LogiCORE IP DDS Compiler v5.0

DS794 March 1, 2011 **Product Specification**

Introduction

The Xilinx LogiCORE™ IP DDS (Direct Digital Synthesizer) Compiler core implements high performance, optimized *Phase Generation* and *Phase to Sinusoid* circuits with AXI4-Stream compliant interfaces.

The core sources sinusoidal waveforms for use in many applications. A DDS consists of a *Phase Generator* and a *SIN/COS Lookup Table (phase to sinusoid conversion)*. These parts are available individually or combined via this core.

Features

- Drop-in module for Virtex®-7 and Kintex™-7, Virtex®-6 and Spartan®-6 FPGAs
- AXI4-Stream-compliant interfaces
- Phase Generator and SIN/COS Lookup table can be generated individually or together with optional dither to provide a complete DDS solution
- High speed, including optimal and optional use of XtremeDSP™ slice
- Sine, cosine, or quadrature outputs
- Lookup table can be stored in distributed or block RAM.
- Optional phase dithering spreads the spectral line energy for greater Spurious Free Dynamic Range (SFDR).
- Phase dithering or Taylor series correction options provide high dynamic range signals using minimal FPGA resources. Supports SFDR from 18 dBs to 150 dBs
- Up to 16 independent time-multiplexed channels
- Optional channel output indication for multi-channels.
- Fine frequency resolution using up to 48-bit phase accumulator with XtremeDSP slice or fabric options
- 3-bit to 26-bit signed output sample precision

LogiCORE IP Facts	
Core Specifics	
Supported Device Family[1]	Kintex-7, Virtex-7 Virtex-6, Spartan-6
Supported User Interfaces	AXI4-Stream
Configuration	See Table 8 through Table 11
Provided with Core	
Documentation	Product Specification
Design Files	Netlist
Example Design	Not Provided
Test Bench	VHDL
Constraints File	N/A
Simulation Model	VHDL and Verilog
Tested Design Tools	
Design Entry Tools	CORE Generator 13.1 System Generator for DSP 13.1
Simulation	Mentor Graphics ModelSim 6.6d Cadence Incisive Enterprise Simulator (IES) 10.2 Synopsys VCS and VCS MX 2010.06 ISIM 13.1
Synthesis Tools	N/A
Support	
Provided by Xilinx, Inc.	

1. For the complete list of supported devices, see the release notes for this core.

Features (continued)

- Optional phase offset capability allows multiple synthesizers with precisely controlled phase differences
- Frequency and phase offset may be independently configured as constant, programmable or dynamic (for modulation)
- Choice of amplitude modes allows either maximal use of output dynamic range or unit circle amplitude
- Optional inversion of sine or cosine outputs
- GUI entry selectable in terms of System (SDFR and frequency resolution) or Hardware (phase and output width) parameters

Applications

- Digital radios and modems
- Software-defined radios (SDR)
- Digital down/up converters for cellular and PCS base stations
- Waveform synthesis in digital phase locked loops
- Generating injection frequencies for analog mixers

General Description

Direct digital synthesizers (DDS), or numerically controlled oscillators (NCO), are important components in many digital communication systems. Quadrature synthesizers are used for constructing digital down and up converters, demodulators, and implementing various types of modulation schemes, including PSK (phase shift keying), FSK (frequency shift keying), and MSK (minimum shift keying). A common method for digitally generating a complex or real valued sinusoid employs a lookup table scheme. The lookup table stores samples of a sinusoid. A digital integrator is used to generate a suitable phase argument that is mapped by the lookup table to the desired output waveform. A simple user interface accepts system-level parameters such as the desired output frequency and spur suppression of the generated waveforms.

Theory of Operation

The simplest form of the DDS Compiler core uses phase truncation, as shown in Figure 1.

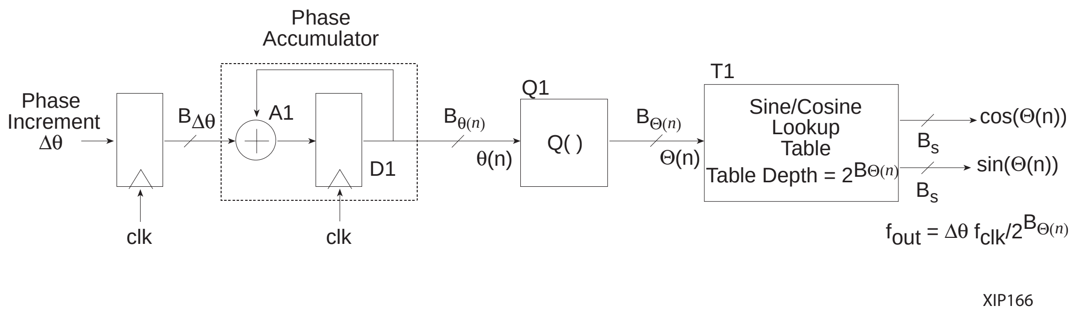

Figure 1: **Phase Truncation DDS (A Simplified View of the DDS Core)**

The integrator (components D1 and A1) computes a phase slope that is mapped to a sinusoid (possibly complex) by the lookup table T1. The quantizer Q1, which is simply a slicer, accepts the high-precision phase angle $\theta(n)$ and generates a lower precision representation of the angle denoted as $\Theta(n)$ in the figure. This value is presented to the address port of a lookup table that performs the mapping from phase-space to time.

The fidelity of a signal formed by recalling samples of a sinusoid from a lookup table is affected by both the phase and amplitude quantization of the process. The depth and width of the lookup table affect the signal's phase angle resolution and the signal's amplitude resolution, respectively. See Spectral Purity Considerations for more details.

Direct digital synthesizers use an addressing scheme with an appropriate lookup table to form samples of an arbitrary frequency sinusoid. If an analog output is required, the DDS presents these samples to a digital-to-analog converter (DAC) and a low-pass filter to obtain an analog waveform with the specific frequency structure. Of course, the samples are also commonly used directly in the digital domain. The lookup table traditionally stores uniformly spaced samples of a cosine and a sine wave. These samples represent a single cycle of a length $N = 2^{B_{\Theta(n)}}$ prototype complex sinusoid and correspond to specific values of the sinusoid's argument $\Theta(n)$ as follows:

$$\Theta(n) = n\frac{2\pi}{N}$$

where n is the time series sample index.

Quarter wave symmetry in the basis waveform can be exploited to construct a DDS that uses shortened tables. In this case, the two most significant bits of the quantized phase angle $\Theta(n)$ are used to perform quadrant mapping. This implementation results in a more resource efficient implementation because the memory requirements are minimized, offering either fewer FPGA block RAMs or reduced distributed memory. Based on the core customization parameters, the DDS core automatically employs quarter-wave or half-wave symmetry when appropriate.[1]

Output Frequency

The output frequency, f_{out}, of the DDS waveform is a function of the system clock frequency, f_{clk}, the phase width, that is, number of bits, $B_{\theta(n)}$, in the phase accumulator and the phase increment value $\Delta\theta$. The output frequency in Hertz is defined by:

$$f_{out} = \frac{f_{clk}\Delta\theta}{2^{B_{\theta(n)}}}$$

For example, if the DDS parameters are:

$$f_{clk} = 120\text{MHz}$$

$$B_{\theta(n)} = 10$$

$$\Delta\theta = 12_{10}$$

then the output frequency is calculated as follows:

1. For very shallow tables, FPGA logic resources are actually minimized by storing a complete cycle. The user is not required to make any design decisions in this context; the CORE Generator software always produces the smallest core possible.

$$
\begin{aligned}
f_{out} &= \frac{f_{clk}\Delta\theta}{2^{B_{\theta(n)}}}\text{Hz} \\
&= \frac{120 \times 10^6 \times 12}{2^{10}} \\
&= 1.406250 \text{ MHz}
\end{aligned}
$$

The phase increment value $\Delta\theta$ required to generate an output frequency f_{out} Hz is:

$$\Delta\theta = \frac{f_{out}2^{B_{\theta(n)}}}{f_{clk}}$$

If we time-division multiplex the DDS core to do multiple channels, then we reduce the effective clock frequency per channel. For C channels, the phase increment required is:

$$\Delta\theta = \frac{Cf_{out}2^{B_{\theta(n)}}}{f_{clk}}$$

Frequency Resolution

The frequency resolution Δf of the synthesizer is a function of the clock frequency and the number of bits $B_{\theta(n)}$ employed in the phase accumulator. The frequency resolution can be determined using:

$$\Delta f = \frac{f_{clk}}{2^{B_{\theta(n)}}}$$

For example, for the following DDS parameters:

$$
\begin{aligned}
f_{clk} &= 120 \text{ MHz} \\
B_{\theta(n)} &= 32
\end{aligned}
$$

the frequency resolution is:

$$
\begin{aligned}
\Delta f &= \frac{f_{clk}}{2^{B_{\theta(n)}}} \\
&= \frac{120 \times 10^6}{2^{32}} \\
&= 0.0279396 \text{ Hz}
\end{aligned}
$$

In the time-division multi-channel case, the frequency resolution is improved by the number of channels, as follows:

$$\Delta f = \frac{f_{clk}}{2^{B_{\theta(n)}} C}$$

Phase Increment

The phase increment is unsigned in the sense that if it needs to be extended it will be padded with zeros rather than sign-extended. However, when the phase increment value width matches the phase width, it can be considered to be unsigned or signed without impact since the range 0 to 2^N describes the range [0,360) degrees whereas the range $-2^{(N-1)}$ to $2^{(N-1)}-1$ describes the range [-180,180) degrees (where N is the number of bits in the phase accumulator). The phase increment term $\Delta\theta$ defines the synthesizer output frequency. Consider a DDS with the following parameterization:

$$f_{clk} = 100\text{ MHz}$$
$$B_{\theta(n)} = 18$$
$$B_{\Theta(n)} = 12$$

To generate a sinusoid with frequency $f_{out} = 19\text{ MHz}$, the required phase increment is:

$$\begin{aligned}\Delta\theta &= \frac{f_{out} B_{\theta(n)}}{f_{clk}} \\ &= \frac{19 \times 10^6 \times 2^{18}}{100 \times 10^6} \\ &= 49807.36\end{aligned}$$

This value must be truncated to an integer giving the following actual frequency:

$$\begin{aligned}f_{out} &= \frac{\Delta\theta f_{clk}}{B_{\theta(n)}} \\ &= \frac{49807 \times 100 \times 10^6}{2^{18}} \\ &= 18.9998627\text{MHz}\end{aligned}$$

Spectral Purity Considerations

The fidelity of a signal formed by recalling samples of a sinusoid from a lookup table is affected by both the phase and amplitude quantization of the process. The depth and width of the lookup table affect the phase angle resolution and the amplitude resolution of the signal, respectively. These resolution limits are equivalent to time base jitter and amplitude quantization of the signal and add spectral modulation lines and a white broad-band noise floor to the signal's spectrum.

In conjunction with the system clock frequency, the phase width determines the frequency resolution of the DDS. The accumulator must have a sufficient field width to span the desired frequency resolution. For most practical applications, a large number of bits are allocated to the phase accumulator to satisfy the system frequency resolution requirements. By way of example, if the required resolution is 1 Hz and the clock frequency is 100 MHz, the required width of the accumulator is:

$$\begin{aligned} B_{\theta(n)} &= \left\lceil \log_2\left(\frac{f_{clk}}{\Delta f}\right)\right\rceil \\ &= \left\lceil \log_2\left(\frac{100 \times 10^6}{1}\right)\right\rceil \\ &= \lceil 26.5754 \rceil \\ &= 27 \text{ bits} \end{aligned}$$

where $\lceil\ \rceil$ denotes the ceiling operator. Due to excessive memory requirements, the full precision of the phase accumulator cannot be used to index the sine/cosine lookup table. A quantized (or truncated) version of the phase angle is used for this purpose. The block labeled Q1 in the phase truncation DDS, Figure 1, performs the phase angle quantization. The lookup table can be located in block or distributed memory.

Quantizing the phase accumulator introduces time base jitter in the output waveform. This jitter results in undesired phase modulation that is proportional to the quantization error, as shown by the following:

$$\begin{aligned} \Theta(n) &= \theta(n) + \delta\theta \\ e^{j\Theta(n)} &= e^{j[\theta(n)+\delta\theta(n)]} = e^{j\theta(n)}e^{j\delta\theta(n)} \\ e^{j\Theta(n)} &\approx e^{j\theta(n)}[1 + j\delta\theta(n)] \\ &\approx e^{j\theta(n)} + j\delta\theta(n)e^{j\theta(n)} \end{aligned}$$

Figure 2 shows the lookup table addressing error, complex output time-series, and the spectral domain representation of the output waveform produced by the DDS structure shown in Figure 1. The normalized frequency for this signal is 0.022 Hz, which corresponds to phase accumulation steps of 7.92 degrees per output sample. The angular resolution of the 256-point lookup table is 360/256 or 1.40625 degrees per address, which is equivalent to 7.92/1.40625 or 5.632 addresses per output sample. Since the address must be an integer, the fractional part is discarded and the resultant phase jitter is the cause of the spectral artifacts. Figure 3 provides an exploded view of the spectral plot in Figure 2 (c).

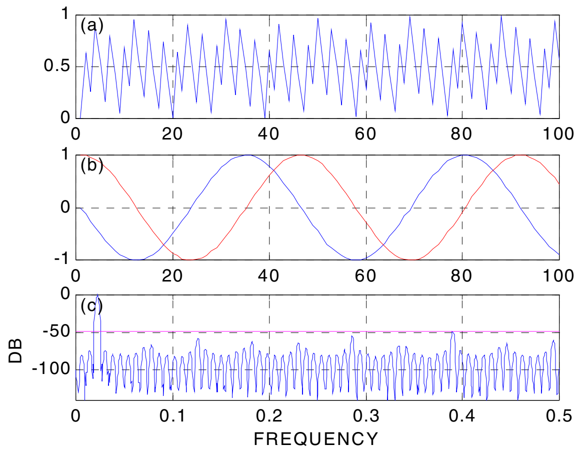

Figure 2: **Phase Truncation DDS. f_{out} = 0.022 Hz, Table Depth = 256 12-Bit Precision Samples. (a) Phase Angle Addressing Error (b) Complex Output Time Series (c) Output Spectrum**

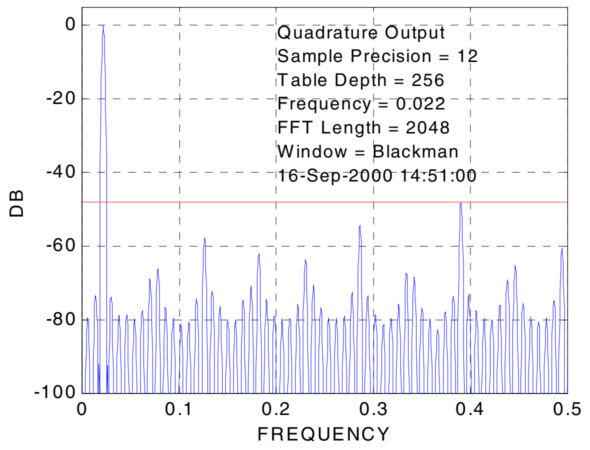

Figure 3: **Exploded View of Figure 2 (c).**

Two observations related to the phase jitter structure level can be made. First, observe that the fractional part of the address count is a periodic (sawtooth) error sequence, which is responsible for the harmonic rich (and aliased) low-level phase modulation evident in Figure 3. Also, the peak distortion level due to incidental phase modulation is approximately 48 dB below the desired signal level, which is consistent with 6 dB/bit of address space. Put another way, if S dB of spur suppression is required in the output waveform, as referenced to the 0 dB primary tone, the DDS lookup table must support at least $\lceil S/6 \rceil$ address bits. For example, if S = 70 dB, which means that the highest spur will be 70 dB below the main signal, then the minimum number of address bits for the lookup table is $\lceil 70/6 \rceil = 12$ bits; that is, a 4096-deep table.

Figures 4 and 5 demonstrate the performance of a similar DDS to the one presented in Figure 2, but in this example, 16-bit precision output samples have been used. Observe that the highest spur is still at the –48 dB level, and allocating four additional bits to the output samples has not contributed to any further spur reduction. For a phase truncation DDS, the only option to further reduce the spur levels is to increase the depth of the lookup table.

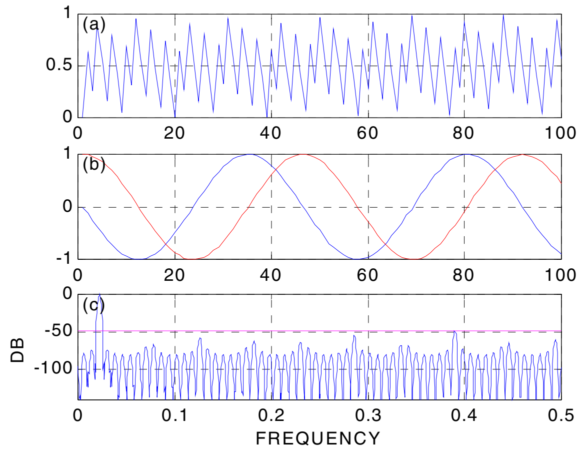

Figure 4: **Phase Truncation DDS. f_{out} = 0.022 Hz, Table Depth = 256 16-Bit Precision Samples. (a) Phase Angle Addressing Error (b) Complex Output Time Series (c) Output Spectrum**

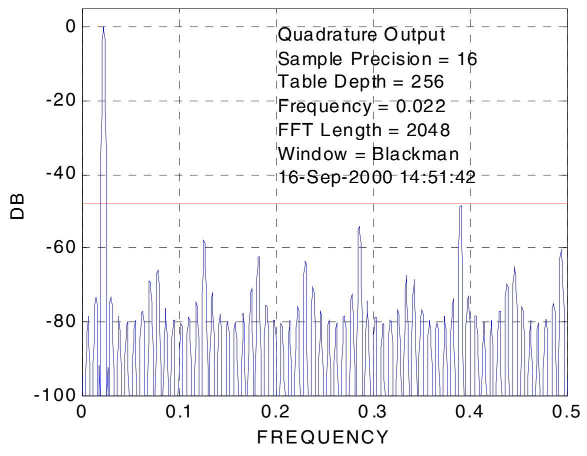

Figure 5: **Exploded View of Figure 4 (c)**

Phase Dithered DDS

In the phase truncation DDS architecture shown in Figure 1, the quantizer Q1 introduces a phase error in the phase slope by discarding the least significant part of the high-precision phase accumulator. The phase error due to the discarded fractional part of the address count is a periodic series which results in an undesired spectral line structure. Figure 6 provides an example of this process for a DDS with a table depth N = 1024 and table sample precision of 16 bits. Figure 6 (a) is the phase error generated by taking the difference between the quantizer input and output signals, Figure 6 (b) is the output time series and Figure 6 (c) is the signal output spectrum. Observe in Figure 6 (a) the periodic sawtooth structure of the phase error signal. The line spectrum associated with this correlated error sequence is impressed on the final output waveform and results in spectral lines in the synthesizer output spectrum. These spurious components can be clearly seen in Figure 6 (c).

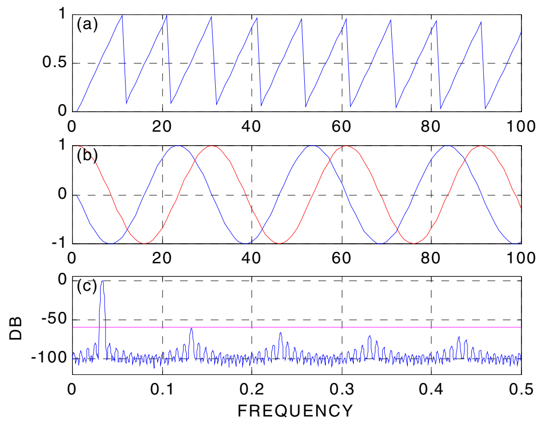

Figure 6: **DDS Plots Showing (a) Phase Error Time Series (b) Complex Output Time Series (c) Output Spectrum. 1024 Deep Lookup Table, 16-Bit Samples, Output Frequency 0.333 Hz**

This structure can be suppressed by breaking up the regularity of the address error with an additive randomizing signal. This randomizing sequence, called *dither*, is a noise sequence, with variance approximately equal to the least significant integer bit of the phase accumulator. The dither sequence is added to the high-precision accumulator output prior to quantization by Q1.

The dithered DDS supplies, approximately, an additional 12 dB of spurious free dynamic range (SFDR) in comparison to a phase truncation design. This is achieved by spreading the spectral energy of the phase error signal. The additional logic resources required to implement the dither sequence generator are not significant.

To provide S dB of spur suppression using a phase truncation DDS, as referenced to the 0 dB primary tone, the internal lookup table must support at least $\lceil S/6 \rceil$ address bits. To achieve this same performance using the dithered architecture requires two fewer address bits, minimizing the number of block RAMs (or logic slices for a distributed memory implementation) used in the FPGA implementation. In summary, for a dithered DDS implementation, the number of address bits needed to support S dB spur suppression is equal to $\lceil S/6 \rceil - 2$.

Figures 7 and 8 provide the results for several dithered DDS simulations. Figure 7 shows eight simulations for a complex dithered DDS employing a table depth N = 4096 and 16-bit precision samples. For each plot the output frequency is different and is annotated on the plot. A phase truncation design would typically generate output spurs 72 dB below the output frequency, independent of the actual value of the output frequency. Indicated on each of the plots by the parameter *A* is the peak spur level achieved for the simulation. The eight spurs are –88.12, –88.22, –86.09, –88.80, –87.21, –87.55, –87.83, –87.12 dB below the output frequency. The worst case value of –86.09 is 14.09 dB better than a similarly configured phase truncation DDS.

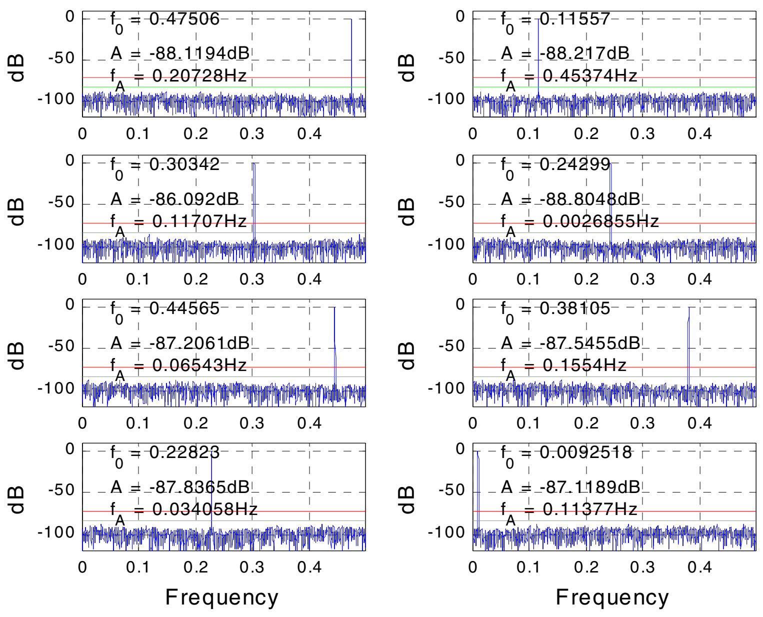

Figure 7: **Dithered DDS Simulations. The DDS configuration is N = 4096, B_S = 16.**

The eight plots are spectral domain representations for eight different output frequencies. Each plot is annotated with the peak spur.

To achieve this same SFDR by extending the table length of a phase truncation design would require increasing the table depth by more than a factor of four.

Figure 8 provides one more dithered DDS simulation where the output frequency is swept over a band of frequencies. The spectrum for each discrete tone in the sweep band is overlaid to construct the final plot. The sweep start frequency, end frequency, number of tones in the sweep, and DDS configuration are annotated on the plot.

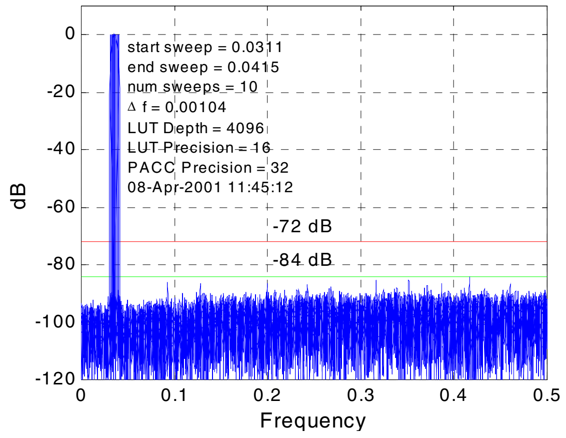

Figure 8: **Example Plot for Dithered DDS Simulation with Frequency Sweep**

In Figure 8, the synthesized signal is swept over a range of frequencies starting from 0.0311 to 0.0415 Hz. There are ten tones in the sweep separated in frequency by 0.00104 Hz. In this example, the phase truncation DDS would produce peak spurs at –72 dB with respect to the 0 dB primary signal. The dithered DDS provides approximately 12 dB better performance with the peak spur –84 dB below the output signal.

A further advantage of the dithered DDS is that the spectral line structure present in a phase truncation design is removed and the out-of-band signal is significantly whitened. This white broadband noise floor is more desirable than the line structured spectrum. In digital communication receivers that use a DDS for generating mixing signals for performing channelization functions, the spurs in a phase truncation DDS can act as low-level mixing tones and cause undesirable spectral contamination of the desired channel. For virtually all applications, the preferred implementation is the dithered DDS.

Taylor Series Corrected DDS

The phase dithered DDS, as well as the phase truncation DDS, have a quantizer Q1 that produces a lower precision $\Theta(n)$ by discarding the fractional component of the high precision $\theta(n)$. The reason for this quantization step is to keep the size of the lookup memory to a reasonable size. The trade-off is spectral purity. With the availability of XtremeDSP slices in FPGAs, it is now practical to use the previously discarded fractional bits to calculate corrections that can be added to the lookup table values to produce outputs with very high SFDR.

Figure 9 through 12 show the simulation results of four different Taylor series corrected DDS simulations. The Taylor series corrected architecture in this example uses a table depth N = 4096 and 18-bit precision samples. However, the precision at the output of the feed-forward error processor is 20 bits. For each plot, the output frequency is different and annotated directly on the plot. A similarly configured phase truncation DDS would produce spurs at –72 dB and a phase dithered DDS at –84 dB. The peak spurs for the four plots are –118.25, –118.13, –118.10, and –118.17 dB below the output frequency.

Figure 13 shows a swept frequency Taylor series corrected DDS. The starting frequency for this example is 0.0313 Hz, the final frequency is 0.0813, and there are 100 tones in the sweep. Using this configuration, a phase truncation DDS would produce peak spurs at approximately 72 dB below the output signal and a phase dithered DDS would produce peak spurs at approximately 84 dB below the output signal. As shown in the plot, the Taylor series corrected DDS produced spurs that are all the way down to 118 dB below the output signal. This result is 34 dB better than the phase dithering DDS, 46 dB better than the phase truncation DDS, and still only consumes a single 18Kb block RAM for the lookup storage. Figure 14 shows another frequency sweep simulation with 35 tones over a broader frequency range.

As shown in the plots, linear correction of the RAM values can extend the SFDR to 118 dB using only a single block RAM and three multipliers. To achieve SFDR beyond 118 dB, it is necessary to deepen the RAM or to use quadratic correction (an extra term of the Taylor series). Since the RAM size would double for each additional 6dB, the DDS Compiler uses quadratic correction to achieve SFDR values of up to 150 dB. Introducing the extra term of the Taylor series expansion of Sine or Cosine requires an additional multiplier per sine and cosine output and an additional block RAM to both scale and square the phase error.

Optimization of Memory Usage

The Taylor Series Correction implementation in the DDS Compiler v5.0 core typically results in an SFDR higher than that requested in order to guarantee SFDR. This results in extra block RAMs for values of SFDR above 102 dBs. However, in many cases, depending on the phase increment values used, a specified SFDR target value of 102 dB will provide higher SDFR, but with one 18k block RAM.

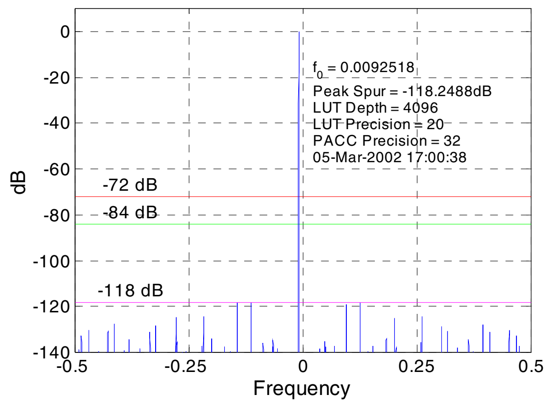

Figure 9: **Taylor Series Corrected DDS – Single-Tone Test, f_0 = 0.0092518**

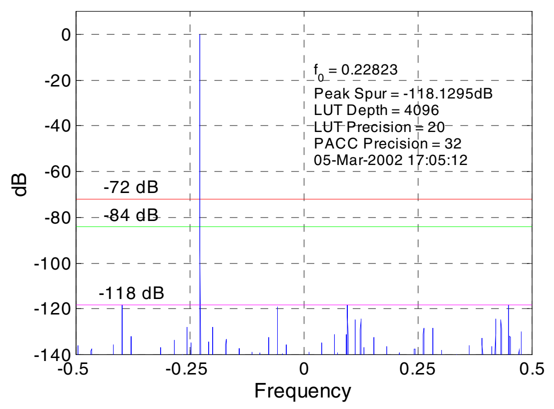

Figure 10: **Taylor Series Corrected DDS – Single-Tone Test, f_0 = 0.22823**

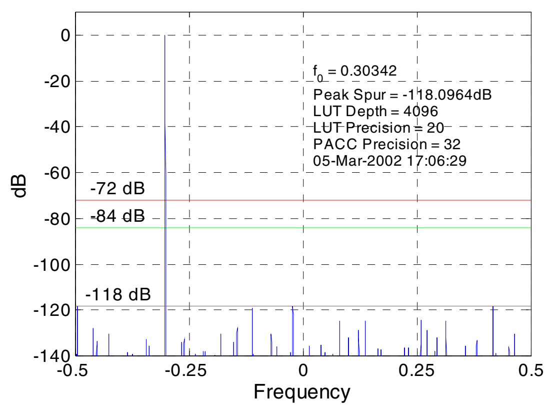

Figure 11: **Taylor Series Corrected DDS – Single-Tone Test, f_0 = 0.30342**

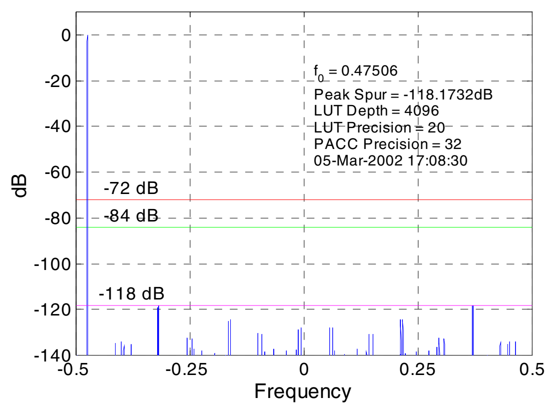

Figure 12: **Taylor Series Corrected DDS – Single-Tone Test, f_0 = 47506**

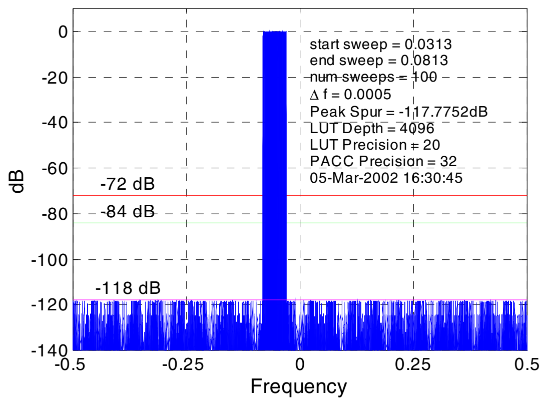

Figure 13: **Taylor Series Corrected DDS – Frequency Sweep Simulation, 100 Tones**

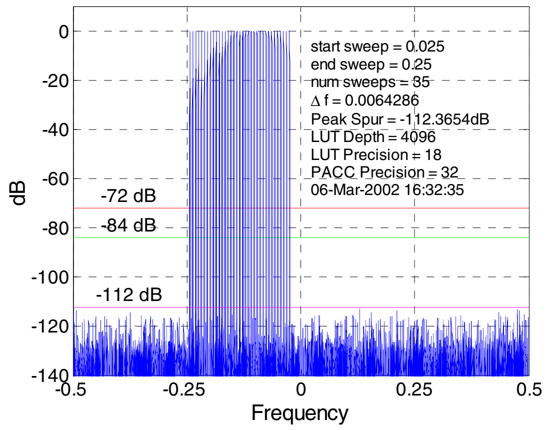

Figure 14: **Taylor Series Corrected DDS – Frequency Sweep Simulation, 35 Tones**

Functional Description

Figure 15 provides a block diagram of the DDS Compiler core. The core consist of two main parts, a Phase Generator and SIN/COS LUT, which can be used independently or together with an optional dither generator to create a DDS capability. A time-division (TDM) multi-channel capability is supported, with independently configurable phase increment and offset parameters.

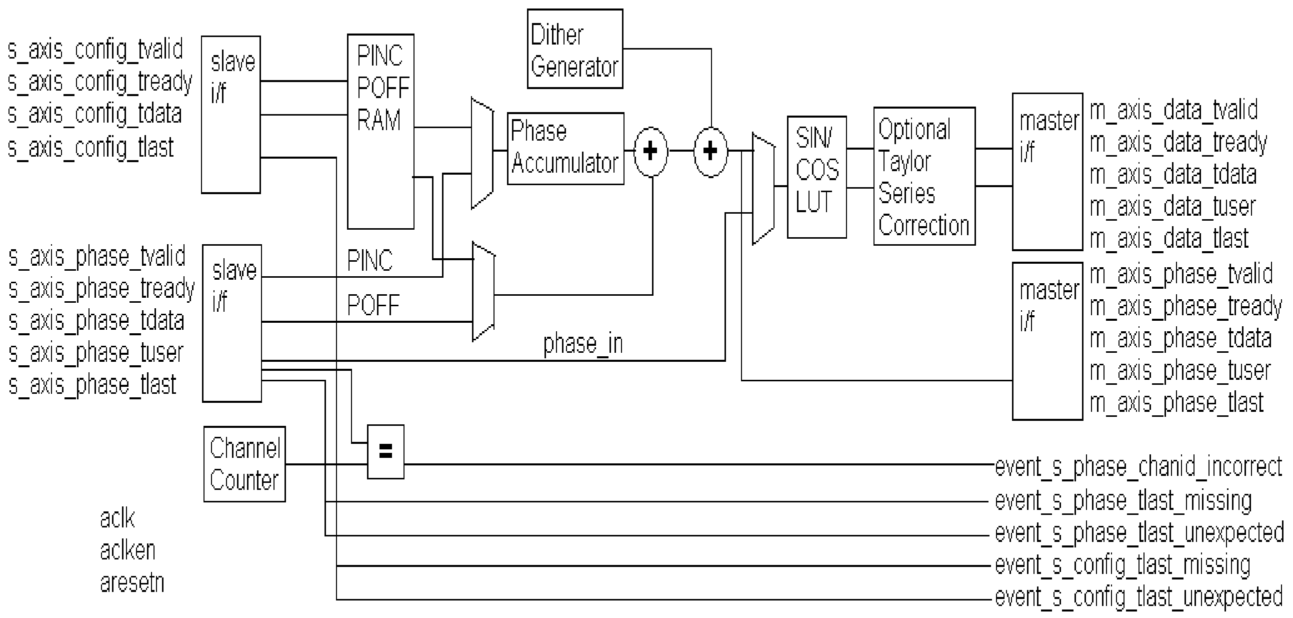

Figure 15: **DDS Core Architecture**

Phase Generator

The Phase Generator consists of an accumulator followed by an optional adder to provide addition of phase offset. When the core is customized, the phase increment (PINC) and phase offset (POFF) can be independently configured to be either fixed, programmable (via the CONFIG channel) or dynamic (via the input PHASE channel).

When set to fixed, the DDS output frequency is set when the core is customized and cannot be adjusted once the core is embedded in a design.

When set to programmable, the CONFIG channel TDATA field will have a subfield for the input in question (PINC or POFF) or both if both have been selected to be programmable. If neither PINC nor POFF is set to programmable, there will be no CONFIG channel.

When set to streaming, the input PHASE channel TDATA field will have a subfield for the input in question (PINC or POFF) or both if both have been selected to be streaming. If neither PINC nor POFF is set to streaming, and the core is configured to have a Phase Generator, then there will be no input PHASE channel.

SIN/COS LUT

When configured as a SIN/COS LUT only, the Phase Generator is not implemented and the PHASE_IN signal is input via the input PHASE channel and transformed into sine and cosine outputs using a look-up table. Efficient memory usage is achieved by exploiting the symmetry of sinusoid waveforms. The core may be configured for sine only output, cosine only output or both (quadrature) output. Each output may be configured independently to be negated. Precision can be increased using optional Taylor Series Correction. This exploits XtremeDSP slices on FPGA families that support them to achieve high SFDR with high speed operation.

Phase Generator and SIN/COS LUT (DDS)

The Phase Generator is used in conjunction with the SIN/COS LUT to provide either a Phase Truncated DDS or Taylor Series Corrected DDS. An optional dither generator can be added between the two blocks to provide a Phase Dithered DDS.

Pinout

The DDS Compiler core pinout is shown in Figure 16. All of the possible pins are shown, though the specific pins in any instance depend upon parameters specified when the core is generated.

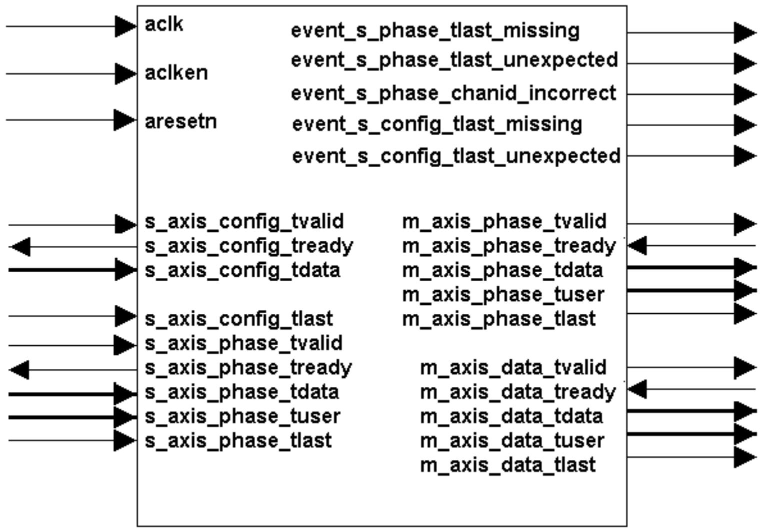

Figure 16: **Core Pinout**

Table 1 summarizes the pinout of the core. If active low input is required for a specific control pin, an inverter must be placed in the path to the pin and will be absorbed appropriately during mapping.

Table 1: **Core Signal Pinout**

Name	Direction	Optional	Description
aclk	Input	no	Rising edge clock
aclken	Input	yes	Active high clock enable
aresetn	Input	yes	Active low synchronous clear. Always takes priority over aclken. aresetn must be driven Low for a minimum of two cycles to reset the core.
s_axis_config_tvalid	Input	yes	TVALID for CONFIG channel
s_axis_config_tready	Output	yes	TREADY for CONFIG channel
s_axis_config_tdata	Input	yes	TDATA for CONFIG channel. See CONFIG Channel TDATA Structure for internal structure and width,
s_axis_config_tlast	Input	yes	TLAST for CONFIG channel. See CONFIG Channel.
s_axis_phase_tvalid	Input	yes	TVALID for input PHASE channel
s_axis_phase_tready	Output	yes	TREADY for input PHASE channel
s_axis_phase_tdata	Input	yes	TDATA for input PHASE channel. See Input PHASE Channel TDATA Structure for internal structure and width.
s_axis_phase_tuser	Input	yes	TUSER for input PHASE channel. See Input PHASE Channel TUSER Structure for internal structure.
s_axis_phase_tlast	Input	yes	TLAST for input PHASE channel. See Input PHASE Channel TLAST Options.
m_axis_phase_tvalid	Output	yes	TVALID for output PHASE channel
m_axis_phase_tready	Input	yes	TREADY for output PHASE channel
m_axis_phase_tdata	Output	yes	TDATA for output PHASE channel. See Output PHASE Channel TDATA Structure.
m_axis_phase_tuser	Output	yes	TUSER for output PHASE channel. See Output PHASE Channel TUSER Structure.
m_axis_phase_tlast	Output	yes	TLAST for output PHASE channel. See Output PHASE Channel TLAST Options
m_axis_data_tvalid	Output	yes	TVALID for output DATA channel
m_axis_data_tready	Input	yes	TREADY for output DATA channel
m_axis_data_tdata	Output	yes	TDATA for output DATA channel. See Output DATA Channel TDATA Structure.
m_axis_data_tuser	Output	yes	TUSER for output DATA channel. See Output DATA Channel TUSER Structure.
m_axis_data_tlast	Output	yes	TLAST for output DATA channel. See Output DATA Channel TLAST Options.

CORE Generator Graphical User Interface

Customization parameter definitions:

Component Name: The name of the core component to be instantiated. The name must begin with a letter and be composed of the following characters: a to z, A to Z, 0 to 9 and "_".

Configuration Options: The full DDS or optionally the Phase Generator part or SIN/COS LUT part may be generated.

- **Phase Generator and SIN/COS LUT:** DDS is provided by combining Phase Generator and SIN/COS LUT with an optional Dither circuit.
- **Phase Generator only:** Only the phase generator is provided.
- **SIN/COS LUT only:** Only the SIN/COS LUT with optional Taylor Series Correction circuit is provided.

System Requirements: The general context of the DDS is set by this group of parameters:

- **System Clock:** The frequency at which the DDS core will be clocked. The value provided influences architectural choices, and is used to calculate the value of phase increment from output frequency (it is the relative value of output frequency to system clock that specifies phase increment, and so doubling System Clock while maintaining output frequency will result in a doubling of phase increment). **The specified clock rate may not be achievable by the final implementation, as this will depend upon the FPGA family and how much is being packed into the device.**
- **Number of Channels:** The DDS and Phase Generator can support up to 16 channels. The channels are time-multiplexed, which reduces the effective clock frequency per channel.
- **Frequency per Channel (Fs):** Because of time division multiplexing, the effective system clock to each channel is the real system clock divided by the number of channels.

Parameter Selection: DDS key parameters may be specified using System Parameters, which are aimed at system architects (frequency domain parameters) or Hardware Parameters, which are aimed at hardware engineers (time-domain parameters). The Phase Generator and SIN/COS LUT are only specified in terms of Hardware parameters.

System Parameters

- **Spurious Free Dynamic Range (SFDR):** The targeted purity of the tone produced by the DDS. This sets the Output Width (as described below) as well as internal bus widths and various implementation decisions.
- **Frequency Resolution:** Specified in Hz, this specifies the minimum frequency resolution and is used to determine the Phase Width, as employed by the phase accumulator and its associated phase increment (`PINC`) and phase offset (`POFF`) values. Small values will give high frequency resolution and will require larger accumulators. Larger values will reduce hardware resources. Depending upon the choice of Noise Shaping, the Phase Width may be increased, and the frequency resolution higher than that specified.

Noise Shaping: This controls whether phase truncation, dithering, or Taylor series correction is used. The options are:

- **None:** Phase truncation DDS is produced.
- **Dithering:** Phase dither is used to improve SFDR at the expense of increased noise floor. See Phase Dithered DDS
- **Taylor Series Corrected:** Sine/cosine values are interpolated using the otherwise discarded bits from phase truncation. See Taylor Series Corrected DDS
- **Auto:** Noise-shaping will be automatically determined, based on System Parameters such as SFDR. The selected noise shaping option is presented in the GUI summary pages. Auto is only available when Parameter Selection is System Parameters.

The availability of particular noise shaping options depends upon the configuration option selected and Parameter Selection method. System Parameter entry will automatically constrain whether a particular Noise Shaping option is possible. When Hardware Parameter entry is selected, the options summarized in Table 2 are made available, and the choice of the Noise Shaping option will then constrain the hardware parameter to ranges to those supported by the selected option.

Table 2: **Availability of Noise Shaping Options for Hardware Parameters**

Setting	DDS Part	Phase Generator Part	SIN/COS LUT Part
None	Available	Available	Available
Dithering	Available		
Taylor	Available		Available
Auto	Available		

Based upon the System Parameters entered and Noise Shaping selected, the minimum Phase Width and Output Width are derived by the GUI in the following way. The Phase Width may be increased to enable a particular Noise Shaping option. For example, Taylor Series Correction requires a minimum Phase Width of 12 bits.

$$\text{Phase Width} = \left\lceil \log_2\left(\frac{\text{DDS Clock Rate}}{\text{Channels} \times \text{Frequency Resolution}}\right)\right\rceil$$

Table 3: **Calculation of Output Width from SFDR and Noise Shaping**

Noise Shaping	Output Width
None and Dithering	$\text{Output Width} = \left\lceil \frac{\text{SFDR}}{6} \right\rceil$
Taylor	$\text{Output Width} = \left\lceil \frac{\text{SFDR}}{6} \right\rceil + 1$

Figure 17 shows the regions of SFDR and Phase Width over which each Noise Shaping option operates. There are three overlapping regions for None, Phase Dithering and Taylor Series Correction, and deeper levels of shading have been used to show where regions overlap. The darkest region is where all 3 regions overlap and all 3 noise shaping options are possible. The lower dashed line signifies that Taylor Series Correction is only valid for SDFR > 66.0 dBs (and not 66.0 dBs). As mentioned previously Phase Width may be increased to maximize the number of noise shaping options for a particular SFDR target.

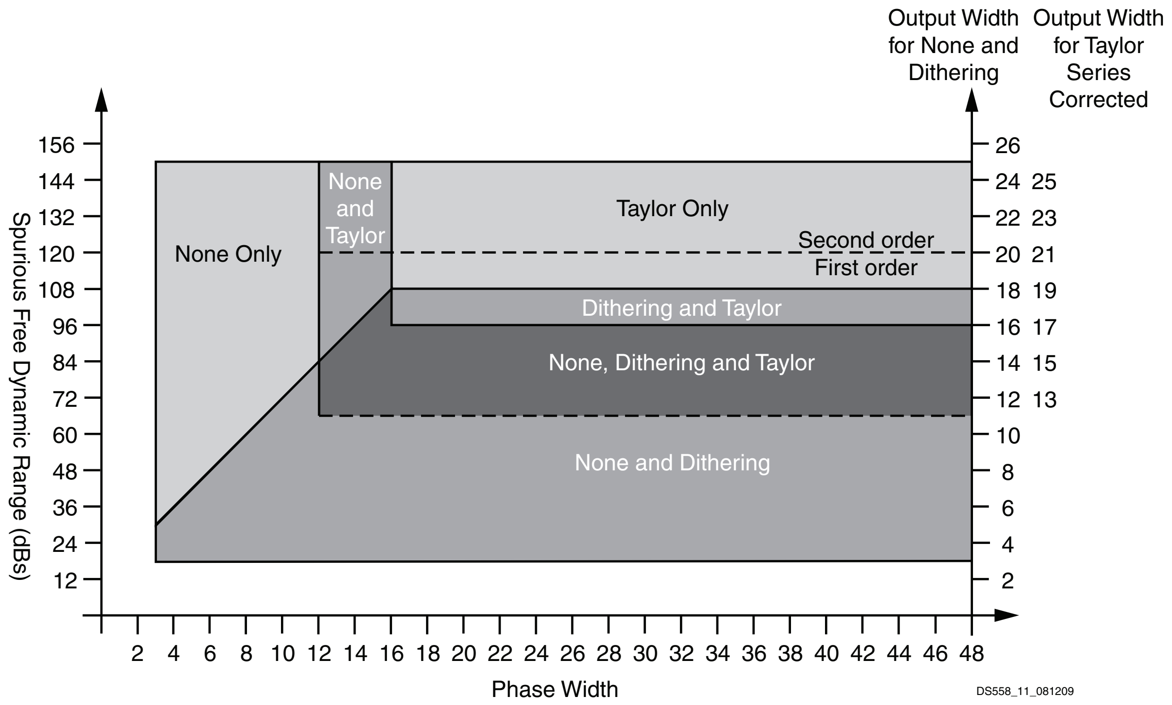

Figure 17: **Noise Shaping Regions**

Hardware Parameters:

- **Phase Width:** Sets the width of the PHASE_OUT field within `m_axis_phase_tdata`, the phase field within `s_axis_phase_tdata` when the DDS is configured to be a SIN/COS LUT only, the phase accumulator, associated phase increment and offset registers and the phase field in `s_axis_config_tdata`.
- **Output Width:** Only enabled when DDS or SIN/COS LUT part selected, as it is not required by the Phase Generator part. Sets the width of SINE and COSINE fields within `m_axis_data_tdata`. The SFDR that this will provide is dependent upon Noise Shaping option previously selected. The following equations can be used to estimate the SFDR that will be achieved:

Table 4: **Calculation of SFDR for given Noise Shaping**

Noise Shaping	SFDR
None, Dither	$\mathrm{SFDR} = \text{Output Width} \times 6$
Taylor	$\mathrm{SFDR} = (\text{Output Width} - 1) \times 6$

Phase Increment Programmability: Selects the means by which the `PINC` value is set.

- **Fixed:** `PINC` is fixed at generation time and cannot be changed at run-time. Fixed requires minimal resource.
- **Programmable:** `PINC` value can be changed at run-time using the CONFIG channel. This is recommended when the DDS frequency is to change between modes of operation.
- **Streaming:** `PINC` value is taken directly from the input PHASE channel. This is recommended when the `PINC` value has to change often, or for example when frequency modulation is required.

Phase Offset Programmability: Selects the means by which the POFF value is set.

- **None:** No phase offset facility and the required hardware is not generated. This saves FPGA resources.
- **Fixed:** POFF is fixed at generation time and cannot be changed at run-time.
- **Programmable:** POFF value can be changed using the CONFIG channel. This is recommended when the DDS phase is to change between modes of operation.
- **Streaming:** POFF value can be changed via the input PHASE channel. This is recommended when the POFF value has to change often, or for example when phase modulation is required.

Output Selection:

- **Output_Selection:** The DDS may have a quadrature SINE and COSINE fields in the m_axis_data_tdata bus, or only one of these two fields. See Output DATA Channel TDATA Structure for m_axis_data_tdata internal structure.
- **Polarity:** The SINE and COSINE fields of m_axis_data_tdata can be inverted. This allows conversion of a DDS used as a transmitter mixer to a receiver mixer, using conjugated outputs; hence both instantiations would be identical except for the values of the two selections here.
  - **Negative Sine:** Checking this selection will result in the SINE field being negated at run-time.
  - **Negative Cosine:** Checking this selection will result in the COSINE field being negated at run-time.
- **Amplitude Mode:** This selection allows for one of two amplitudes from the DDS.
  - **Full Range:** Aimed at communications applications where the maximum amplitude within the two's complement representation is desired, but the exact value of amplitude is not very important (indeed, the target amplitude is $1\text{-}2^{(\text{Output Width-2})}$ in most cases).
  - **Unit Circle:** For applications where the exact amplitude of the DDS output is important, say for FFT twiddle factor generation. When Unit Circle, the DDS output amplitude will be half full range (that is, values will range from 01000..(+0.5). to 110000..(-0.5)). As the amplitude is reduced over Full Range by a factor of 2, the SDFR will be reduced by 6dBs. Increase SFDR or Output Width to accommodate this requirement.

Implementation Options

- **Memory Type:** This controls the implementation of the SIN/COS LUT. The Auto setting will select Distributed ROM for small cases where the table can be contained in a single layer of memory and will select Block ROM for larger cases. (That is, Distributed ROM will be selected when Phase Width $\leq$5-bits). This selection can be overridden by selecting Distributed ROM or Block ROM explicitly.
- **Optimization Goal:** In some cases, circuit clock speed can be increased at the expense of extra pipelining registers. This selection controls whether the implementation decisions will target highest speed or lowest resource.
- **DSP48 Use:** This controls the implementation of the phase accumulator and following addition stages (for phase offset and/or dither noise addition). When set to Minimal, the phase accumulator and following stages will be implemented in fabric. When Maximal, all will be implemented using XtremeDSP slices. In the case of single channel, the XtremeDSP slice can also provide the register to store programmable phase increment and/or phase offset and thereby save further fabric resources. This will not be done if either phase increment or phase offset is Streaming and only when Optimization Goal is Area. When this optimization is performed, the initial value of the PINC and/or POFF register must be zero. This is enforced by the GUI by setting the initial value of PINC and/or POFF to zero and disabling entry.

Latency Options: Select whether Latency should be configured automatically by the GUI or manually:

- **Auto:** Will cause the DDS to be pipelined for optimal performance (taking into account the Optimization Goal).
- **Configurable:** Where optimal performance is beyond requirements, Latency may be set to configurable and a smaller value of latency selected. This will reduce the number of pipeline stages and will generally result in resource savings. A minimum value of latency is imposed, where a cycle of latency arises from each of the following sources:
  - Streaming phase increment
  - Block ROM within SIN/COS LUT (can be avoided by selecting Distributed ROM).
  - Block ROM within second order Taylor series correction (used for SFDR above 120 dBs).

    Note that when TREADY is selected the AXI interfaces buffer data as a FIFO. This buffering results in non-deterministic latency. However, this action can only increase in latency. In this case, the minimum latency possible is 6 cycles plus the minimum latency described above.

Optional Pins: Certain inputs and outputs may be disabled to save resources.

- **Has Phase Out:** When checked the core will have the output PHASE channel.
- **aclken:** When checked the core will have an `aclken` (active high clock enable) port.
- **aresetn:** When checked the core will have an `aresetn` (active low synchronous reset) port.

AXI Channel Options: The action of certain AXI interface signals may be configured.

- **TLAST**: Enabled when there is more than one DDS channel (as opposed to AXI channel). Limited options are also available when only the PHASE channel is present. Options are:
  - **Not Required:** In this mode, no TLAST signals are present on the input PHASE channel or the output channels. In multichannel configurations, TLAST on the CONFIG channel is used to denote the last channel to be reconfigured, and is always present, regardless of this setting.
  - **Vector Framing:** A TLAST pulse on the input PHASE channel and output channels denotes the last channel in a cycle of channels (for example, 12th of 12 channels). If the TLAST pulse is not applied at the correct time to match the core's channel state, an event will be flagged on the `event_s_phase_tlast_missing` or `event_s_phase_tlast_unexpected` event outputs.
  - **Packet Framing:** A TLAST pulse is conveyed from the input PHASE channel to the output channels with the same latency as TDATA. TLAST in this configuration may be used to trigger a reconfiguration. See Config Channel Options on page 23. This mode is intended as a service to ease system design for cases where signals must accompany the datastream, but which have no application in the DDS.
  - **Config Triggered:** This option causes the core to generate an output TLAST on the last TDM channel before a new configuration is applied to the core. Subsequent output samples will be generated using the new core configuration. This mode is only available when the CONFIG channel is present.
- **TREADY**: When selected, the output channels will have a TREADY and hence support the full AXI handshake protocol with inherent back-pressure. If there is an input PHASE channel, the presence of its TREADY is also determined by this control, so that the datapath from input PHASE channel to output channels as a whole supports backpressure or not.

- **TUSER options**: The core supports two distinct uses of the TUSER field; to denote the time-division-multiplex channel index or as conduit to pass a user field (auxiliary data associated with TDATA) from input PHASE channel to output channels. These choices are independent for the input PHASE channel. However, since the selection of a user field implies the desire to convey the TUSER field from input to output, the selection of a user field on the input PHASE channel forces a user field to be present in each of the output channel TUSER ports. Options for the input PHASE channel are shown below. Options for the each output channel are constrained by the input PHASE channel choice, but are otherwise independent.
  - **Not required**: Neither of the above uses is required, the channel in question will not have a TUSER field.
  - **Chan ID field**: In this mode, the TUSER field identifies the time-division-multiplexed channel for the transfer. For the input PHASE channel, this gives the user a mechanism to synchronize to the internal DDS channel state. If the applied Channel ID does not match the core's internal state, an event will be flagged on the `event_s_phase_chanid_incorrect` output.
  - **User Field**: In this mode, the core ignores the content of the TUSER field, but passes it unaltered from input PHASE channel to the output channels.
  - **User and Chan ID field**: In this mode the TUSER field will have both a user field and a channel ID field, with the channel ID field in the least significant bits. The minimal number of bits required to describe the channel will determine the width of the channel ID field, for example, seven channels will require three bits.
  - **User field width**: This field determines the width of the bit field which will be conveyed from input to output unaltered by the DDS. It does not include the width of the Channel ID field, if it is present.
- **Config Channel Options**: This selection deals with the timing of re-configuration when both CONFIG and PHASE channels are present. The configuration channel will take configuration data asynchronously to the phase of the channel counter and store the reconfiguration data in a buffer. This selection determines when that new configuration data takes effect on the datapath.:
  - **On Vector**: In this mode, the reconfiguration data will be applied when the channel counter rolls over to start a new cycle of time-division-multiplexed channels.
  - **On Packet:** In this mode, available when TLAST is set to packet framing, a TLAST on the input PHASE channel will trigger the reconfiguration. This mode is targeted at cases where each set of configuration data is to be associated with the packets implied by the input TLAST indicator.

Parameter Entry Pages: The following pages appear for entry of parameters when either Phase Increment or Phase Offset are either Fixed or Programmable. If Programmable, the initial value of the register is specified through the Parameter Entry Pages. If an XtremeDSP register is used, as described under Implementation Options, the initial value of phase increment and/or offset is assumed to be zero.

System Parameters:

- **Output Frequencies:** This page appears when Parameter Selection is set to System Parameters and Phase Increment Programmability is Fixed or Programmable. For each channel, an independent frequency (MHz) can be entered into the table. The allowable range is displayed as 0 to Fs (where Fs is the frequency per channel). Values from Fs/2 to Fs will alias to -Fs/2 to 0 respectively, so can be used to input negative frequencies.
- **Phase Offset Angles:** This page appears when Parameter Selection is set to System Parameters and Phase Offset is set to Fixed or Programmable. This table allows the phase offset to be specified for each channel as a fraction of a cycle. The valid range is -1.0 to 1.0. For example enter 0.5 for 180 degrees (that is, π radians). This range is greater than a single cycle, but is allowed, as negative values will map to equivalent positive values.

Hardware Parameters:

- **Phase Angle Increment Values:** This page appears when Parameter Selection is set to Hardware Parameters and Phase Increment Programmability is Fixed or Programmable. Values must be entered in binary. The range is 0 to the weight of the accumulator, that is, $2^{\text{Phase Width}}-1$, which corresponds to a single cycle. The angle in radians can be obtained by converting the unsigned fractional number to decimal and multiplying by 2π. Entries will be extended to Phase Width bits by zero padding to the left.
- **Phase Offset Values:** This page appears when Parameter Selection is set to Hardware Parameters and Phase Offset is set to Fixed or Programmable. Values must be entered in binary. The range is 0 to the weight of the accumulator, that is, $2^{\text{Phase Width}}-1$, which corresponds to a single cycle. The angle in radians can be obtained by converting the unsigned fractional number to decimal and multiplying by 2π. Entries will be extended to Phase Width bits by zero padding to the left.

Summary (2 pages): The final two pages of the GUI are devoted to feedback fields.

- **Summary (Page 1):** This page presents the resolved values of the selected part. For instance, these fields indicate the result of automatic memory type and latency allocation. They also indicate the expected SFDR and frequency resolution for the DDS when hardware parameters are used for input, or vice versa. There are also resource estimates (XtremeDSP slices and 18 kbit block RAM primitives).
- **Summary (Page 2):** This is only presented when Phase Increment and/or Phase Offset are fixed or programmable, and provides a summary of the hexadecimal values used to obtain a particular frequency or phase offset. The actual value of frequency and phase (the latter as a fraction of a cycle) is also given as a floating-point number.

Using the DDS Compiler IP Core

Simulation Models

The core has a number of options for simulation models:

- VHDL RTL-based simulation model
- Verilog UniSim-based structural simulation model

The models required may be selected in the CORE Generator software project options.

Xilinx recommends that simulations utilizing UniSim-based structural models are run using a resolution of 1ps. Some Xilinx library components require a 1ps resolution in either functional or timing simulation. The UniSim-based structural models may produce incorrect results if simulated with a resolution other than 1ps. See the "Register Transfer Level (RTL) Simulation Using Xilinx Libraries" section in *Chapter 6* of the Synthesis and Simulation Design Guide. This document is part of the ISE® Software Manuals set available at www.xilinx.com/support/software_manuals.htm.

XCO File Parameters

The XCO parameters are summarized in Table 5.

Table 5: **XCO Parameters**

GUI Field	XCO Parameter	XCO Values[1]	Description
Component name	Component_Name	string	The name of the CORE Generator software instance
Configuration Options	PartsPresent	**Phase_Generator_and_SIN_COS_LUT**, Phase_Generator_only, SIN_COS_LUT_only	Allows for parts of DDS to be instanced separately
System Clock	DDS_Clock_Rate	0.01 to 550, default is **100**	MHz
Number of Channels	Channels	Integer, **1** to 16	Number of time-division-multiplexed channels to implement
Parameter Selection	Parameter_Entry	**System_Parameters**, Hardware_Parameters	
Spurious Free Dynamic Range	Spurious_Free_Dynamic_Range	18[2] to 150, default is **36**	dB
Frequency Resolution	Frequency_Resolution	Frequency per Channel divided by $2^{\text{Phase Width}}$ to Frequency per Channel/2^{48}, default is **0.4**Hz	Hz
Noise Shaping	Noise_Shaping	**Auto**, None, Phase_Dithering, Taylor_Series_Corrected	Available parameters depend on the core configuration
Phase Width	Phase_Width	3 to 48. Default is **16**	Defined width of phase buses hence frequency resolution
Output Width	Output_Width	3[3] to 26, default is **6**	Defines the output width, hence precision
Phase Increment Programmability	Phase_Increment	**Fixed**, Programmable, Streaming	
Phase Offset Programmability	Phase_offset	**None**, Fixed Programmable, Streaming	
Output Selection	Output_Selection	Sine, Cosine, **Sine_and_Cosine**	
Negative Sine	Negative_Sine	**false**, true	
Negative Cosine	Negative_Cosine	**false**, true	
Amplitude Mode	Amplitude_Mode	**Full_Range**, Unit_Circle	Selects maximum possible amplitude or exact power-of-two amplitude
Memory Type	Memory_Type	**Auto**, Distributed_ROM, Block_ROM	
Optimization Goal	Optimization_Goal	**Auto**, Area, Speed	
DSP48 Use	DSP48_Use	**Minimal**, Maximal	
Latency Options	Latency_Configuration	**Auto**, Configurable	The allowed range of latencies depends on whether TREADY is present on the AXI channels present.
	Latency	0,1,...21	
Has Phase Out	Has_Phase_Out	false, **true**	
ACLKEN	Has_ACLKEN	**false**, true	

Table 5: **XCO Parameters *(Cont'd)***

GUI Field	XCO Parameter	XCO Values[1]	Description
ARESETn	Has_ARESETn	**false**, true	
Output TREADY	Has_TREADY	**false**, true	
TUSER Input	S_PHASE_Has_TUSER	**Not_Required,** Chan_ID_Field, User_Field, User_and_Chan_ID_Field	
TUSER (DATA output)	M_DATA_Has_TUSER	**Not_Required,** Chan_ID_Field, User_Field, User_and_Chan_ID_Field	Permitted values depend on TUSER input setting
TUSER (PHASE output)	M_PHASE_Has_TUSER	**Not_Required,** Chan_ID_Field, User_Field, User_and_Chan_ID_Field	Permitted values depend on TUSER input setting
User Field Width	S_PHASE_TUSER_Width	Range **1**..256	
Synchronization Mode	S_CONFIG_Sync_Mode	**On Vector,** On Packet	
Output Frequencies	Output_Frequency1, Output_Frequency2,... Output_Frequency16	**0.0** to Fs (Fs=DDS_Clock_Rate/ Channels)	MHz. Frequencies above the Nyquist limit [Fs/2 to Fs] will alias to negative frequencies [-Fs/2 to 0] respectively.
Phase Offset Angles	Phase_Offset_Angles1, Phase_Offset_Angles2,... Phase_Offset_Angles16	real (-1.0 to +1.0), default is **0.0**	-0.5 is -180 degrees, +0.5 is +180 degrees
Phase Angle Increment Values	PINC1, PINC2, ..., PINC16	**0** to 2**Phase_Width -1	Unsigned binary
Phase Angle Offset Values	POFF1, POFF2, ..., POFF16	**0** to 2**Phase_Width -1	Unsigned binary

1. Default value highlighted in bold.
2. Note that SDFR must be greater than 18 dBs if Phase Dithering is to be used.
3. The minimum value for Phase Width and Output Width is 4 if Phase Dithering is to be used.

Demonstration Test Bench

When the core is generated using CORE Generator, a demonstration test bench is created. This is a simple VHDL test bench that exercises the core.

The demonstration test bench source code is one VHDL file: `demo_tb/tb_<component_name>.vhd` in the CORE Generator output directory. The source code is comprehensively commented.

Using the Demonstration Test Bench

The demonstration test bench instantiates the generated DDS Compiler core. Either the behavioral model or the netlist can be simulated within the demonstration test bench.

- Behavioral model: Ensure that the CORE Generator project options are set to generate a behavioral model. After generation, this creates a behavioral model wrapper named `<component_name>.vhd`. Compile this file into the work library (see your simulator documentation for more information on how to do this).
- Netlist: If the CORE Generator project options were set to generate a structural model, a VHDL or Verilog netlist named `<component_name>.vhd` or `<component_name>.v` was generated. If this option was not set, generate a netlist using the netgen program, for example in Linux:

  ```
  netgen -sim -ofmt vhdl <component_name>.ngc <component_name>_netlist.vhd
  ```

  Compile the netlist into the work library (see your simulator documentation for more information on how to do this).

Compile the demonstration test bench into the work library. Then simulate the demonstration test bench. View the test bench's signals in your simulator's waveform viewer to see the operations of the test bench.

The Demonstration Test Bench in Detail

The demonstration test bench performs the following tasks:

- Instantiate the core
- Generate a clock signal
- Drive the core's input signals to demonstrate core features (see below for details)
- Checks that the core's output signals obey AXI protocol rules (data values are not checked in order to keep the test bench simple)
- Provide signals showing the separate fields of AXI TDATA and TUSER signals

The operations performed by the demonstration test bench are appropriate for the configuration of the generated core:

- If phase increment and offset are fixed:
  - Run to produce sine / cosine / phase outputs
- If phase increment and/or offset are programmable, and neither is streaming:
  - Program an initial configuration
  - Run to produce sine / cosine / phase outputs
  - Program a different configuration
  - Run again to produce sine / cosine / phase outputs
- If one of phase increment or offset are streaming and the other is fixed:
  - Stream in constant phase increment or offset to produce sine / cosine / phase outputs
  - If phase offset is streaming, stream in incrementing phase offset to produce higher frequency sine / cosine / phase outputs

- If one of phase increment or offset are streaming, and the other is programmable:
  - Program an initial configuration
  - Stream in constant phase increment or offset to produce sine / cosine / phase outputs
  - If phase offset is streaming, stream in incrementing phase offset to produce higher frequency sine / cosine / phase outputs
  - Continue streaming in phase increment or phase offset, and simultaneously program a different configuration
- If phase increment and offset are both streaming:
  - Stream in constant phase increment and zero phase offset to produce sine / cosine / phase outputs
  - Stream in zero phase increment and incrementing phase offset to produce sine / cosine / phase outputs
- For SIN/COS LUT only:
  - Stream in incrementing phase to produce sine / cosine outputs
- For all configurations:
  - Demonstrate backpressure by deasserting TREADY of master channels (if TREADY is present)
  - Demonstrate use of clock enable (if present)
  - Demonstrate use of reset (if present)

Customizing the Demonstration Test Bench

It is possible to modify the demonstration test bench to drive the core's inputs with different data or to perform different operations.

All operations performed by the demonstration test bench to drive the core's inputs are done in the `stimuli` process. This process is comprehensively commented, to explain clearly what is being done. New operations, potentially with different input data, can be added by copying and modifying sections of this process.

The clock frequency of the core can be modified by changing the `CLOCK_PERIOD` constant.

System Generator for DSP Graphical User Interface

This section describes the System Generator for DSP GUI and details the parameters that differ from the CORE Generator software GUI.

The DDS Compiler core may be found in the Xilinx Blockset in the DSP section. The block is called “DDS Compiler v5.0.”

See the System Generator for DSP Help page for the “DDS Compiler v5.0” block for more information on parameters not mentioned here.

The System Generator for DSP GUI offers the same parameters as the CORE Generator software GUI. However, whereas in the CORE Generator software the Hardware Parameters are hidden when System Parameter entry is selected, in System Generator for the DSP GUI the Hardware Parameters are simply disabled. Likewise, System Parameters are disabled when Hardware Parameter entry is selected.

AXI4-Stream Considerations

The conversion to AXI4-Stream interfaces brings standardization and enhances interoperability of Xilinx IP LogiCORE solutions. Other than general control signals such as `aclk`, `aclken` and `aresetn` and event indication outputs, all inputs and outputs of the DDS Compiler are conveyed via AXI4-Stream channels. A channel always consists of TVALID and TDATA, plus several optional ports and fields. In the DDS Compiler, the optional ports supported are TREADY, TLAST and TUSER. Together, TVALID and TREADY perform a handshake to transfer a message, where the payload is TDATA, TUSER and TLAST. The DDS Compiler operates on the operands contained in the TDATA field of the input channels and outputs the result in the TDATA field of the output channels. The DDS Compiler provides configuration options for the use of TUSER and TLAST.

Basic Handshake

Figure 18 shows the transfer of data in an AXI4-Stream channel. TVALID is driven by the source (master) side of the channel and TREADY is driven by the receiver (slave). TVALID indicates that the value in the payload fields (TDATA, TUSER and TLAST) is valid. TREADY indicates that the slave is ready to receive data. When both TVALID and TREADY are asserted in a cycle, a transfer occurs. The master and slave will set TVALID and TREADY respectively for the next transfer appropriately.

The DDS Compiler 'datapath' channels (all except the CONFIG channel) can be configured to have no TREADY.

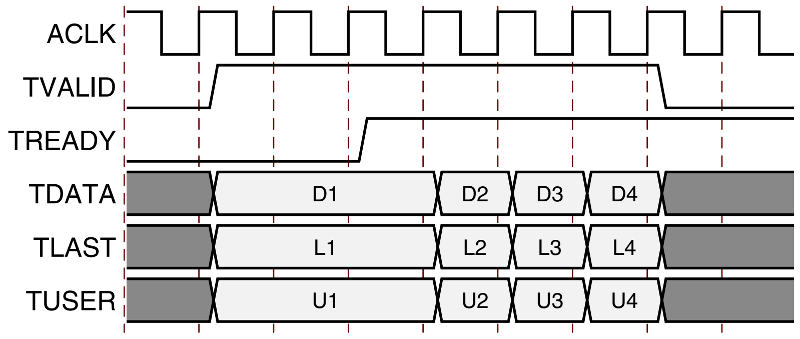

Figure 18: **Data Transfer in an AXI4-Stream Channel**

This is equivalent to setting TREADY for each of these channels permanently asserted. This inability to indicate backpressure simplifies the interface behavior and allows resource savings to be made. This mode is recommended when the system can be designed to ensure that full throughput (one sample per cycle) can be assured.

Note that the AXI handshake shown in the diagram above will forbid the output of data until downstream is ready. This allows for easy synchronization of circuits following power-up or reset without complicated latency calculations.

CONFIG Channel

The CONFIG channel (s_axis_config_t*) replaces the programmatic interface of DDS Compiler v4.0. For the CONFIG channel, there is the concept of a vector. The vector in question is a complete set of values (PINC and/or POFF) for all channels. The CONFIG channel is non-blocking, which means that the other channels of the DDS Compiler will not wait upon data from the CONFIG channel. To program the CONFIG channel N transfers must occur, where N is the number of channels. Each transfer will contain the PINC and/or POFF values for each channel in sequence starting with channel 0. Only the last transfer, for channel (index N-1) must have TLAST asserted. Failure to do so will cause either event_s_config_tlast_missing or event_s_config_tlast_unexpected outputs to be asserted for a cycle. The packet is only deemed to be received when complete. Only when it is completely received is it eligible to be used pending a synchronization event. Synchronization events are either when the TDM channel counter rolls over (vector framing) or when the input PHASE channel is configured to receive packet TLASTs and one such TLAST is received (packet framing).

The diagram below illustrates programming CONFIG data for a six-channel DDS. In the first programming cycle, TLAST is incorrectly applied, and the event outputs trigger accordingly. The second programming cycle shows correct application of TLAST.

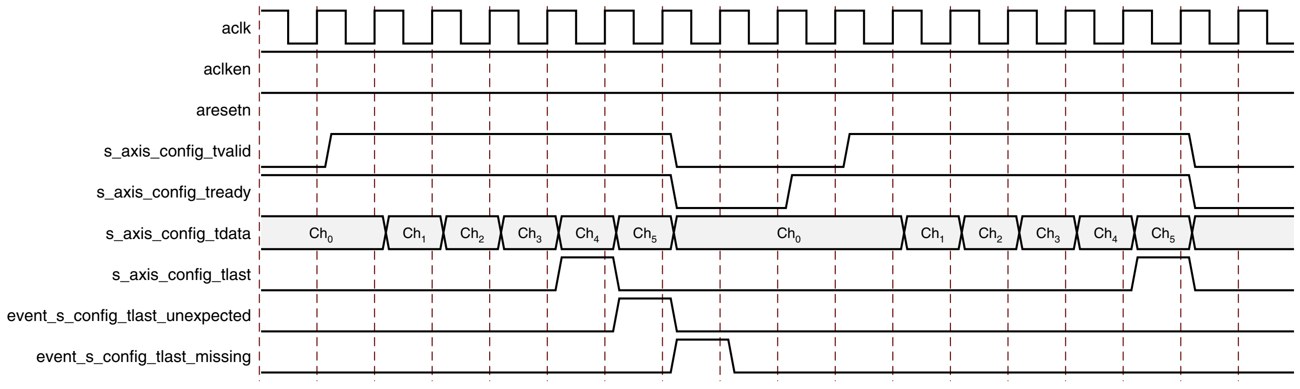

Figure 19: **CONFIG Channel Programming**

When the core is configured for single-channel operation, TLAST is not required and the pin is not present on the CONFIG channel.

CONFIG Channel TDATA Structure

When the CONFIG channel is configured to supply both PINC and POFF values for each TDM channel, each field is sign-extended to fit a byte boundary, then these byte-oriented fields are concatenated with PINC in the least significant positions. For example, for a phase width of 11 bits, PINC would occupy bits 10:0 and POFF would occupy 26:16. Thus `s_axis_config_tdata` would be 31:0 overall.

The diagram below shows the structure for the widths in the preceding example for the following configurations;

a) both PINC and POFF are configured to be programmable.

b) PINC only is set to programmable.

c) POFF only is set to be programmable

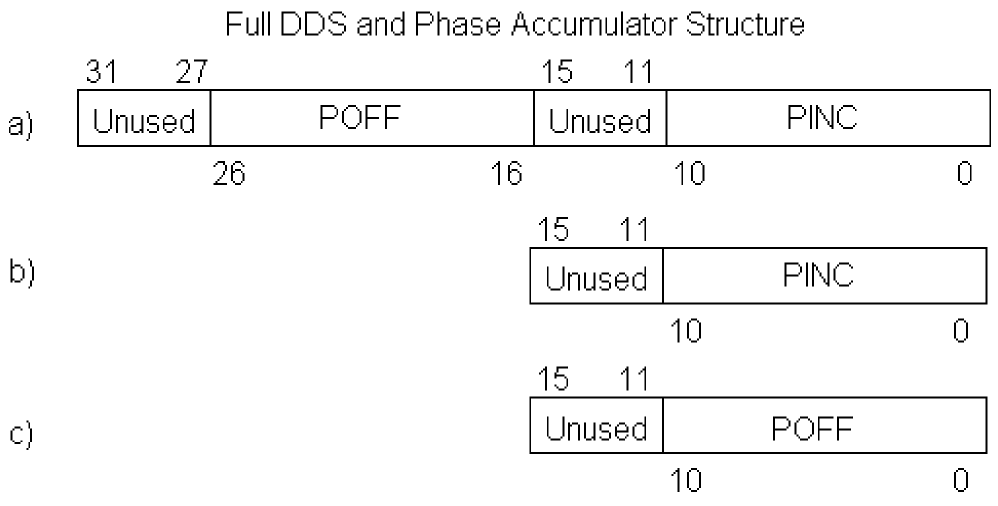

Figure 20: **CONFIG Channel TDATA Structure**

Input PHASE Channel

The input PHASE channel (`s_axis_phase_t*`) replaces the streaming interface (PINC_IN and POFF_IN) or PHASE_IN ports of the DDS Compiler v4.0. The input PHASE channel is intended for applications where the DDS Compiler is to perform a dynamic function such as phase or frequency modulation, where there is an output sample for each input sample. The fact that there is a one-to-one relationship between input and output means that back pressure applied to the output (TREADY deasserted) will result in the deassertion of TREADY on the input PHASE channel (delayed according to internal buffer capacity). Likewise a starvation of input data on the PHASE channel (deassertion of TVALID) will propagate to become a deassertion of TVALID on the output channels.

When the DDS Compiler is configured to have a Phase Accumulator and either Phase Increment or Phase Offset is selected to be 'Streaming' the input PHASE channel interface will exist. When the DDS Compiler is configured to be a SIN/COS LUT only, the PHASE_IN field will be input via the TDATA bus of the input PHASE channel. These two configurations are mutually exclusive.

Input PHASE Channel TDATA Structure

As noted earlier, the two configurations in which the DDS Compiler can have an input PHASE channel are mutually exclusive, so although there are 3 fields which can occur in the TDATA, all 3 cannot occur together.

When the DDS Compiler is configured to be a SIN/COS LUT only, the PHASE_IN field is mapped to `s_axis_phase_tdata`. The PHASE_IN field occupies a byte-oriented field in the least significant portion of the bus. So the width of `s_axis_phase_tdata` will be the minimum multiple of 8 bits required to accommodate the PHASE_IN width. As this is an input, any additional bits required to achieve this byte orientation are simply ignored by the core and will be optimized away during synthesis or mapping.

The diagram below shows the structure of `s_axis_phase_tdata` where Phase_Width = 11 for the following configurations;

a) both PINC and POFF are configured to be programmable

b) PINC only is set to programmable.

c) POFF only is set to be programmable

d) The DDS is configured to be a SIN/COS LUT only

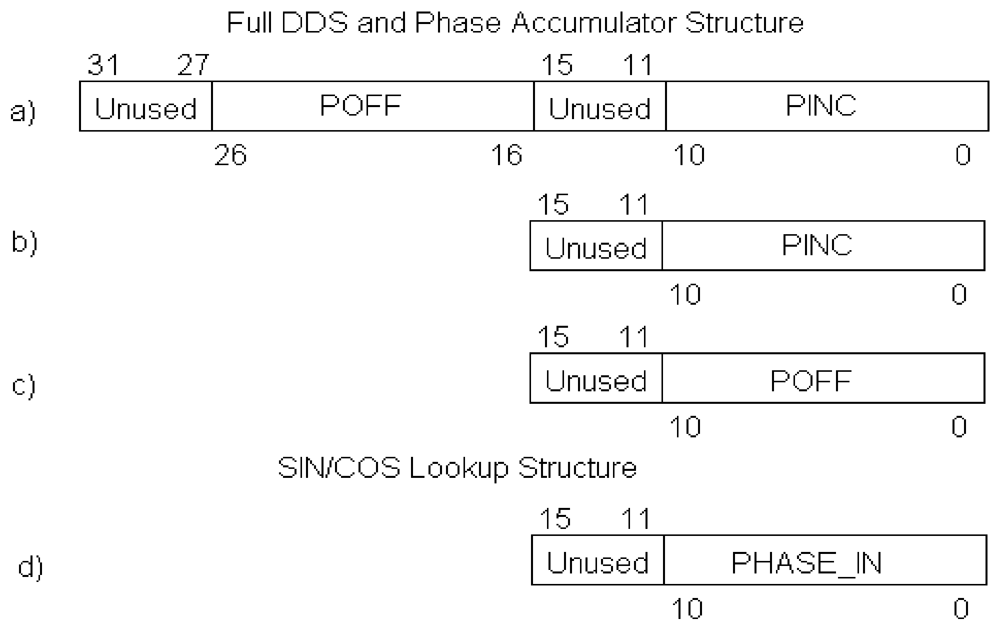

Figure 21: **Input PHASE Channel TDATA Structure**

Input PHASE Channel TUSER Structure

The input PHASE channel may be configured to have no TUSER port, to have a user field or to carry the TDM Channel index, or both a user field and TDM Channel index. There is no byte orientation to these fields. The TDM channel index, if configured, will have the minimum width required to describe the number of TDM channels. The width of the user field is determined by user selection from 1 to 256 bits. The two fields are concatenated with the TDM channel ID field in the least significant place. If only one field exists, it occupies the least significant bits of `s_axis_phase_tuser`.

The diagram below shows the 3 possible combinations; both user field and chan_id field, chan_id field only and user field only.

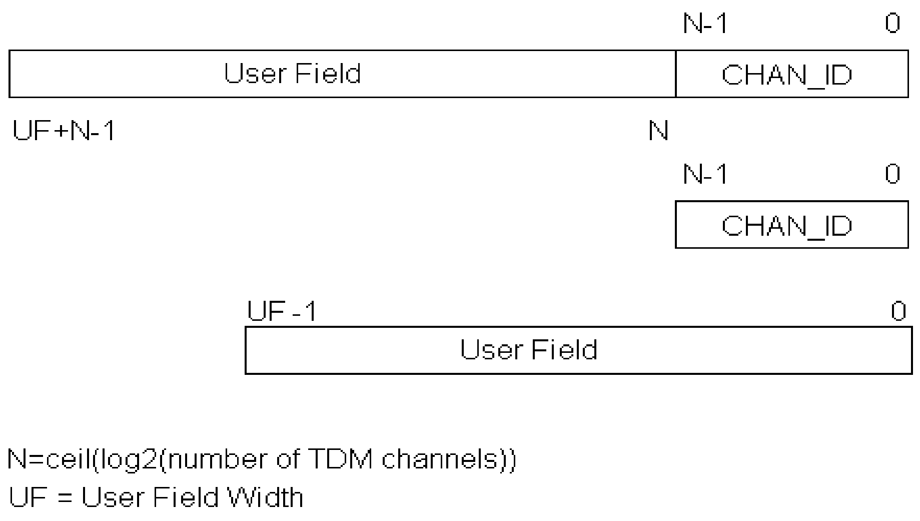

Figure 22: **Input PHASE Channel TUSER Structure**

Input PHASE Channel TLAST Options

The input PHASE channel may be configured to have no TLAST, to have a vector framing TLAST or to have a packet framing TLAST.

When Vector Framing is selected, TLAST is expected to indicate the last channel in a TDM cycle of channels. If TLAST does not match the internal expectation of when TLAST should arrive, one of two event signals will be asserted for a clock cycle.

When Packet Framing is selected, the core does not have any expectation of the timing of TLAST so the event signals are not present, but TLAST will be conveyed to the output channels with the same latency as the TDATA input.

Output DATA channel

The Output Data channel exists whenever the DDS Compiler is configured to have a SIN/COS LUT. This channel replaces the SINE and COSINE outputs of DDS Compiler v4.0. These former outputs now exist as fields of `m_axis_data_tdata`.

Output DATA Channel TDATA Structure

The sine and cosine output fields are sign extended to the next byte boundary then concatenated, with cosine in the least significant portion, to create `m_axis_data_tdata`. If only one of sine or cosine is selected, then it will be sign extended and put into the least significant portion of `m_axis_data_tdata`.

Figure 23 shows the internal structure of TDATA for the three configurations; quadrature outputs, cosine only and sine only. An 11-bit output has been shown in the diagram for example, sign extended to 16 bits. The <<< denotes sign extension.

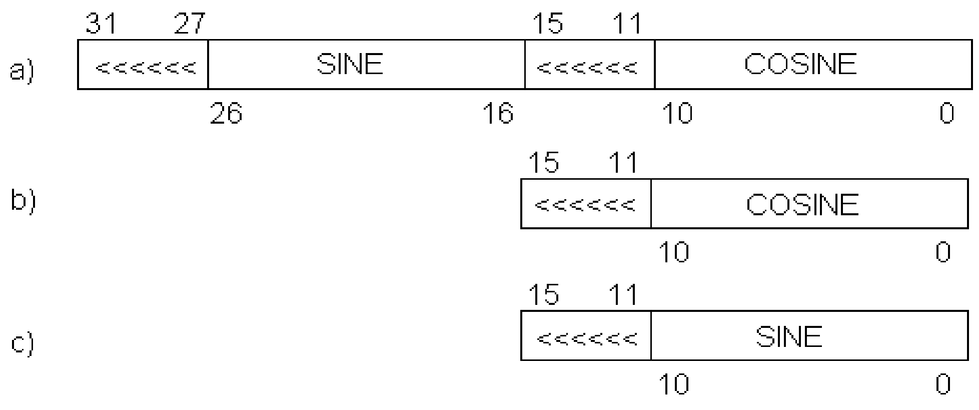

Figure 23: **Output DATA Channel TDATA Structure**

Output DATA Channel TUSER Structure

The output DATA channel can be configured to have no TUSER field, or for TUSER to hold a user field, or a TDM channel index or both a user field and a TDM channel ID. When both user field and TDM channel ID are selected the fields are concatenated with the TDM channel ID in the least significant position. The TDM channel ID qualifies the fields in the TDATA bus for that transfer as belonging to the TDM channel described. See Figure 22 for the structure, as this is identical to the structure for the Output DATA channel TUSER port.

The user field is not used nor interpreted by the core. It is provided as a service to allow the system designer to pass information through the core with latency identical to the main datapath (input PHASE channel to output channels). For instance, the user field could contain flags and other ancillary information irrelevant to the DDS, but relevant to some core downstream from the DDS.

Output DATA Channel TLAST Options

The output DATA channel may be configured to have no TLAST, to have vector framing, to have packet framing or to have a TLAST which is 'Configuration triggered'.

When set to vector framing TLAST will be asserted for the transfer which contains the TDATA of the last TDM channel of a cycle of TDM channels (e.g. channel 12 of 12).

When set to packet framing, the TLAST of the input PHASE channel will be passed unaltered with latency equal to the latency of the main datapath. This is intended to be a service to the system designer, where TLAST may have a meaning which is irrelevant to the DDS, but relevant to some core downstream.

When set to 'Config Triggered' the TLAST will be generated internally by the DDS rather than conveyed from the input PHASE channel. It will be asserted on the last channel of a TDM cycle immediately before a configuration change is effected. In other words, if a configuration change is provoked via the CONFIG channel, TLAST will be asserted on the last sample of the old configuration.

Output PHASE Channel

The output PHASE channel replaces the PHASE_OUT port of DDS Compiler v4.0. The PHASE_OUT port now exists as a field of `m_axis_phase_tdata`.

Output PHASE Channel TDATA Structure

The PHASE_OUT field will be sign extended to the next multiple of 8 bits and become m_axis_phase_tdata. E.g. if PHASE_OUT is 20 bits, m_axis_phase_tdata will be 24 bits wide [23:0], occupied by a sign extended PHASE_OUT.

The diagram below shows this for an example width of 11 bits sign extended to 16 bits.

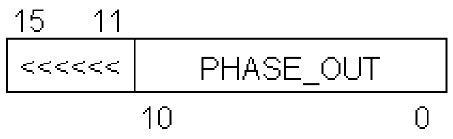

Figure 24: **Output PHASE Channel TDATA Structure**

Output PHASE Channel TUSER Structure

The TUSER field has the same configuration options as the output DATA channel, but the options are independent for the two output channels, so, for instance, one may be configured to have a user field while the other has a TDM channel ID field. See the diagram in Input PHASE Channel TUSER Structure as this is identical to the structure options for the output PHASE Channel TUSER port.

Output PHASE Channel TLAST Options

The output PHASE channel uses the same TLAST setting as the output DATA channel.

Event Interface

In order to allow users to synchronize correctly to the internal channel counter, and flag errors when writing multi-channel data to the CONFIG and/or PHASE channels, the DDS Compiler core provides a number of event outputs to indicate when unexpected conditions have occurred. Event outputs obey `aclken` and `aresetn` conditions, occur immediately and are asserted for each cycle that a discrepancy is present.

If the event outputs are not required, they may be left unconnected and the associated logic will be optimized away by the Xilinx software tools.

CONFIG Channel Event Outputs

For multi-channel configuration, the CONFIG channel expects a single pulse on `s_axis_config_tlast` to indicate the last sample in a sequence of channels (a "vector"). If `s_axis_config_tlast` is asserted when the core does not expect it to be, `event_s_config_tlast_unexpected` will be asserted. If the `s_axis_config_tlast` pulse is not asserted with the last channel's configuration data, `event_s_config_tlast_missing` will be asserted.

PHASE Channel Event Outputs

When the PHASE channel is present with multiple channels, the PHASE channel expects a single pulse on `s_axis_phase_tlast` to indicate the last sample in a sequence of channels (a "vector"). If `s_axis_phase_tlast` is asserted when the core does not expect it to be, `event_s_phase_tlast_unexpected` will be asserted. If the `s_axis_phase_tlast` pulse is not asserted with the final channel's data, `event_s_phase_tlast_missing` will be asserted.

The `s_axis_phase_tuser` configuration options allow the user to input a Channel ID value to facilitate synchronization with the internal channel counter. If this option is selected, the `event_s_phase_chanid_incorrect` output will be present on the core and will be asserted for every cycle where the input Channel ID on `s_axis_phase_tuser` does not match the core's current channel.

Reset Behavior

The active low reset signal, `aresetn`, is registered within the DDS Compiler core to ensure high performance. This signal has the requirement that it must be driven Low for a minimum of two cycles to guarantee there will be no violation of the AXI4-Stream protocol on the core outputs. The registering of reset appears as a delay in entering the reset state, and the signal states after reset differ depending on whether TREADY signals are present.

The following diagram shows the core entering reset when there are no TREADY signals, using a multi channel DDS as an example The subscript on the valid data outputs indicates a channel number. The true behavioral model will output X values for TDATA fields during reset. All channel payloads are qualified with their respective TVALID outputs.

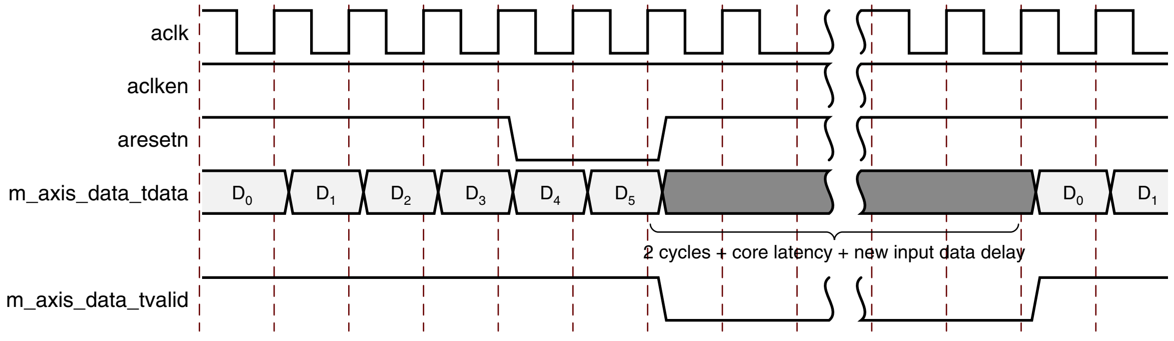

Figure 25: **Reset Behavior (no TREADY)**

The following diagram shows the core entering reset when TREADY is present, using a multi channel DDS as an example. The subscript on the valid data outputs indicates a channel number. The reset event will force the FIFOs within the core back into their initial state; no X value is produced by the VHDL RTL-based behavioral model in this case. Again, all channel payloads are qualified with their respective TVALID outputs.

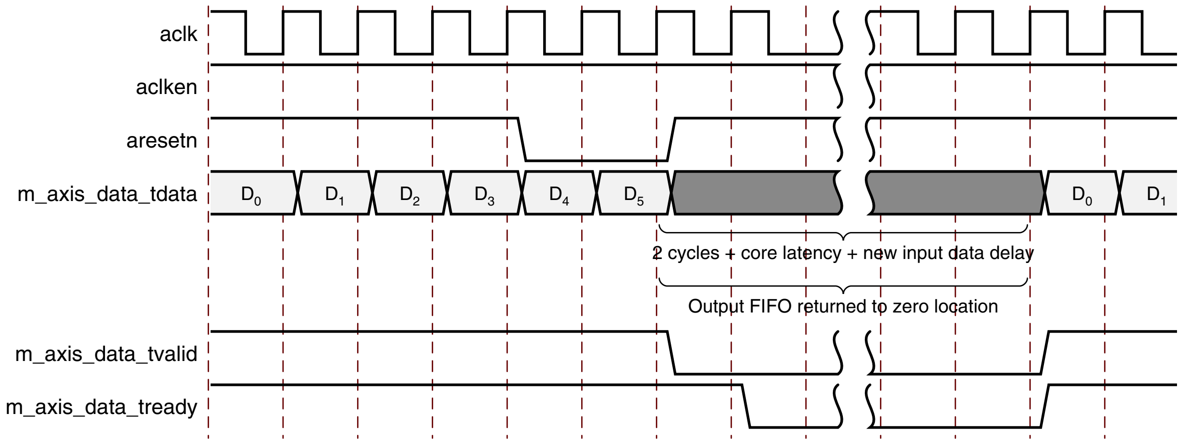

Figure 26: **Reset Behavior (with TREADY)**

Multi-Channel

When configured for more than one channel, the DDS, or Phase Generator part, will generate outputs for each channel in a time-multiplexed fashion. As such, the output for a particular channel will be given every N-cycles, where N is the number of channels selected when the core was customized. The outputs for channel 0 are given first.

Latency

The latency of the core can be specified through the GUI or be automatically set to the optimum value based upon the Optimization Goal.

For streaming inputs (`s_axis_phase_t*`) the latency specifies the minimum number of cycles between input and the associated output.

For configuration inputs, the latency is the minimum latency from the first cycle `aresetn` becomes inactive until a valid output. The latency from configuration channel inputs to output channel outputs is not deterministic due to the fact that configuration inputs will be synchronized to the internal channel counter phase introducing a delay which is unknown external to the core.

Migrating to DDS Compiler v5.0 from Earlier Versions

XCO Parameter Changes

The CORE Generator core update functionality may be used to update an existing XCO file from v4.0 to DDS Compiler v5.0, but it should be noted that the update mechanism alone will not create a core compatible with v4.0. See Instructions for Minimum Change Migration. DDS Compiler v5.0 has additional parameters for AXI4-Stream support. Table 6 shows the changes to XCO parameters from version 4.0 to version 5.0.

Table 6: **XCO Parameter Changes from v4.0 to v5.0**

Version v4.0	Version 5.0	Notes
PartsPresent	PartsPresent	Unchanged
DDS_Clock_Rate	DDS_Clock_Rate	Unchanged
Channels	Channels	Unchanged
Parameter_Entry	Parameter_Entry	Unchanged
Spurious_Free_Dynamic_Range	Spurious_Free_Dynamic_Range	Unchanged
Frequency_Resolution	Frequency_Resolution	Unchanged
Noise_Shaping	Noise_Shaping	Unchanged
Phase_Width	Phase_Width	Unchanged
Output_Width	Output_Width	Unchanged
Phase_Increment	Phase_Increment	Unchanged
Phase_Offset	Phase_Offset	Unchanged
Output_Selection	Output_Selection	Unchanged
Negative_Sine	Negative_Sine	Unchanged
Negative_Cosine	Negative_Cosine	Unchanged
Amplitude_Mode	Amplitude_Mode	Unchanged
Memory_Type	Memory_Type	Unchanged
Optimization_Goal	Optimization_Goal	Unchanged
DSP48_Use	DSP48_Use	Unchanged
Latency_Configuration	Latency_Configuration	Unchanged
Latency	Latency	Unchanged
Has_Phase_Out	Has_Phase_Out	Unchanged
SCLR_pin	Has_ARESETn	Name change only. Note that the signal itself, `aresetn`, is active low and must be driven Low for at least 2 cycles.
Clock_Enable	Has_ACLKEN	Name change only
RFD	-	Deprecated. Closest equivalent is Has_TREADY
RDY	-	Deprecated. Equivalent to AXI4-Stream TVALID, which is not optional
Channel_Pin	-	Deprecated. The presence of a channel indication field is now specified by M_DATA_Has_TUSER and M_PHASE_Has_TUSER
Output_Frequency(1 to 16)	Output_Frequency(1 to 16)	Unchanged

Table 6: **XCO Parameter Changes from v4.0 to v5.0 *(Cont'd)***

Version v4.0	Version 5.0	Notes
PINC(1 to 16)	PINC(1 to 16)	Unchanged
Phase_Offset_Angles(1 to 16)	Phase_Offset_Angles(1 to 16)	Unchanged
POFF(1 to 16)	POFF(1 to 16)	Unchanged
POR_mode	POR_mode	Unchanged
-	DATA_Has_TLAST	New to version 5.0
-	S_PHASE_Has_TUSER	New to version 5.0
-	S_PHASE_TUSER_Width	New to version 5.0
-	Has_TREADY	New to version 5.0
-	M_DATA_Has_TUSER	New to version 5.0
-	M_PHASE_Has_TUSER	New to version 5.0
-	S_CONFIG_Sync_Mode	New to version 5.0

For more information on this upgrade feature, see the CORE Generator software documentation.

Port Changes

Table 7 details the changes to port naming, additional or deprecated ports and polarity changes from v4.0 to v5.0.

Table 7: **Port Changes from Version 4.0 to Version 5.0**

Version 4.0	Version 5.0	Notes
CLK	aclk	Rename only
CE	aclken	Rename only
SCLR	aresetn	Rename, change of sense (now active low), requirement to drive reset Low for a minimum of 2 cycles
ADDR	Deprecated	Replaced by s_axis_config_t* (CONFIG channel)
REG_SELECT		
WE		
DATA		
PINC_IN	Deprecated	Replaced by s_axis_phase_t* (input PHASE channel)
POFF_IN		
PHASE_IN	Deprecated	Replaced by s_axis_phase_t* (Input PHASE channel)
RDY	Deprecated	Nearest equivalent is m_axis_data_tvalid
RFD	Deprecated	Nearest equivalent is s_axis_phase_tready
CHANNEL	Deprecated	Channel ID may be carried as a subfield of m_axis_phase_tuser or m_axis_data_tuser
COSINE	Deprecated	Both these fields are now subfields of m_axis_data_tdata. See Output DATA Channel TDATA Structure.
SINE		
PHASE_OUT(N-1:0)	m_axis_phase_tdata(byte(N)-1: 0)	PHASE_OUT is now the payload of m_axis_phase_tdata
-	s_axis_config_tvalid	TVALID (AXI4-Stream channel handshake signal) for each channel
-	s_axis_phase_tvalid	
-	m_axis_phase_tvalid	
-	m_axis_data_tvalid	
-	s_axis_config_tready	TREADY (AXI4-Stream channel handshake signal) for each channel.
-	s_axis_phase_tready	
-	m_axis_phase_tready	
-	m_axis_data_tready	
-	s_axis_config_tlast	TLAST (AXI4-Stream packet signal indicating the last transfer of a data structure) for each channel. See the TLAST User section for each channel.
-	s_axis_phase_tlast	
-	m_axis_phase_tlast	
-	m_axis_data_tlast	
-	s_axis_phase_tuser(E-1:0)	TUSER (AXI4-Stream ancillary field for application-specific information) for each channel. See the TUSER Packing section for each channel.
-	m_axis_phase_tuser(F-1:0)	
-	m_axis_data_tuser(G-1:0)	
-	event_s_phase_tlast_missing	Asserted on the last transaction of an incoming vector if s_axis_phase_tlast is not seen asserted.

Table 7: **Port Changes from Version 4.0 to Version 5.0 *(Cont'd)***

Version 4.0	Version 5.0	Notes
-	event_s_phase_tlast_unexpected	Asserted on every transaction where s_axis_phase_tlast is unexpectedly seen asserted.
-	event_s_phase_chaind_incorrect	Asserted on every transaction where the s_axis_phase_tuser Channel ID field does not match the value expected by the core.
-	event_s_config_tlast_missing	Asserted on the last transaction of an incoming vector if s_axis_config_tlast is not seen asserted.
-	events_s_config_tlast_unexpected	Asserted on every transaction where s_axis_config_tlast is unexpectedly seen asserted.

Notes:

1. N, E, F and G are all arbitrary independent integers.

Latency Changes

The latency of DDS Compiler v5.0 will be different compared to v4.0 and greater in general. The update process cannot account for this and guarantee equivalent performance.

Importantly, when in Blocking Mode (that is, TREADY handshaking is present), the latency of the core will be variable, so only the minimum possible latency can be determined. When in Non-Blocking Mode (no TREADY), the latency of the core is as shown in the latency field of the GUI and is constant.

Behavioral Changes

DDS Compiler v5.0 remains bit-accurate with respect to DDS Compiler v4.0. However, versions of the DDS Compiler core prior to v5.0 had different latencies for the programmable and streaming interfaces (the number of cycles between applying an input and that input affecting the calculated phase), which resulted in a different output sequence depending on the configuration.

This behavior has been standardized in DDS Compiler v5.0; however, this may not match the behavior of previous versions. The generated phase is described by the following equation:

$$\text{PHASE\_OUT}(n) = \sum_{i=0}^{n} \text{PINC}(n) + \text{POFF}(n)$$

Instructions for Minimum Change Migration

To configure the DDS Compiler v5.0 to most closely mimic the behavior of v4.0 the translation is as follows:

Parameters

Most parameters remain unchanged. Uncheck Output TREADY. All other new parameters will default as required for legacy operation.

- If the CHANNEL output was used, set DATA Output TUSER setting to "Chan_ID_Field"

Ports

Rename ports as described in the table above.

- WE, REG_SELECT, ADDR and DATA are replaced by the CONFIG channel, where WE is equivalent to TVALID, ADDR has no equivalent, but is replaced internally by an incrementing count with `s_axis_config_tlast` denoting the last transfer of the TDM sequence, DATA(T-1:0) is replaced by `s_axis_config_tdata` (See CONFIG Channel TDATA Structure). REG_SELECT is no longer required, since both PINC and POFF may be written in a single transfer.
- PINC_IN and POFF_IN are mapped to `s_axis_phase_tdata` as described in Input PHASE Channel TDATA Structure. Connect `s_axis_phase_tvalid` to logical '1'.
- SINE(P-1:0) and COSINE(P-1:0) are mapped to `m_axis_data_tdata` as described in Output DATA Channel TDATA Structure. RDY becomes `s_axis_data_tvalid`.
- CHANNEL(N-1:0) becomes `m_axis_data_tuser`(N-1:0).
- PHASE_OUT(W-1:0) becomes `m_axis_phase_tdata`(W-1:0) as described in Output PHASE Channel TDATA Structure.

Miscellaneous Changes

The following changes must also be accounted for.

- The synchronous, active low reset pin `aresetn` must be driven Low for a minimum of two clock cycles to correctly reset the core. The reset is registered within the core, causing some delay between the assertion/deassertion of `aresetn` and the effect being seen on the interface.

Performance

To achieve equivalent performance to v4.0 Latency should be set to Latency+1. Alternatively, set Latency to be the same as Latency for v4.0, but there may be drop in performance. Resource allocation for this configuration will be greater than v4.0 in flip-flop count by approximately the number of bits in any fields in `s_axis_phase_tdata`.

Performance and Resource Utilization

Table 8 through 11 show the performance of the DDS Compiler core in terms of resource usage and maximum achieved operating frequency.

All examples are for the DDS, have `aresetn` and `aclken` ports, both sine and cosine outputs present (without negation), full range amplitude and block RAM memory type. All other parameters are specified in the tables. As indicated by the name of the case in the second table for each family, the case aims to provide a given SFDR.

These results were obtained with ISE 12.3 software. The resource count and speed of the core can change depending on the surrounding circuitry of your design. Therefore, these figures should be taken only as a guide.

The maximum clock frequency results were obtained by double-registering input and output ports (using IOB flip-flops) to reduce dependence on I/O placement. The first level of registers used a separate clock signal to measure the path from the input registers to the first output wrapper register through the core.

The resource usage results do not include the aforementioned wrapper registers and represent the true logic used by the core. LUT counts include SRL16s/SRL32s.

The tool settings to achieve these results were as follows:

```
map -ol high -pr b
par -ol high
```

Note: The tool settings can have a significant effect on area use and speed. The Xilinx Xplorer script can be used to find the optimal settings.

The Virtex-6 FPGA test cases in Table 8 used an XC6VLX240T-FF1156C, (-1 speed grade) device and ISE software speed file version "PRODUCTION 1.08 2010-07-27, STEPPING level 0."

Table 8: **Virtex-6 FPGA Performance and Resource Utilization**

Description	Small	Medium	Large A	Large B	Large C	Large D	Large E	Large F	Taylor A	Taylor B
Phase Width	8	23	30	30	30	30	30	30	20	20
Output Width	6	16	17	17	17	17	17	17	18	18
Noise shaping	None	None	Dithering	Dithering	Dithering	Dithering	Dithering	Dithering	Taylor	Taylor
Channels	1	1	1	1	1	16	1	1	1	16
Use_DSP48	Minimal	Minimal	Maximal	Maximal	Maximal	Maximal	Maximal	Maximal	Maximal	Maximal
Optimization	Speed	Area	Speed	Speed	Speed	Speed	Speed	Speed	Speed	Speed
Latency	2	2	Auto	Auto	Auto	Auto	Auto	Auto	Auto	Auto
Phase Increment	Prog	Prog	Fixed	Prog	Prog	Prog	Stream	Stream	Prog	Prog
Phase Offset	None	None	Fixed	Fixed	Fixed	Fixed	Fixed	Fixed	None	None
Phase Angle Width	8	12	15	15	15	15	15	15	11	11
Has TREADY	No	No	No	No	Yes	Yes	Yes	Yes	No	No
USER Field	No	No	No	No	No	No	No	Yes / 16 bits	No	No
LUT FF pairs	20	102	131	159	216	265	237	323	106	162
LUTs	18	100	81	131	158	207	170	216	83	116
FFs	19	50	169	227	268	292	309	389	163	196
Block RAM36/18s	0/1	0/1	4/0	4/0	4/0	4/0	4/0	4/0	0/1	0/1
DSP48E1s	0	0	2	2	2	2	2	2	4	4
Frequency (MHz)	342	270	445	445	445	445	445	438	324	445

Table 9: **Virtex-6 FPGA Performance and Resource Utilization**

Description	SFDR70		SFDR84		SFDR110		SFDR140	
Phase Width	12		12		12		25	
Output Width	12		14		20		25	
Noise Shaping	None		Dithering		Taylor		Taylor	
Channels	1		1		1		1	
Optimization Goal	Speed		Speed		Speed		Speed	
Latency	Auto		Auto		Auto		Auto	
Phase Increment	Programmable		Programmable		Programmable		Programmable	
Phase Offset	None		None		None		None	
Phase Angle Width	12		12		11		11	
Has TREADY	No		No		No		No	
USER Field	No		No		No		No	
Use DSP48	Minimal	Maximal	Minimal	Maximal	Minimal	Maximal	Minimal	Maximal

Table 9: **Virtex-6 FPGA Performance and Resource Utilization *(Cont'd)***

Description	SFDR70		SFDR84		SFDR110		SFDR140	
LUT FF pairs	111	111	134	121	142	113	230	173
LUTs	70	61	101	80	116	84	183	118
FFs	127	129	164	154	191	171	313	259
Block RAM36/18s	0/1	0/1	0/1	0/1	1/0	1/0	1/1	1/1
DSP48E1s	0	1	0	2	3	4	5	6
Frequency (MHz)	445	445	445	445	445	445	395	395

The Spartan-6 FPGA test cases in Table 10 used an XC6SLX45-FGG484 (-2 speed grade) device and ISE software speed file version "PRODUCTION 1.11 2010-07-27."

Table 10: **Spartan-6 FPGA Performance and Resource Utilization**

Description	Small	Medium	Large A	Large B	Large C	Large D	Large E	Large F	Taylor A	Taylor B
Phase Width	8	23	30	30	30	30	30	30	20	20
Output Width	6	16	17	17	17	17	17	17	18	18
Noise shaping	None	None	Dithering	Dithering	Dithering	Dithering	Dithering	Dithering	Taylor	Taylor
Channels	1	1	1	1	1	16	1	1	1	16
Use_DSP48	Minimal	Minimal	Maximal	Maximal	Maximal	Maximal	Maximal	Maximal	Maximal	Maximal
Optimization	Speed	Area	Speed	Speed	Speed	Speed	Speed	Speed	Speed	Speed
Latency	2	2	Auto	Auto	Auto	Auto	Auto	Auto	Auto	Auto
Phase Increment	Prog	Prog	Fixed	Prog	Prog	Prog	Stream	Stream	Prog	Prog
Phase Offset	None	None	Fixed	Fixed	Fixed	Fixed	Fixed	Fixed	None	None
Phase Angle Width	8	12	15	15	15	15	15	15	11	11
Has TREADY	No	No	No	No	Yes	Yes	Yes	Yes	No	No
USER Field	No	No	No	No	No	No	No	Yes / 16 bits	No	No
LUT FF pairs	16	94	133	188	246	311	272	303	120	303
LUTs	13	77	114	132	206	298	220	271	66	135
FFs	19	50	172	229	268	308	309	389	163	196
Block RAM18/9s	0/1	1/0	8/0	8/0	8/0	8/0	8/0	8/0	1/0	1/0
DSP48A1s	0	0	3	3	3	3	3	3	4	4
Frequency (MHz)	213	176	256	256	235	214	235	239	256	249

Table 11: **Spartan-6 FPGA Performance and Resource Utilization**

Description	SFDR70		SFDR84		SFDR110		SFDR140	
Phase Width	12		12		12		25	
Output Width	12		14		20		25	
Noise Shaping	None		Dithering		Taylor		Taylor	
Channels	1		1		1		1	
Optimization Goal	Speed		Speed		Speed		Speed	
Latency	Auto		Auto		Auto		Auto	
Phase Increment	Programmable		Programmable		Programmable		Programmable	
Phase Offset	None		None		None		None	
Phase Angle Width	12		12		11		11	
Has TREADY	No		No		No		No	
USER Field	No		No		No		No	
Use DSP48	Minimal	Maximal	Minimal	Maximal	Minimal	Maximal	Minimal	Maximal
LUT FF pairs	98	101	122	117	139	133	252	195
LUTs	63	67	96	84	98	68	166	110
FFs	127	131	164	154	191	171	331	277
Block RAM18/9s	1/0	1/0	1/0	1/0	1/1	1/1	2/1	2/1
DSP48A1s	0	1	0	2	3	4	5	6
Frequency (MHz)	256	256	256	256	256	256	165	184

Known Issues

Sub-Harmonic Frequencies

The equations for SFDR rely on the assumption that rounding errors from the finite precision of phase and amplitude are incoherent. This assumption is violated for values of Phase Increment that are not mutually prime with the weight of the Phase Accumulator. The anomalies, such as spurii, will be more obvious for larger common factors between the Phase Increment Value and the weight of the accumulator (2^{Phase_Width}). This is because such values may not access every location in the SIN/COS LUT, so the rounding errors are not randomly spread. To avoid this, do not use values of Output Frequency that are simple rational fractions of the frequency per channel, Fs, such as 3/8, 1/64.

Design Examples

The DDS Compiler GUI accepts system-level parameters instead of low-level parameters such as the width of the phase accumulator, width of the phase angle, etc. Because of this, all preceding requirements can be entered into the GUI directly without having to calculate low-level core details. The GUI also provides feedback of the hardware parameters so the translation of system-level parameters to low-level parameters can been seen. Alternatively, hardware parameters may be entered directly.

Example 1

For a single-channel DDS with 1 MHz system clock, frequency resolution of 1 Hz, Phase Width is 20-bits. To synthesize an output of 23.4 kHz, an Output Frequency value of 0.0234 MHz must be entered into the GUI, which then returns a value of 5FD8 in hexadecimal, which is 24536 in decimal.

This will give a synthesized frequency of $24536/2^{20}$ *1 MHz = 23399.35 Hz.

If the application requires this to be modulated by one of 8 phase offsets, the phase offset bus need only be 3-bits precision, but these must be the top 3 bits of the phase offset input. Hence, the phase offset of 1/8 of a cycle would be entered as 0.125 in the GUI. This returns a value of 20000(hex). This could be entered on the 3-bit bus as 001(binary).

Example 2 (DDS Requiring Negative Frequencies)

For a 4-channel DDS with 100 MHz System Clock, Frequency Resolution of 1 Hz the Phase Width is 25-bits. Frequencies of -3 MHz, -1 MHz, 1 MHz and 3 MHz are required. Fs is the frequency per channel which is System Clock/Number of Channels, that is, 25 MHz. The negative frequencies alias to every Fs Hz. The legal range to enter in the GUI is 0 to Fs, so the entered frequencies for this example must be 22 MHz (Fs-3 MHz), 24 MHz (Fs-1 MHz), 1 MHz and 3 MHz respectively.

Support

Xilinx provides technical support for this LogiCORE product when used as described in the product documentation. Xilinx cannot guarantee timing, functionality, or support of product if implemented in devices that are not defined in the documentation, if customized beyond that allowed in the product documentation, or if changes are made to any section of the design labeled *DO NOT MODIFY*.

Refer to the IP Release Notes Guide (XTP025) for further information on this core. There is a link to all the DSP IP and then to each core. For each core, there is a master Answer Record that contains the Release Notes and Known Issues list for each core. The following information is listed for each version of the core:

- New Features
- Bug Fixes
- Known Issues

Ordering Information

This LogiCORE IP module is included at no additional cost with the Xilinx ISE Design Suite software and is provided under the terms of the Xilinx End User License Agreement. Use the CORE Generator software included with the ISE Design Suite to generate the core. For more information, please visit the core page.

Information about additional Xilinx LogiCORE IP modules is available at the Xilinx IP Center. For pricing and availability of other Xilinx LogiCORE IP modules and software, please contact your local Xilinx sales representative.

References

1. *Xilinx AXI Design Reference Guide* (UG761)
2. AMBA 4 AXI4-Stream Protocol Version: 1.0 Specification
3. IP Release Notes Guide

List of Acronyms

Acronym	Spelled Out
AMBA	Advanced Microcontroller Bus Architecture
AXI	Advanced eXtensible Interface
dB	decibel
DDS	Direct Digital Synthesizer
DSP	Digital Signal Processing
FF	Flip-Flop
FPGA	Field Programmable Gate Array
GUI	Graphical User Interface
Hz	Hertz
I/O	Input/Output
IP	Intellectual Property
ISE	Integrated Software Environment
LUT	Lookup Table
MHz	Mega Hertz
MSK	Minimum Shift Keying
NCO	Numerically Controlled Oscillator
PCS	Physical Coding Sublayer
PSK	Phase Shift Keying
RAM	Random Access Memory
ROM	Read Only Memory
RTL	Register Transfer Level
SDR	Software-defined radios
SFDR	Spurious Free Dynamic Range
VHDL	VHSIC Hardware Description Language (VHSIC an acronym for Very High-Speed Integrated Circuits)
XCO	Xilinx CORE Generator core source file
XST	Xilinx Synthesis Technology

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
09/21/10	1.0	First release of the core with AXI interface support. The previous release of this document was ds558.
03/01/11	1.1	Support added for Virtex-7 and Kintex-7. ISE Design Suite 13.1

Notice of Disclaimer