EXPLOITING COST EFFECTIVE DIGITAL FPGA TECHNOLOGY TO ACCURATELY SYNTHESIZE S-BAND WAVEFORMS

Jonathan Riley C.Eng MIET

Slipstream Engineering Design Ltd

Slipstream Design, 11 Mercury Quays, Ashley Lane, Shipley, BD17 7DB

jriley@slipstream-design.co.uk

Abstract - *Generating frequency sweeps in the S-Band can be achieved in a number of ways. However, achieving the performance requirements of modern communication and radar systems with tight tolerances on frequency, timing and amplitude across temperature and batch variation is seldom easy.*

Add to this the desire from manufacturing to remove the need for individual module tuning and alignment makes these requirements harder still. Keeping the flexibility to change how the sweep is generated for example in a radar system (none / linear / hyperbolic / arbitrary) is a technical challenge and can often lead to complex and expensive solutions. The aim of this paper is to show that all these objectives can be met using cost effective commercially available FPGAs and DACs using the strengths of today's technology and exploiting Nyquist point of normal DAC operation. The implementation of these concepts are illustrated with a practical example of how this is achieved. This real life example provides an insight into the factors to consider when designing a system to directly digitally synthesize an S-Band output.

Introduction

This paper will discuss the generation of a single selectable S-Band (2-4GHz) frequency with a design goal of using low-cost commercially available technology.

The method of generating the S-Band signal in the digital domain is referred to as Direct Digital Synthesis, (DDS). This approach does not necessarily require a FPGA and DAC, single-chip solutions provide a very cost effective means of synthesising lower frequency signals and linear chirps. However such parts may be limited both in terms of the scope to adjust the modulation (e.g. chirps that are non-linear) and also in the output level. Combining the modus operandi of this class of components with a good understanding of FPGA and DAC technology it is possible to provide an enhanced solution to the problem.

A Background to Direct Digital Synthesis

There are three main aspects to the versatile DDS system, the FPGA which is used to generate the signal and modulation, the output DAC, followed by a low pass filter. This is shown below in Figure 1 System Topology.

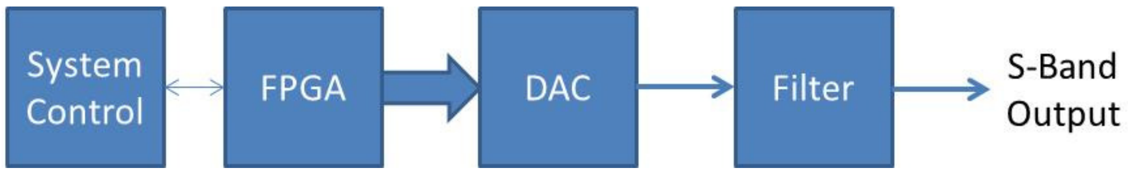

Figure 1 System Topology

There are two ways to create a sine wave digitally before passing the value to the output DAC to convert into the analogue domain 'Sequential Sine Generation' and 'Phase Accumulator Generation'.

Sequential Sine Generation

This method populates a block of memory with the pre-computed values that are sent to the DAC in a sequential order. This sequence produces the required output signal, the memory contents is passed to the DAC in the order in which they appear to keep the process straight forwards. This is illustrated in Figure 2 Sequential Sine Generator.

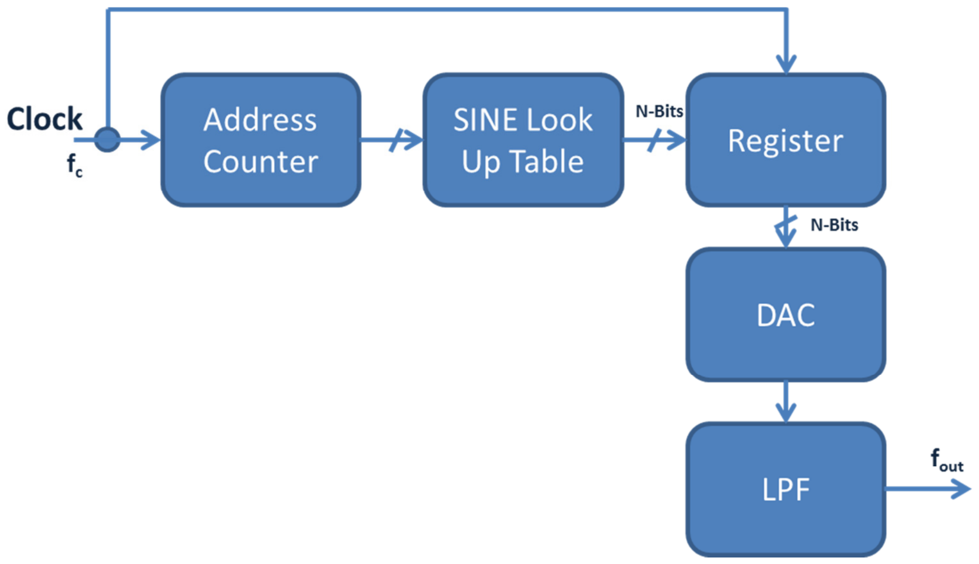

Figure 2 Sequential Sine Generator

Phase Accumulator Generation

The Phase Accumulator Generation, which will be discussed though this paper, has memory that holds one complete cycle of a sine wave. In this case the method of generating a frequency is to add the phase angle required to the present angle and use this to index the sine table. This is shown in Figure 3 Phase Angle Incrementing Sine Generator.

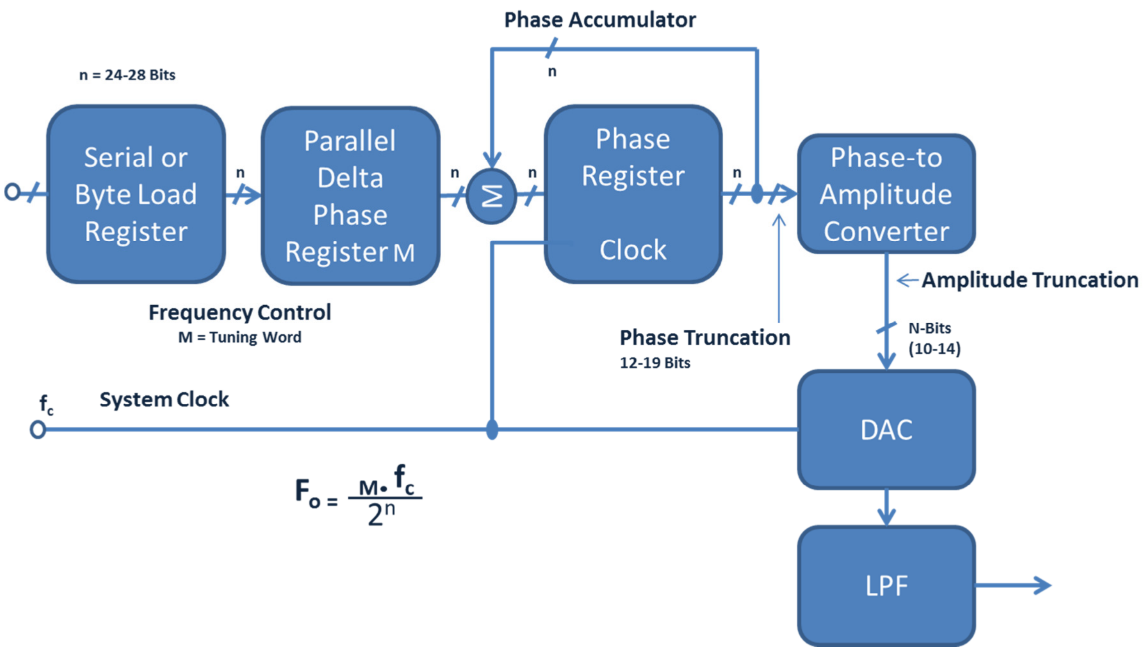

Figure 3 Phase Angle Incrementing Sine Generator

The two Sine generators methods are almost identical in terms of the hardware required. For the Sequential Sine Generator, the “Address Counter” is seen to be equivalent to the “Phase accumulator” of the Phase Angle Incrementing Sine Generator with the exception that in the former, the increment value is always 1, whereas in the latter the value is the phase increment (M).

Comparing the two methods, the benefit of the Phase Angle Incrementing Sine Generator is seen when changing frequency. The Sequential Sine Generator requires either a change of the clock frequency f_C or the data in the Sine Lookup Table; both of which are not easily done while the system runs. The Phase Angle Incrementing Sine Generator simply requires the new value of the phase increment (M). A diagram of the phase incrementing method is shown in Figure 4 Phase incrementing method.

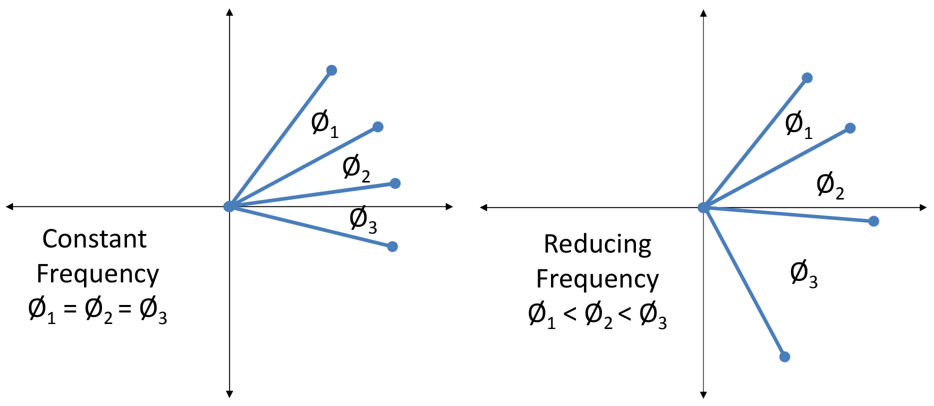

Figure 4 Phase incrementing method

For S-Band frequencies, taking 2.900 GHz to 3.100 GHz as the example band section of interest, the upper frequency determines the DAC clock speed. If the Nyquist sampling theorem is to be satisfied, the minimum clock frequency for the DAC will need to be at least 6.5 GHZ without use of a significant 'brick wall' filter. Using a DAC with 8-bit resolution, the data transfer rate that the DAC requires is therefore the product of these two numbers or 52 Gbps. FPGAs and DACs to meet these requirements are extremely expensive. However, the Nyquist sampling frequency rule may be broken if suitable care is given to the selection of operating frequency ranges.

Super-Nyquist Operation

When the Nyquist sampling frequency rule is broken, a series of frequency reflections takes place. If the sampling frequency is f_C then the Nyquist frequency is $f_C/2$. The frequency band from DC – ($f_C/2$) is reflected at the Nyquist frequency and has a direct 1:1 mapping to the band f_C - ($f_C/2$) respectively. This mapping of the same data into other frequency bands is called Aliasing. For illustration, if f_C is 500 MHz, the Nyquist frequency is 250 MHz, so a signal present at 200 MHz has a mirror image counterpart at 300 MHz. The truth of this is shown in Figure 5 Reconstruction of 200 MHz and 300 MHz signals from data sampled with a 500 MHz clock, note that the sampling intervals are marked in red and both the 200 MHz and 300 MHz waves have the same values at these points in time. Thus from the same data both 200 MHz and 300 MHz signals are valid solutions and either may be extracted with suitable filtering.

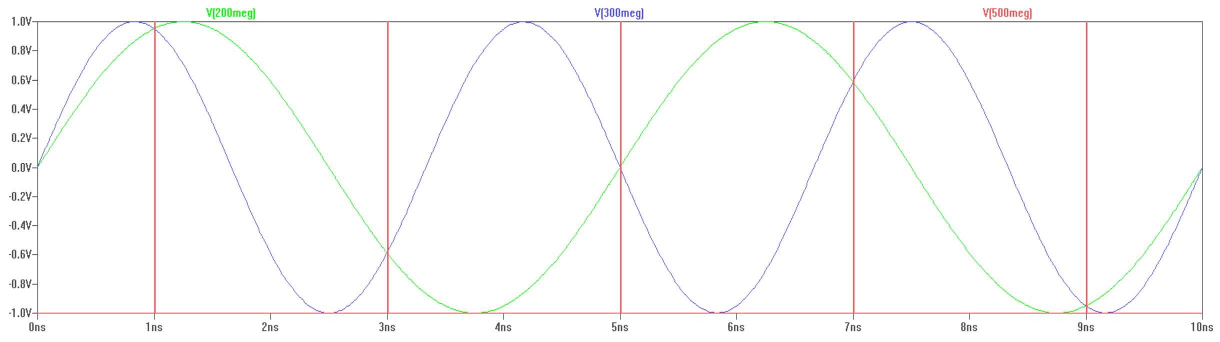

Figure 5 Reconstruction of 200 MHz and 300 MHz signals from data sampled with a 500 MHz clock

If the data comprises just the sample points marked with the cyan line, a low-pass filter will recover the 200 MHz wave (green), or a band-pass filter can be used to recover the 300 MHz wave (blue).

Of course, the frequency band does not stop at f_C, or 500 MHz in the case of the illustration given, it continues to higher frequencies. This block reflects at all integer multiples of f_C upwards. Just as 200 MHz is the image in the base-band range, usually referred to as the First Nyquist zone; and 300 MHz is the image if the filtering had selected the Second Nyquist zone; so does this pattern continue, in theory ad infinitum. The extraction of the alias from any frequency range just requires the appropriate band-pass filter.

While it is possible to extract higher frequency components, there are limitations to consider. The first of these is that the available output is fundamentally limited by the sin(x) / x function, otherwise known as the "sinc" function. This results in the available signal dropping to zero at all integer multiples of the sampling frequency f_C. Attempting to generate frequencies close to DC in base-band result in very low amplitude signals in the aliases in higher Nyquist zones. This is shown in Figure 6 Nyquist zones for a 0.9 GHz sampling clock with example filter for 3.0 GHz signal. The maximum possible amplitude available is also shown to decrease with increasing frequency according to the 1/x asymptotes require.

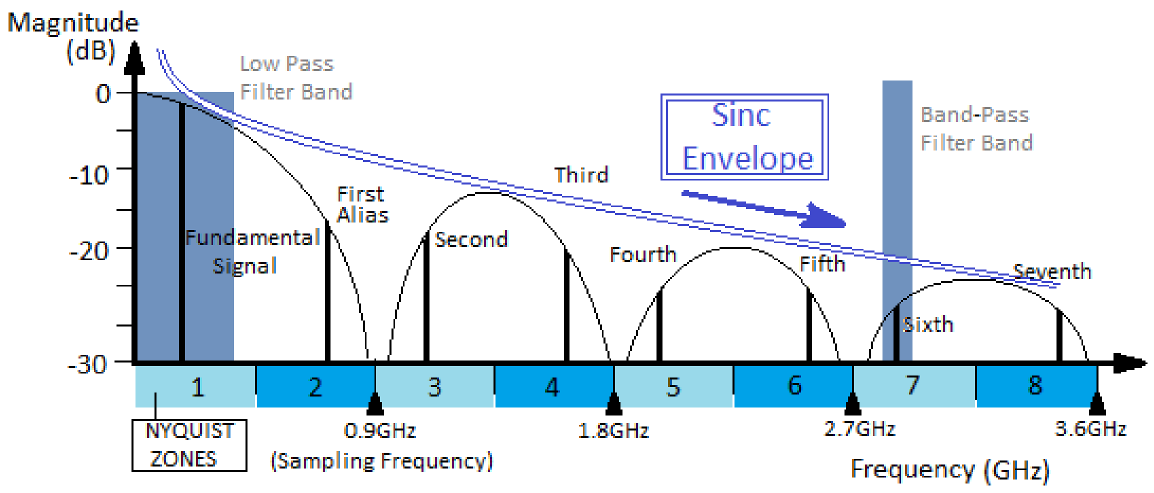

Figure 6 Nyquist zones for a 0.9 GHz sampling clock with example filter for 3.0 GHz signal

It might be thought that as the samples at higher frequencies are attenuated, but by progressively smaller amounts, it should be possible to use the 50th Nyquist zone to extract very high frequencies. The reality is that this is generally not possible for two key reasons. Firstly the width of the band-pass filter and the cut-off rate become unreasonably narrow. Secondly, the assumption that the DAC has an infinite slew rate is of course not the case and the device also produces bandwidth related attenuation. A DAC with a sampling frequency of 2.3 GHz might have a -3dB bandwidth of 2.1 GHz. This figure is relative to the *sinc* waveform and means the *sinc* function will determine the level of a full-scale signal at 2.1 GHz, but the actual level coming out of the DAC is 3dB lower.

The Noise floor should also be considered when selecting a DAC. Using a DAC with a low noise floor should allow operation in the frequency range 2.900 – 3.100 GHz. By looking at the difference between the signal level required and the achievable noise floor allows a decision to be made on the use of a lower cost DAC that meets requirements. Thus if the sinc function predicts a 25dB signal loss and the noise floor of the DAC is 70dB below the level of the baseband signal, there still remains a 45dB headroom which in many applications is quite sufficient. If requirements are for 60dB headroom above the noise floor a higher specification DAC is probably needed.

In choosing to use super-Nyquist operation, and the actual sampling frequency is about 1 GHz, an 8-bit DAC will require data at 8 Gbps. If a higher resolution DAC is chosen, this figure goes up. This has the effect of making a single high-speed serial link between the FPGA and the DAC difficult or impossible with today's technology without incurring significant cost increases.

The alternative to a fast serial link is a parallel interface architecture. Many high performance DACs have a two-port interface. When the dual-port standard mode is used, the data width is twice the bit width of the DAC and therefore runs at half the speed. So if the DAC is 11 bits resolution, there will now be a 22-bit interface, but it only needs to run at 500 MHz allowing the use of the low-cost FPGAs.

Choosing Sampling Frequencies

The frequency range stated as the required band is 200 MHz wide. It will have been noticed that the *sinc* function shows that the output level spread across a 200 MHz frequency range shows about 2-3 dB drop-off for the base-band but is a steeper slope at the S-Band frequencies. The curve can be predicted quite easily, but it is much nicer if this can be ignored completely so that the only significant effect to have to deal with is being on the bandwidth cut-off slope. This may be quite feasible if a little more of the requirements can be pinned down. In many circumstances, the need to operate anywhere in the frequency band exists, but if a chirp is required, it is very unlikely to need to sweep the entire 200 MHz allocation. Instead, suppose the chirp is only required to sweep 30 MHz, a width as narrow as this on the *sinc* curve represents a little over 1 dB of difference when the starting frequency is on the flatter part of the curve. This is something that can be quite easily managed.

Generating frequencies at or close to ¼ f_C has many benefits. First, it is fairly close to the peak of the *sinc* curve for that particular Nyquist zone – so a good signal level is feasible and the amplitude variation over a frequency sweep of say 30 MHz is probably small enough to be safe to ignore. Second, it is the point at which the aliases in other Nyquist zones will be the furthest away – which eases the design of the band-pass filter on the DAC output. The DAC will need either the true clock frequency or a lower frequency that it can multiply up if it has an internal PLL. For now, presume no PLL exists in the DAC, in which case the true clock frequency has to be generated externally.

If the decision is to use the seventh Nyquist zone, and that the frequency needs to be about ¼ f_C it follows that the frequency to synthesise by the DAC is given by this formula:

$$f_{DAC} = \frac{f_{Sband}}{13}$$

Also the corresponding sampling clock f_C is four times this figure.

Thus the range 2.900 GHz to 3.100 GHz requires the sampling clock frequencies f_C to be 892.31 MHz to 953.85 MHz.

Considering the positions of the aliases will show how steep the skirts of the band-pass filter need to be. The closest side is the one governed by the lowest sampling clock frequency and thus will be closest to the high end of the band.

$$f_{hialias} = f_{SbandLow} + \frac{2\mathrm{f}_{SbandLow}}{13}$$

The low side alias is given by

$$f_{loalias} = f_{SbandHigh} - \frac{2\mathrm{f}_{SbandHigh}}{13}$$

Thus the figures are the high-alias is at 3346 MHz (i.e. 246.15 MHz above 3.1 GHz) and the low-alias is at 2623 MHz (i.e. 276.93 MHz below 2.9 GHz).

From here, how steep the skirts of the band-pass filter need to be can be calculated. It is important to add that clock break-through from the DAC should be expected at 3 f_C so the highest frequency will be closest to being in-band. In this case it will be at 2.8616 GHz. While this sounds very close to the pass-band, a good DAC should suppress this and make it at least 60 dB below the base-band signal level. In the seventh Nyquist zone, it works out to being about 30-40 dB below the intended output.

The other two significant remarks that need to be made before closing this section concern accuracy of a generated sweep and start/stop time.

Unlike harmonics where a 10 MHz change in the fundamental results in the n^{th} harmonic changing by n * 10 MHz; aliases retain a 1:1 scaling, thus they are the same change whatever Nyquist zone is being used. This means that if the accuracy of the frequency sweep at base-band is correct to within 100ppm, then the proportional error in S-Band is reduced in proportion to the ratio from base-band to S-Band. Thus for the seventh Nyquist zone, this factor is $1/13^{th}$, or in this example about 7.7ppm.

The matter of the time taken to start producing S-Band output from the time that the base-band signal is generated can be approximated to 1 complete base-band cycle. Thus if the base-band output is 230 MHz, the S-Band output is seen having ramped up to full amplitude in approximately 4.5 ns. The time to stop producing S-Band output relative to the base-band output is the same. This too suggests that the sweet-spot of operation is at or close by the f_C / 4 frequency.

FPGA Requirements

With the DAC related aspects known, the FPGA can be determined. In the example using an 11bit DAC, we have a dual 11-bit data interface. The DAC is clocked at up to 953.85 MHz, so because two 11-bit words are transferred to the DAC in each operation, the FPGA-DAC link therefore runs at up to 476.93 MHz. This sort of transfer rate exceeds the capabilities of the many single-ended interface standards that FPGAs support. The use of differential signalling becomes mandatory. The commonest interface standard for these sorts of transfer rates is LVDS. Although there are twenty two differential pairs, plus the clock pair; the fact that the transfer rate of just under 500 MHz is less than half the speed that this interface could be run at, means that signal flight-time matching is not a critical area of the PCB design.

For the low-cost FPGAs to be able to support twenty two LVDS differential pairs plus a clock pair is not usually too troublesome. The primary factor is whether the internal design of the FPGA can be made to run fast enough. Running the internal clocks at 476.93 MHz and accessing the Sine table values at this rate is not reasonable in the low-cost FPGAs. However, there is no need to do it this way.

FPGAs may be run quickly, but the main way by which they achieve their high levels of performance is through parallelism. Even running the device at 476.93 MHz would still have required 2-fold parallel architecture to be used because the DAC really runs at twice that speed. It is only the data path that has been made wider to slow it down.

In many parallel architecture systems the number of parallel computation blocks is often chosen to be a power of 2. This is because many operations are much easier to scale for as multiplication or division by powers of 2 is extremely simple, just like multiplication and division by powers of 10 in decimal is similarly trivial. For this system, 2-fold parallelism needs a 476.93 MHz clock, and is too fast to easily implement. A 4-fold system needs a clock of 238.46 MHz, which is feasible. But 8-fold parallelism is better, requiring a clock of 119.23 MHz.

The reason why 8-fold parallelism makes a good choice is due to how to perform the data multiplexing to compress 8 channels at 119.23 MHz into 2 channels to the DAC at 476.93 MHz. The newer FPGA families include special SERDES blocks (SERialiser / DESerialiser) in many of the low-cost FPGAs. The SERDES blocks are a much simpler implementation of the high-speed serial transceiver functions. Both perform serial / parallel conversion operations but the SERDES blocks able to run at about 1 Gbps whilst the high-speed transceivers are usually good for 6-12 Gbps. The SERDES blocks are custom hardware built into the device for the specific purpose of taking a slow but wide data channel and reducing it down to a fast and narrow channel. As is always the case, making use of the dedicated functions in the FPGA is more efficient and simpler than making the multiplexers from first principles. When such multiplexing is done, the choice of the input width is determined by what the hardware can support. The option of 2:1 may not be available, but a 4:1 multiplexer certainly is. Hence two lanes with 4:1 multiplexing behind it requires 8-fold parallelism.

Apart from the I/O, the two other key factors to consider with FPGAs are the amount of logic elements and the amount of RAM. Sometimes a figure giving an estimate of the equivalent number of logic gates is given, but this is not a good metric to use for comparison. When we consider implementing the design for the Phase Angle Incrementing Sine Generator shown in figure 2, the phase accumulator requires 32 logic elements (although the number used to index the RAM is truncated to the top 12 bits only). The adder will also use about 32 logic elements (these may be part of the phase accumulator if the compiler chooses to implement it that way, but for now the assumption will be that the functions are not combined). The phase increment value will be assumed to be held in a register of 32-bits. This part requires 32 logic elements, but could probably be shared between all eight Sine Generators (for now assume that this is not the case). There will also be the need to pre-set the initial phase angles in all eight Sine Generators, but this shouldn't take any more logic than is already accounted for. Thus the complete logic usage is around 100 logic elements per Sine Generator.

In addition to this, there needs to be an amount of RAM to perform the phase angle to amplitude look-up function. If the DAC needs 11 bits of data and the chosen phase accumulator resolution for indexing is chosen to be 12 bits, there will be 4096 11-bit words, or 45056 bits needed to store one sine wave cycle. In the Altera devices the FPGA memory blocks are 10 kbits in size, but because of data widths, eight are likely to be needed. In Xilinx devices the FPGA memory blocks are 36 kbits in size, and so two will be needed. A very nice FPGA feature is that the RAM is of the true Dual-Port variety, which means two different accesses can be taking place on the same clock cycle. Thus only half the amount of RAM is necessary because each block is shared between two Sine Generators.

Looking at the smallest member of the low-cost families, both the Altera and Xilinx devices have around 20,000 logic elements available. Thus in terms of logic, about 5% of the device is needed. Remembering that the Xilinx RAM blocks are about 4 times bigger than the Altera ones; the smallest Altera part has 176 RAM blocks, thus about 18% of these are needed. The smallest Xilinx part has 25 RAM blocks of which 8 are needed, which is about 32%.

Overall, it is immediately obvious that if even the smallest members of the low-cost families have sufficient logic and memory resources, the choice of which part to use is primarily governed by the amount of IO needed for the FPGA to DAC link.

FPGA Enhancements – Example of a Linear Frequency Chirp

Up to this point, the FPGA has only been acting as a simple Sine Generator. Now it's time to expand this with the example of generating a linear chirp.

A chirp starts at one frequency f_S and rises linearly to another frequency f_E at which the chirp ends. The clock frequency provided to the DAC is f_C which is $4f_S$ by definition (it could be otherwise, but it makes things simple using this figure).

In this example, the length of the chirp needs to be 10us. If the chirp begins at 2.950 GHz and must sweep to 2.965 GHz, the phase increment needs to change, see Figure 4 Phase incrementing method. In the following the phase angle is normalised such that the true angle in degrees is divided by 360. This is because it simplifies the conversion to binary which for a 32-bit register is best understood as 0 represents 0 and 2^{32} represents 1 (i.e. 360 degrees). Note that expressing numbers in binary makes them hard to read, so hexadecimal will be used instead; when it is used the number will take the prefix “0x”.

$$f_S = \frac{f_{Start}}{13} = 226.92MHz$$

$$f_C = 4 \times f_S = 907.69MHz$$

$$PhaseAngle_{Start} = \frac{f_{Start}}{f_C} = 0.25 = 0\text{x}40000000$$

$$PhaseAngle_{End} = \frac{f_{End}}{f_C} = \frac{f_{Start} + 15\text{MHz}}{f_C} = 0.2665 = 0\text{x}44395810$$

$$PhaseAngle_{change} = PhaseAngle_{End} - PhaseAngle_{Start} = 0.0165 = 0\text{x}04395810$$

$$PhaseAngle_{ChangePerSample} = \frac{PhaseAngle_{Change}}{NumberofSamples} = \frac{PhaseAngle_{Change}}{duration \times f_C} = 0\text{x}00001\text{E}7\text{F}$$

Thus with every phase angle, the value of the phase angle also needs to be incremented by this amount. It is clear that this incrementing of the phase angle (M) can be done in much the same way as the overall angle is modified. This roughly doubles the number of logic elements needed, but does not affect the amount of RAM needed. This brings the requirements up to almost 10% of the capacity of the smallest parts available.

Further FPGA Enhancements

Other modulation of the output signal might include changes in signal level. This kind of operation is just a multiplication by a scale factor. The low-cost families of FPGAs include DSP blocks that permit 18-bit x 18-bit multiplication. This therefore gets added after the Sine table look-up but before being passed to the SERDES blocks. As the smallest FPGAs have around 50 such DSP blocks, of which 8 are required, it follows that this can be added using 16% of the available resource.

So far dedicated DDS parts have been able to match these abilities, but there are a number of other things that put the FPGA based solution in a league of its own. Some common modulation schemes involve set phase changes at given points in time according to the data pattern to be conveyed. This too is very easy to do given the way how the phase accumulator works. Adding in another offset according to the data pattern it simple. We have shown how frequency transitions can be made progressively over a moderate number of data samples, but it is also possible to frequency hop almost instantaneously. Most of the time when DDS parts perform this operation, they do so with an abrupt phase change too, but in the FPGA method, this may be done in either an abrupt way or it can be done in a phase coherent manner.

Generating chirps using linear methods is easy, but it is also possible to create hyperbolic modulated signals too. Depending on the aims, practically any arbitrary modulation method of changing the phase value can be done.

Perhaps the best overall feature though is future-proofing. FPGAs are configured to perform the desired function each time they are powered up. The configuration image is often stored in either a dedicated Flash memory, or it may be part of the system memory. In both cases, the images can be changed permitting the update of hardware in field.

Conclusion

FPGA/DAC based systems for generating S-Band signals have much to commend them. The fact that the cost of the hardware can be brought right down through being able to make use of the lowest cost FPGA families is the starting point. From there, the winning features centre on the inherent flexibility that FPGAs offer as standard. This includes being able to configure the system to have various different modulation mechanisms applied, instantaneous frequency change and also the signal output level which may be applied individually or in combination. This may also provide the ability to remove the need for later processing of the signal further down the signal path since it may be possible to do the same thing in the digital domain inside the FPGA.

As a consequence, the use of FPGAs with an external DAC as an approach to DDS is worthy of serious consideration in practically all places where pure DDS is the first thought that comes to mind, and in many more complex systems too.