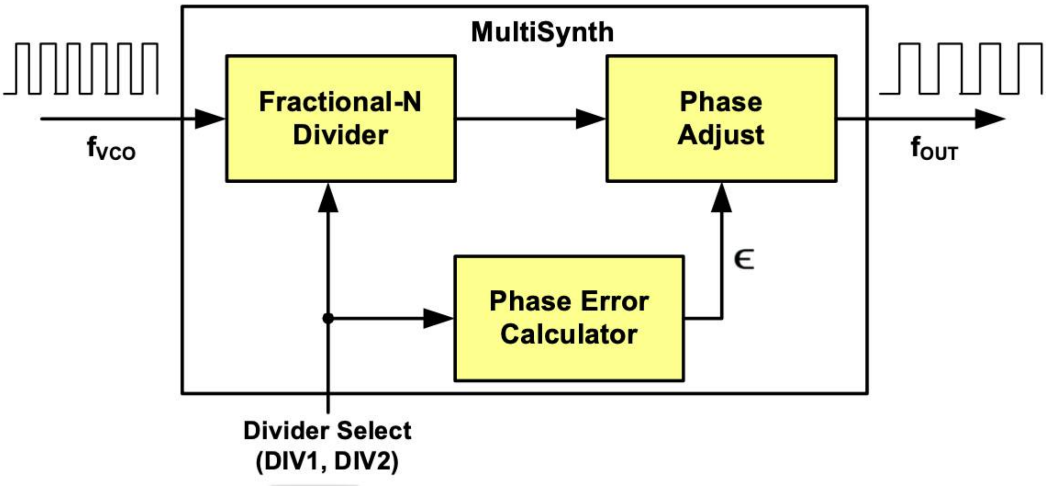


Features (that are useful in VFO service);

- Up to 3 outputs covering 2.5KHz to over 100 MHz. Not all can be flexibly chosen.
- Crystal or TCXO controlled for stability
- Programmed via I2C
- Can do quadrature outputs over ~3.5MHz - useful for SDR
- Glitchless frequency changes

The two PLLs both use VCOs running at 600 to 900 MHz. The VCO frequency appears to be locked to the crystal frequency using a fractional N divider (but the data sheet is a bit vague on this ;-)

The VCO outputs can be routed to the “MultiSynths” (which contain fractional N dividers) then to output dividers & drivers



This is described in the data sheet for the Si5334 (a related device) as:

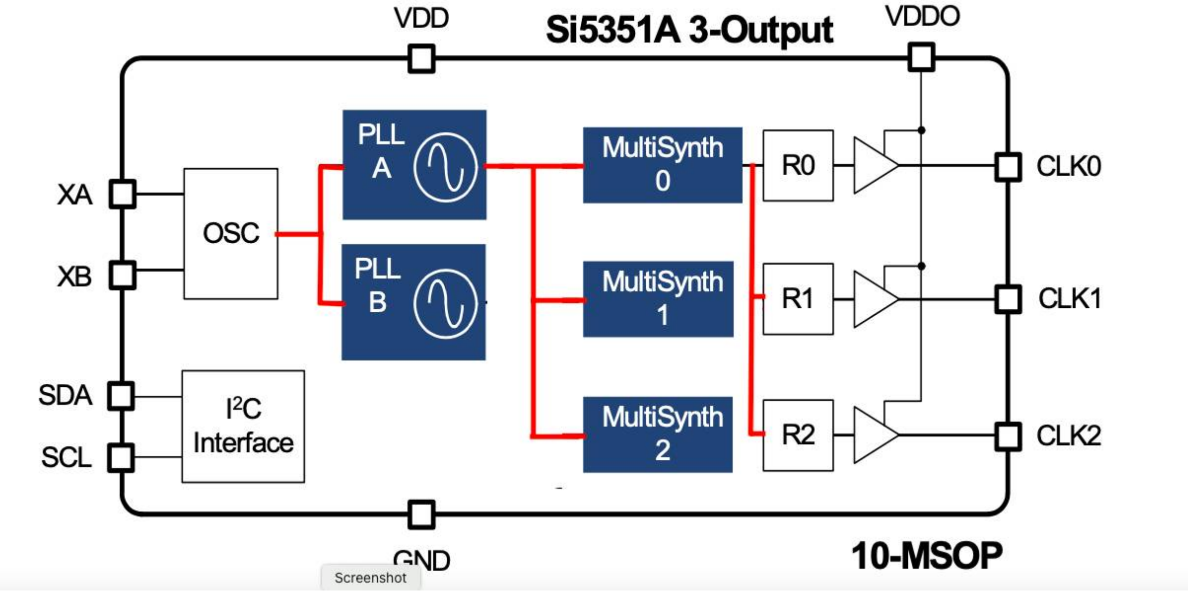
“Based on a fractional-N PLL, the heart of the architecture is a low phase noise, high frequency VCO. The VCO supplies a high frequency output clock to the MultiSynth block on each of the four independent output paths. Each MultiSynth operates as a high speed fractional divider with Skyworks Solutions' proprietary phase error correction to divide down the VCO clock to the required output frequency with very low jitter.

The first stage of the MultiSynth architecture is a fractional-N divider which switches seamlessly between the two closest integer divider values to produce the exact output clock frequency with 0 ppm error. To eliminate phase error generated by this process, MultiSynth calculates the relative phase difference between the clock produced by the fractional-N divider and the desired output clock and dynamically adjusts the phase to match the ideal clock waveform. This novel approach makes it possible to generate any output clock frequency without sacrificing jitter performance.”

For a more extensive, but less informative description, have a look at US patent US7295077, which can be found by a Google search. It is full of obfusticated detail & legal jargon, but it claims to describe the MultiSynth operation. I gave up reading halfway down the first page.

How I used the “beast”:

- Both PLLs were programmed identically
- PLLA was used to drive all MultiSynths
- The PLL A fractional N divider was used as the “frequency control”
- MultiSynth 0 was used to drive all outputs
- All MultiSynths were set to the same even integer division for minimum jitter/noise: A single value usually covers a whole Amateur band. It does not have to be changed for every change of VFO frequency.
- All output dividers R were set to 1



PLLA (& B) Programming:

The PLL frequency is equal to the crystal frequency multiplied by the first stage fractional N divider as described in part 3 of AN619

$$Fvco = Fxtal \times \left(a + \frac{b}{c}\right) \qquad (1)$$

Which can be rearranged to:

$$\frac{Fvco}{Fxtal} = \left(a + \frac{b}{c}\right) \qquad (2)$$

Then both sides can be multiplied by 2^N

$$\frac{Fvco}{Fxtal} \times 2^N = \left(a + \frac{b}{c}\right) \times 2^N \qquad (3)$$

and rearranged again to give:

$$Fvco \times \left(\frac{2^N}{Fxtal}\right) = \left(a + \frac{b}{c}\right) \times 2^N \qquad (4)$$

By careful choice of N, $\frac{2^N}{Fxtal}$ can be made to have a 32 bit value where the most significant bit is 1. This becomes the software “Frequency Scaling Factor” (which can be adjusted to compensate for errors in Fxtal.)

The VFO frequency is given by $Rx + BFO$ (or $Tx + BFO$ for transmit)
Note: use high side mixing to minimise harmonic problems.

$$(Rx + BFO) \times OPdiv = Fvco \qquad (5)$$

Combining (4) and (5) gives:

$$(Rx + BFO) \times OPdiv \times FSF = \left(a + \frac{b}{c}\right) \times 2^N \qquad (6)$$

“c” is chosen to be a “nice round binary number”. 2^{19} is a good choice.

With “N” also carefully chosen (I think it was around 56), the left hand side of this equation (6) produces a binary number that can be sliced & diced to give the required programming information ie, the numbers “a” and “b”. These are then used as per AN619 part 3.2.

MultiSynth Programming:

These are programmed to an even integer value for minimum birdies. To see how noisy fractional N division is, have a look at the spectra on the page 6.

See AN619 page 6

For the all MultiSynths:

$$OPdiv = MSdivision \times R = \frac{Fvco}{Fvfo} \qquad (7)$$

$$MSdivision = a + \frac{b}{c} \qquad (8)$$

with a = desired integer division ratio, b = 0 and C = 1

$$MS0P1[17:0] = 128 \times a + floor\left(\frac{128 \times b}{c}\right) - 512 \qquad (9)$$

With the chosen values of a, b & c, this results in:

$$MS0P1[17:0] = 128 \times a - 512$$ (10) -> registers 44, 45 & 46

$$MS0P2[19:0] = 128 \times b - c \times floor\left(\frac{128 \times b}{c}\right) \qquad (11)$$

With the chosen values of a, b & c, this results in:

$$MS0P2[19:0] = 0$$ (12) -> registers 49, 48 & part of 47

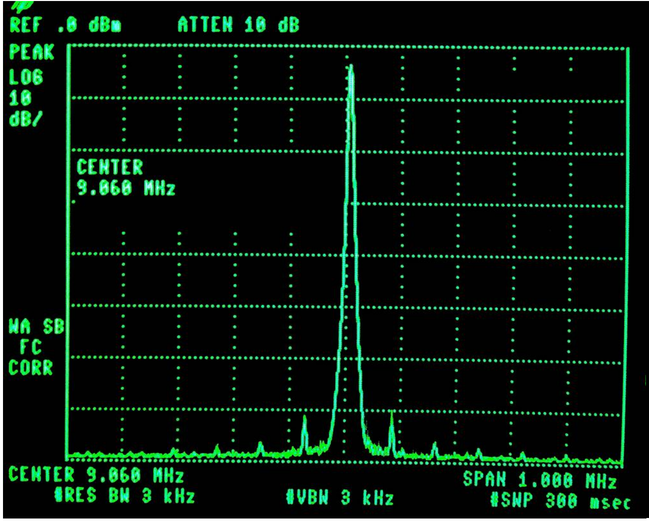
$$MS0P3[19:0] = 1$$ (13) -> registers 42, 43 & part of 47

Summary of design steps:

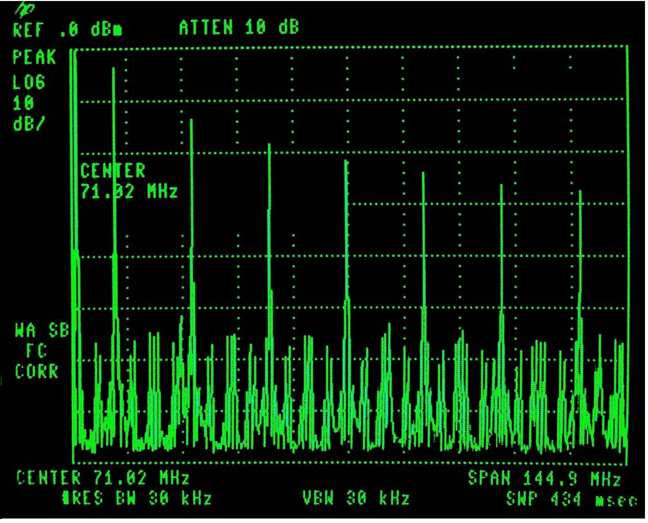
- Decide on VFO frequency (Preferably, Signal Frequency + IF frequency)
- Choose an even integer for OPdiv so that Fvco is between 600 and 900 MHz
- Use R=1 for the output divider
- Find a crystal in the range 25 to 28 MHz.
  Check that it does not fall on the image frequency $Fif + 2 \times Fsig$
- Calculate FSF
- Use equation (6) to get “a” and “b”.
  (“a” is the integer part of the Fractional N divisor, “b” is the fractional part.)
- HINT: If you decide to use a PIC processor, then ready made programs are available to do ALL the hard work. You just have to decide on frequencies (Crystal, Band, IF/BFO)

Output Spectra:

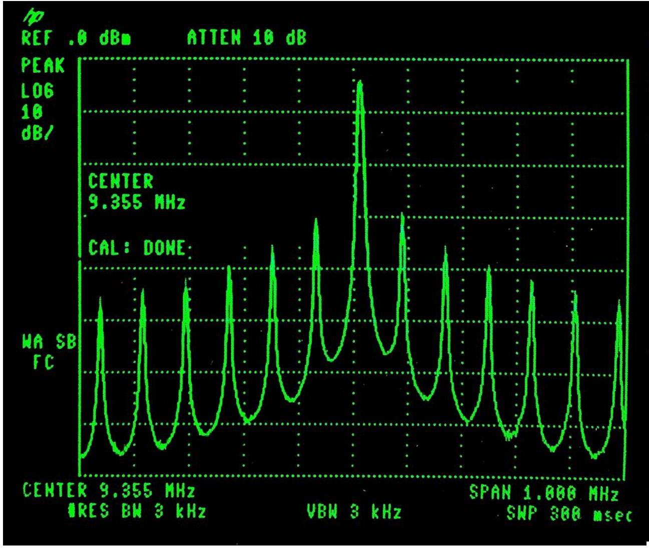
MS0 Integer division (narrow band)



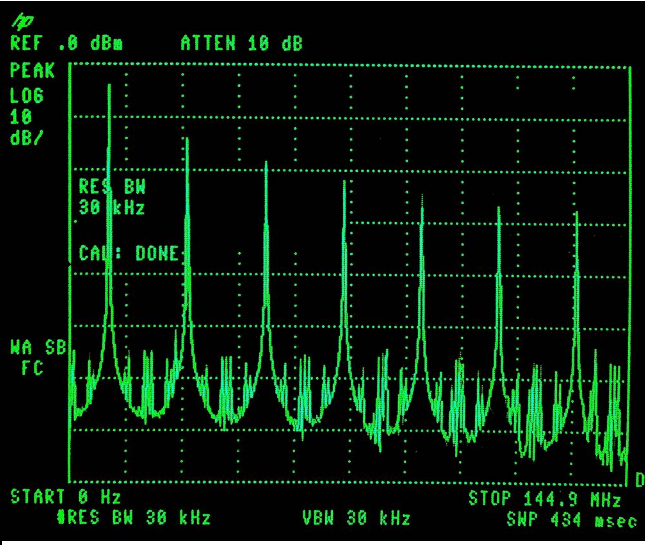
MS0 Integer division (wide band)



MS0 Fractional N division (narrow band)



MS0 Fractional N division (wide band)


REF .0 dBm ATTEN 10 dB
PEAK
LOG
dB/
RES BW
30 kHz
CAL: DONE
WA SB
FC
START 0 Hz
#RES BW 30 kHz
VBW 30 kHz
STOP 144.9 MHz
SWP 434 msec