NCO IP Core

User Guide

Updated for Intel® Quartus® Prime Design Suite: **17.1**

Online Version
Send Feedback

UG-NCO

ID: 683406
Version: 2017.11.06

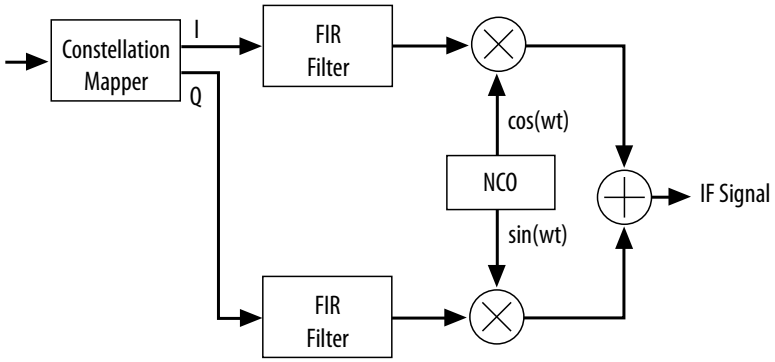
Contents

1. About the NCO IP Core

The Altera® NCO IP core generates numerically controlled oscillators (NCOs) customized for Intel devices. A numerically controlled oscillator (NCO) synthesizes a discrete-time, discrete-valued representation of a sinusoidal waveform.

Typically, you can use NCOs in communication systems as quadrature carrier generators in I-Q mixers, in which baseband data is modulated onto the orthogonal carriers in one of a variety of ways.

Figure 1. Simple Modulator



You can also use NCOs in all-digital phase-locked-loops (PLLs) for carrier synchronization in communications receivers, or as standalone frequency shift keying (FSK) or phase shift keying (PSK) modulators. In these applications, the phase or the frequency of the output waveform varies directly according to an input data stream.

You can implement ROM-based, CORDIC-based, and multiplier-based NCO architectures,. The wizard also includes time and frequency domain graphs that dynamically display the functionality of the NCO, based on your parameter settings.

To decide which NCO implementation to use, consider the spectral purity, frequency resolution, performance, throughput, and required device resources. Also, consider the trade-offs between some or all of these parameters.

1.1. Intel® DSP IP Core Features

- Avalon® Streaming (Avalon-ST) interfaces
- DSP Builder for Intel® FPGAs ready
- Testbenches to verify the IP core
- IP functional simulation models for use in Intel-supported VHDL and Verilog HDL simulators

1.2. NCO IP Core Features

- 32-bit precision for angle and magnitude
- Source interface compatible with the *Avalon Interface Specification*
- Multiple NCO architectures:
  - Multiplier-based implementation using DSP blocks or logic elements (LEs), (single cycle and multi-cycle)
  - Parallel or serial CORDIC-based implementation
  - ROM-based implementation using embedded array blocks (EABs), embedded system blocks (ESBs), or external ROM
- Single or dual outputs (sine/cosine)
- Variable width frequency modulation input
- Variable width phase modulation input
- User-defined frequency resolution, angular precision, and magnitude precision
- Frequency hopping
- Multichannel capability
- Simulation files and architecture-specific testbenches for VHDL and Verilog HDL
- Dual-output oscillator and quaternary frequency shift keying (QFSK) modulator example designs

1.3. DSP IP Core Device Family Support

Intel offers the following device support levels for Intel FPGA IP cores:

- Advance support—the IP core is available for simulation and compilation for this device family. FPGA programming file (`.pof`) support is not available for Quartus Prime Pro Stratix 10 Edition Beta software and as such IP timing closure cannot be guaranteed. Timing models include initial engineering estimates of delays based on early post-layout information. The timing models are subject to change as silicon testing improves the correlation between the actual silicon and the timing models. You can use this IP core for system architecture and resource utilization studies, simulation, pinout, system latency assessments, basic timing assessments (pipeline budgeting), and I/O transfer strategy (data-path width, burst depth, I/O standards tradeoffs).
- Preliminary support—Intel verifies the IP core with preliminary timing models for this device family. The IP core meets all functional requirements, but might still be undergoing timing analysis for the device family. You can use it in production designs with caution.
- Final support—Intel verifies the IP core with final timing models for this device family. The IP core meets all functional and timing requirements for the device family. You can use it in production designs.

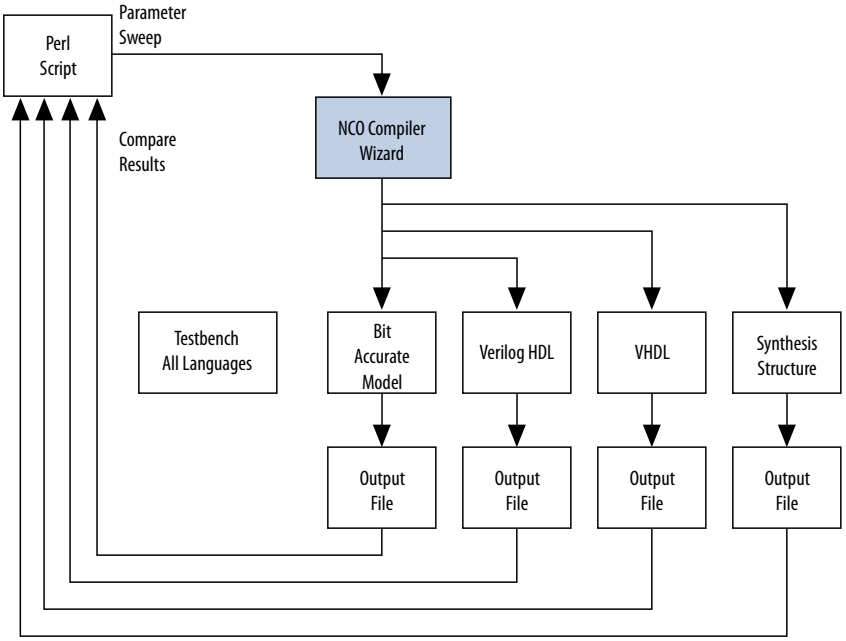
Table 1. DSP IP Core Device Family Support

Device Family	Support
Arria® II GX	Final
Arria II GZ	Final
	continued...

Device Family	Support
Arria V	Final
Intel Arria 10	Final
Cyclone® IV	Final
Cyclone V	Final
Intel Cyclone 10	Final
Intel MAX® 10 FPGA	Final
Stratix® IV GT	Final
Stratix IV GX/E	Final
Stratix V	Final
Intel Stratix 10	Advance
Other device families	No support

1.4. NCO IP Core MegaCore Verification

Figure 2. Regression Flow



1.5. NCO IP Core Release Information

Table 2. NCO IP Core Release Information

Item	Description
Version	17.1
Release Date	November 2017
Ordering Code	IP-NCO

Intel verifies that the current version of the Quartus Prime software compiles the previous version of each IP core. Intel does not verify that the Quartus Prime software compiles IP core versions older than the previous version. The *Intel FPGA IP Release Notes* lists any exceptions.

Related Information

- Intel FPGA IP Release Notes
- Errata for NCO IP core in the Knowledge Base

1.6. NCO IP Core Performance and Resource Utilization

Table 3. NCO IP Core Performance

Typical performance using the Quartus II software with the Arria V (5AGXFB3H4F40C4), Cyclone V (5CGXFC7D6F31C6), and Stratix V (5SGSMD4H2F35C2) devices

Device	Parameters	ALM	DSP Blocks	Memory		Registers		f_{MAX} (MHz)
				M10K	M20K	Primary	Secondary	
Arria V	Cordic	838	0	1	--	1,879	8	340
Arria V	Large Rom	56	0	12	--	149	0	350
Arria V	Multiplier Based	92	2	2	--	244	2	310
Arria V	Small ROM	132	0	6	--	300	0	350
Cyclone V	Cordic	838	0	1	--	1,881	6	260
Cyclone V	Large Rom	56	0	12	--	149	0	275
Cyclone V	Multiplier Based	92	2	2	--	244	2	275
Cyclone V	Small ROM	120	0	6	--	300	0	275
Stratix V	Cordic	838	0	--	1	1,881	6	644
Stratix V	Large Rom	56	0	--	5	149	0	700
Stratix V	Multiplier Based	92	2	--	2	245	1	500
Stratix V	Small ROM	126	0	--	3	300	0	700

2. NCO IP Core Getting Started

1.

2.1. Installing and Licensing Intel FPGA IP Cores

The Intel Quartus® Prime software installation includes the Intel FPGA IP library. This library provides many useful IP cores for your production use without the need for an additional license. Some Intel FPGA IP cores require purchase of a separate license for production use. The Intel FPGA IP Evaluation Mode allows you to evaluate these licensed Intel FPGA IP cores in simulation and hardware, before deciding to purchase a full production IP core license. You only need to purchase a full production license for licensed Intel IP cores after you complete hardware testing and are ready to use the IP in production.

The Intel Quartus Prime software installs IP cores in the following locations by default:

Figure 3. IP Core Installation Path

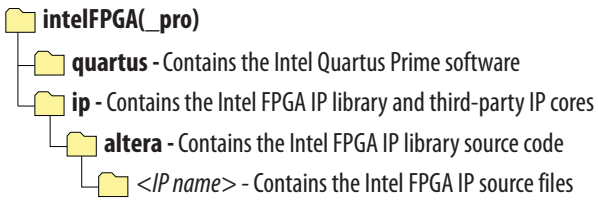


Table 4. IP Core Installation Locations

Location	Software	Platform
<drive>:\intelFPGA_pro\quartus\ip\altera	Intel Quartus Prime Pro Edition	Windows*
<drive>:\intelFPGA\quartus\ip\altera	Intel Quartus Prime Standard Edition	Windows
<home directory>:/intelFPGA_pro/quartus/ip/altera	Intel Quartus Prime Pro Edition	Linux*
<home directory>:/intelFPGA/quartus/ip/altera	Intel Quartus Prime Standard Edition	Linux

Note: The Intel Quartus Prime software does not support spaces in the installation path.

2.1.1. Intel FPGA IP Evaluation Mode

The free Intel FPGA IP Evaluation Mode allows you to evaluate licensed Intel FPGA IP cores in simulation and hardware before purchase. Intel FPGA IP Evaluation Mode supports the following evaluations without additional license:

- Simulate the behavior of a licensed Intel FPGA IP core in your system.
- Verify the functionality, size, and speed of the IP core quickly and easily.
- Generate time-limited device programming files for designs that include IP cores.
- Program a device with your IP core and verify your design in hardware.

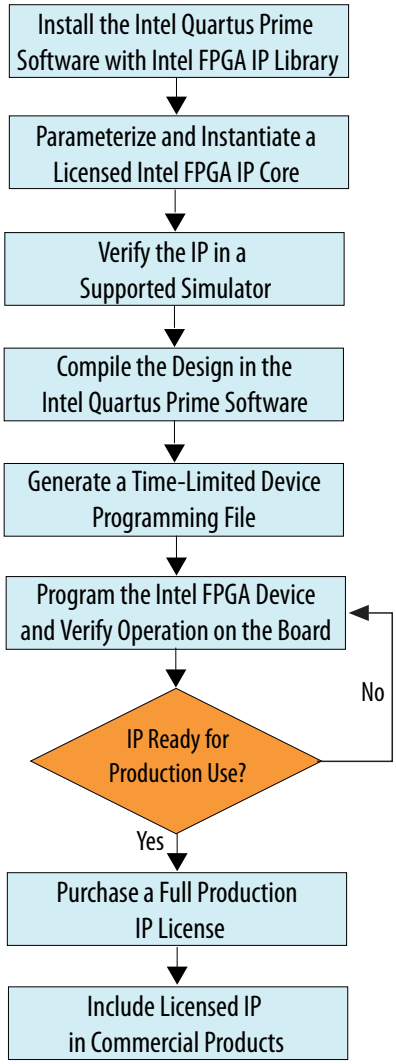
Intel FPGA IP Evaluation Mode supports the following operation modes:

- **Tethered**—Allows running the design containing the licensed Intel FPGA IP indefinitely with a connection between your board and the host computer. Tethered mode requires a serial joint test action group (JTAG) cable connected between the JTAG port on your board and the host computer, which is running the Intel Quartus Prime Programmer for the duration of the hardware evaluation period. The Programmer only requires a minimum installation of the Intel Quartus Prime software, and requires no Intel Quartus Prime license. The host computer controls the evaluation time by sending a periodic signal to the device via the JTAG port. If all licensed IP cores in the design support tethered mode, the evaluation time runs until any IP core evaluation expires. If all of the IP cores support unlimited evaluation time, the device does not time-out.
- **Untethered**—Allows running the design containing the licensed IP for a limited time. The IP core reverts to untethered mode if the device disconnects from the host computer running the Intel Quartus Prime software. The IP core also reverts to untethered mode if any other licensed IP core in the design does not support tethered mode.

When the evaluation time expires for any licensed Intel FPGA IP in the design, the design stops functioning. All IP cores that use the Intel FPGA IP Evaluation Mode time out simultaneously when any IP core in the design times out. When the evaluation time expires, you must reprogram the FPGA device before continuing hardware verification. To extend use of the IP core for production, purchase a full production license for the IP core.

You must purchase the license and generate a full production license key before you can generate an unrestricted device programming file. During Intel FPGA IP Evaluation Mode, the Compiler only generates a time-limited device programming file (<*project name*>`_time_limited.sof`) that expires at the time limit.

Figure 4. Intel FPGA IP Evaluation Mode Flow



Note: Refer to each IP core's user guide for parameterization steps and implementation details.

Intel licenses IP cores on a per-seat, perpetual basis. The license fee includes first-year maintenance and support. You must renew the maintenance contract to receive updates, bug fixes, and technical support beyond the first year. You must purchase a full production license for Intel FPGA IP cores that require a production license, before generating programming files that you may use for an unlimited time. During Intel FPGA IP Evaluation Mode, the Compiler only generates a time-limited device programming file (`<project name>_time_limited.sof`) that expires at the time limit. To obtain your production license keys, visit the Self-Service Licensing Center.

The Intel FPGA Software License Agreements govern the installation and use of licensed IP cores, the Intel Quartus Prime design software, and all unlicensed IP cores.

Related Information

- Intel Quartus Prime Licensing Site
- Introduction to Intel FPGA Software Installation and Licensing

2.1.2. NCO IP Core Intel FPGA IP Evaluation Mode Timeout Behavior

All IP cores in a device time out simultaneously when the most restrictive evaluation time is reached. If a design has more than one IP core, the time-out behavior of the other IP cores may mask the time-out behavior of a specific IP core .

For IP cores, the untethered time-out is 1 hour; the tethered time-out value is indefinite. Your design stops working after the hardware evaluation time expires. The Quartus II software uses Intel FPGA IP Evaluation Mode Files (`.ocp`) in your project directory to identify your use of the Intel FPGA IP Evaluation Mode evaluation program. After you activate the feature, do not delete these files..

When the evaluation time expires, the output of NCO IP core goes low.

Related Information

AN 320: OpenCore Plus Evaluation of Megafunctions

2.2. IP Catalog and Parameter Editor

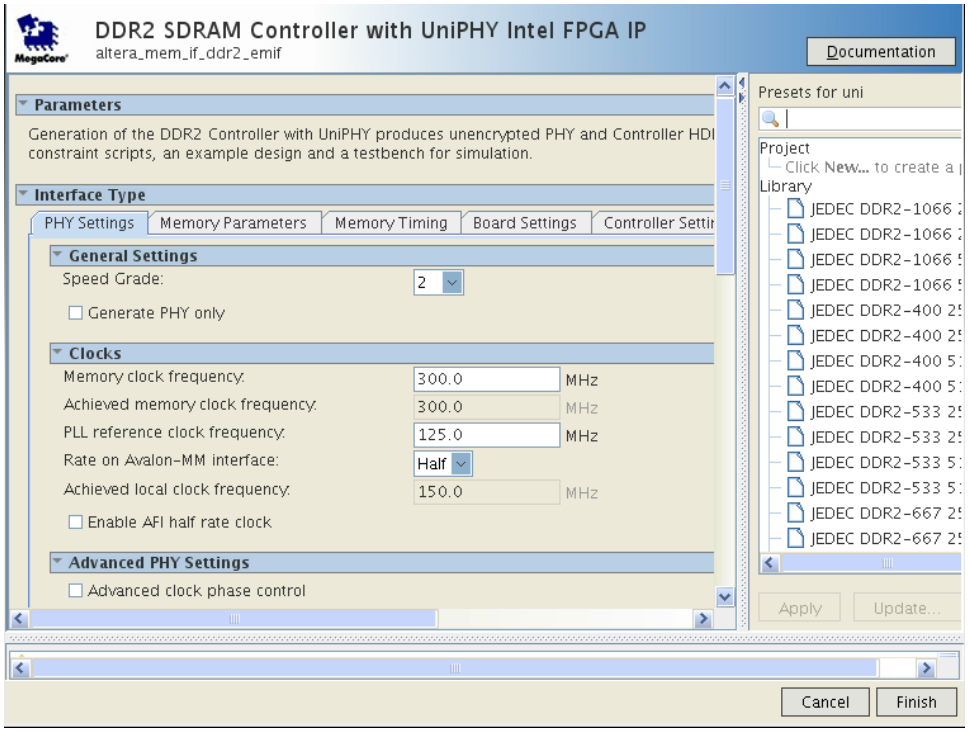
The IP Catalog displays the IP cores available for your project, including Intel FPGA IP and other IP that you add to the IP Catalog search path.. Use the following features of the IP Catalog to locate and customize an IP core:

- Filter IP Catalog to **Show IP for active device family** or **Show IP for all device families**. If you have no project open, select the **Device Family** in IP Catalog.
- Type in the Search field to locate any full or partial IP core name in IP Catalog.
- Right-click an IP core name in IP Catalog to display details about supported devices, to open the IP core's installation folder, and for links to IP documentation.
- Click **Search for Partner IP** to access partner IP information on the web.

The parameter editor prompts you to specify an IP variation name, optional ports, and output file generation options. The parameter editor generates a top-level Intel Quartus Prime IP file (`.ip`) for an IP variation in Intel Quartus Prime Pro Edition projects.

The parameter editor generates a top-level Quartus IP file (`.qip`) for an IP variation in Intel Quartus Prime Standard Edition projects. These files represent the IP variation in the project, and store parameterization information.

Figure 5. IP Parameter Editor (Intel Quartus Prime Standard Edition)



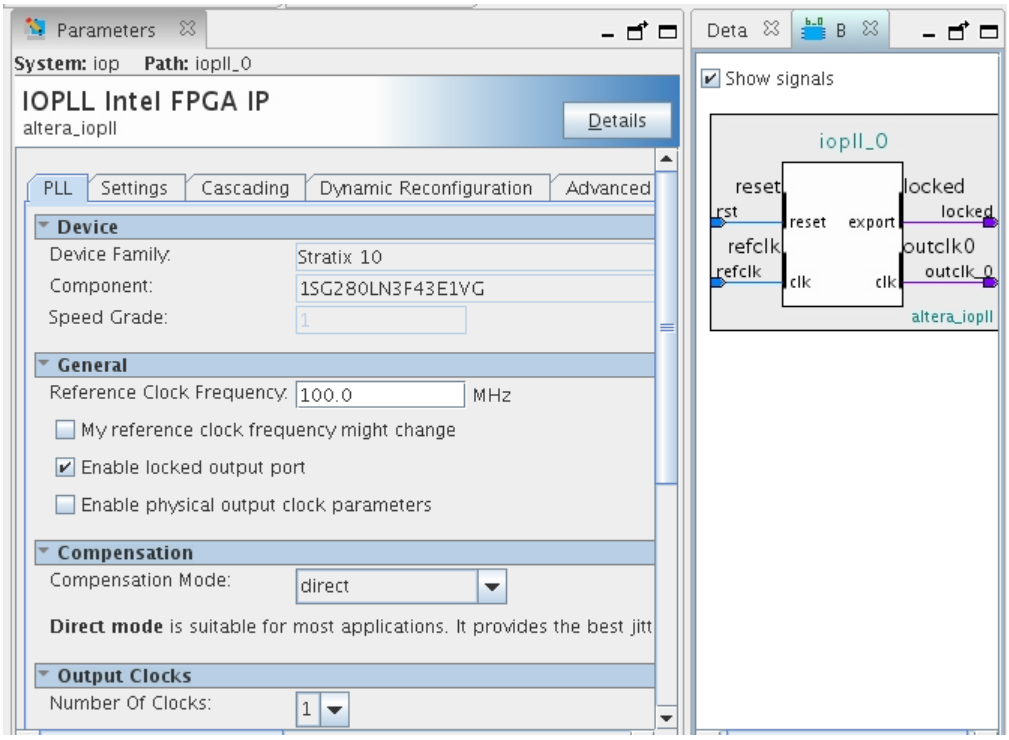
2.3. Generating IP Cores (Intel Quartus Prime Pro Edition)

Quickly configure Intel FPGA IP cores in the Intel Quartus Prime parameter editor. Double-click any component in the IP Catalog to launch the parameter editor. The parameter editor allows you to define a custom variation of the IP core. The parameter editor generates the IP variation synthesis and optional simulation files, and adds the `.ip` file representing the variation to your project automatically.

Follow these steps to locate, instantiate, and customize an IP core in the parameter editor:

1. Create or open an Intel Quartus Prime project (`.qpf`) to contain the instantiated IP variation.
2. In the IP Catalog (**Tools ➤ IP Catalog**), locate and double-click the name of the IP core to customize. To locate a specific component, type some or all of the component's name in the IP Catalog search box. The New IP Variation window appears.
3. Specify a top-level name for your custom IP variation. Do not include spaces in IP variation names or paths. The parameter editor saves the IP variation settings in a file named <*your_ip*>`.ip`. Click **OK**. The parameter editor appears.

Figure 6. IP Parameter Editor (Intel Quartus Prime Pro Edition)



4. Set the parameter values in the parameter editor and view the block diagram for the component. The **Parameterization Messages** tab at the bottom displays any errors in IP parameters:
   - Optionally, select preset parameter values if provided for your IP core. Presets specify initial parameter values for specific applications.
   - Specify parameters defining the IP core functionality, port configurations, and device-specific features.
   - Specify options for processing the IP core files in other EDA tools.

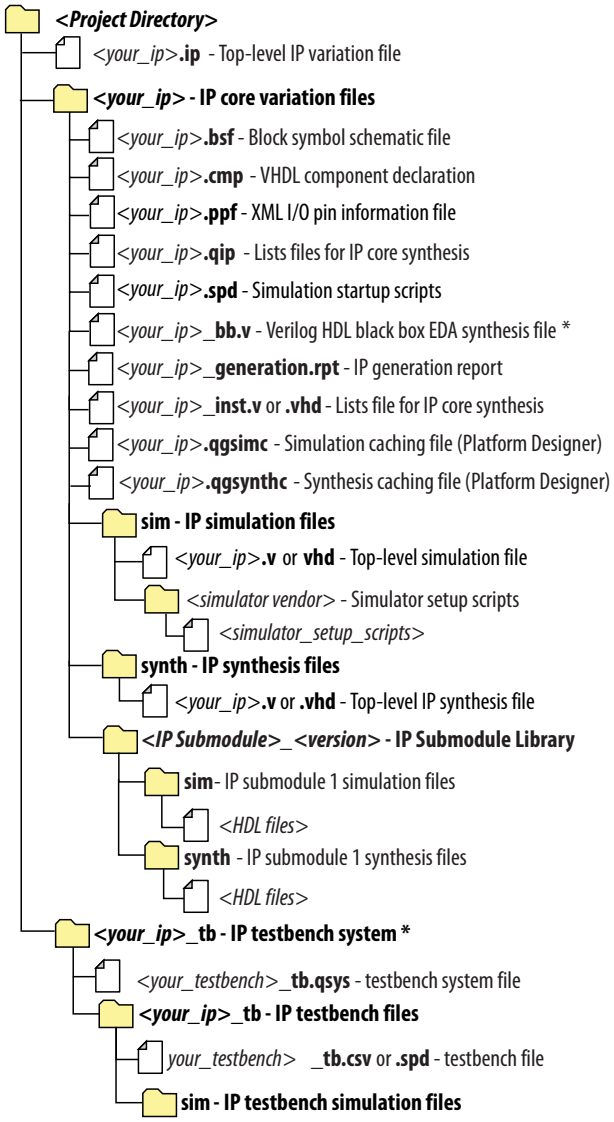
   Note: Refer to your IP core user guide for information about specific IP core parameters.
5. Click **Generate HDL**. The **Generation** dialog box appears.
6. Specify output file generation options, and then click **Generate**. The synthesis and simulation files generate according to your specifications.
7. To generate a simulation testbench, click **Generate ➤ Generate Testbench System**. Specify testbench generation options, and then click **Generate**.
8. To generate an HDL instantiation template that you can copy and paste into your text editor, click **Generate ➤ Show Instantiation Template**.
9. Click **Finish**. Click **Yes** if prompted to add files representing the IP variation to your project.
10. After generating and instantiating your IP variation, make appropriate pin assignments to connect ports.

Note: Some IP cores generate different HDL implementations according to the IP core parameters. The underlying RTL of these IP cores contains a unique hash code that prevents module name collisions between different variations of the IP core. This unique code remains consistent, given the same IP settings and software version during IP generation. This unique code can change if you edit the IP core's parameters or upgrade the IP core version. To avoid dependency on these unique codes in your simulation environment, refer to *Generating a Combined Simulator Setup Script*.

2.3.1. IP Core Generation Output (Intel Quartus Prime Pro Edition)

The Intel Quartus Prime software generates the following output file structure for individual IP cores that are not part of a Platform Designer system.

Figure 7. Individual IP Core Generation Output (Intel Quartus Prime Pro Edition)



* If supported and enabled for your IP core variation.

Table 5. Output Files of Intel FPGA IP Generation

File Name	Description
<*your_ip*>.ip	Top-level IP variation file that contains the parameterization of an IP core in your project. If the IP variation is part of a Platform Designer system, the parameter editor also generates a .qsys file.
<*your_ip*>.cmp	The VHDL Component Declaration (.cmp) file is a text file that contains local generic and port definitions that you use in VHDL design files.
<*your_ip*>_generation.rpt	IP or Platform Designer generation log file. Displays a summary of the messages during IP generation.
<*your_ip*>.qgsimc (Platform Designer systems only)	Simulation caching file that compares the .qsys and .ip files with the current parameterization of the Platform Designer system and IP core. This comparison determines if Platform Designer can skip regeneration of the HDL.
<*your_ip*>.qgsynth (Platform Designer systems only)	Synthesis caching file that compares the .qsys and .ip files with the current parameterization of the Platform Designer system and IP core. This comparison determines if Platform Designer can skip regeneration of the HDL.
<*your_ip*>.qip	Contains all information to integrate and compile the IP component.
<*your_ip*>.csv	Contains information about the upgrade status of the IP component.
<*your_ip*>.bsf	A symbol representation of the IP variation for use in Block Diagram Files (.bdf).
<*your_ip*>.spd	Input file that ip-make-simscript requires to generate simulation scripts. The .spd file contains a list of files you generate for simulation, along with information about memories that you initialize.
<*your_ip*>.ppf	The Pin Planner File (.ppf) stores the port and node assignments for IP components you create for use with the Pin Planner.
<*your_ip*>_bb.v	Use the Verilog blackbox (_bb.v) file as an empty module declaration for use as a blackbox.
<*your_ip*>_inst.v or _inst.vhd	HDL example instantiation template. Copy and paste the contents of this file into your HDL file to instantiate the IP variation.
<*your_ip*>.regmap	If the IP contains register information, the Intel Quartus Prime software generates the .regmap file. The .regmap file describes the register map information of master and slave interfaces. This file complements the .sopcinfo file by providing more detailed register information about the system. This file enables register display views and user customizable statistics in System Console.
<*your_ip*>.svd	Allows HPS System Debug tools to view the register maps of peripherals that connect to HPS within a Platform Designer system. During synthesis, the Intel Quartus Prime software stores the .svd files for slave interface visible to the System Console masters in the .sof file in the debug session. System Console reads this section, which Platform Designer queries for register map information. For system slaves, Platform Designer accesses the registers by name.
<*your_ip*>.v <*your_ip*>.vhd	HDL files that instantiate each submodule or child IP core for synthesis or simulation.
mentor/	Contains a msim_setup.tcl script to set up and run a ModelSim* simulation.
aldec/	Contains a Riviera-PRO* script rivierapro_setup.tcl to setup and run a simulation.
/synopsys/vcs /synopsys/vcsmx	Contains a shell script vcs_setup.sh to set up and run a VCS* simulation. Contains a shell script vcsmx_setup.sh and synopsys_sim.setup file to set up and run a VCS MX simulation.

continued...

File Name	Description
/cadence	Contains a shell script ncsim_setup.sh and other setup files to set up and run an NCSim simulation.
/xcelium	Contains an Xcelium* Parallel simulator shell script xcelium_setup.sh and other setup files to set up and run a simulation.
/submodules	Contains HDL files for the IP core submodule.
<*IP submodule*>/	Platform Designer generates /synth and /sim sub-directories for each IP submodule directory that Platform Designer generates.

2.4. Simulating Intel FPGA IP Cores

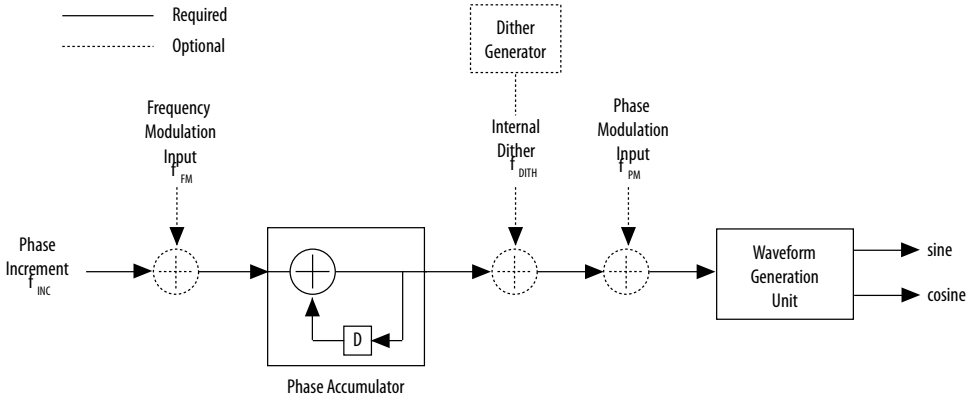
The Intel Quartus Prime software supports IP core RTL simulation in specific EDA simulators. IP generation creates simulation files, including the functional simulation model, any testbench (or example design), and vendor-specific simulator setup scripts for each IP core. Use the functional simulation model and any testbench or example design for simulation. IP generation output may also include scripts to compile and run any testbench. The scripts list all models or libraries you require to simulate your IP core.

The Intel Quartus Prime software provides integration with many simulators and supports multiple simulation flows, including your own scripted and custom simulation flows. Whichever flow you choose, IP core simulation involves the following steps:

1. Generate simulation model, testbench (or example design), and simulator setup script files.
2. Set up your simulator environment and any simulation scripts.
3. Compile simulation model libraries.
4. Run your simulator.

3. NCO IP Core Functional Description

Figure 8. NCO Block Diagram



The NCO IP core allows you to generate a variety of NCO architectures. Your custom NCO includes both time- and frequency-domain analysis tools. The custom NCO outputs a sinusoidal waveform in two's complement representation.

The waveform for the generated sine wave is defined by the following equation:

$s(nT) = A \sin[2\pi(f_O + f_{FM})nT + \square_{PM} + \square_{DITH})]$

where:

- T is the operating clock period
- f_O is the unmodulated output frequency based on the input value $\square_{INC}$
- f_{FM} is a frequency modulating parameter based on the input value $\square_{FM}$
- Φ_{PM} is derived from the phase modulation input value P and the number of bits (P_{width}) used for this value by the equation: $\square_{PM} = P/2\text{^}P_{width}$
- Φ_{DITH} is the internal dithering value
- A is 2^N-1 where N is the magnitude precision (and N is an integer in the range 10 to 32

The generated output frequency, f_o for a given phase increment, $\square_{inc}$ is determined by the equation: $f_0 = \square_{inc} f_{clk}/2M$ Hz

where M is the accumulator precision and f_{clk} is the clock frequency

The minimum possible output frequency waveform is generated for the case where $\square_{inc} = 1$. This case is also the smallest observable frequency at the output of the NCO, also known as the frequency resolution of the NCO, f_{res} given in Hz by the equation:

$f_{RES} = f_{clk}/2M$ Hz

For example, if a 100 MHz clock drives an NCO with an accumulator precision of 32 bits, the frequency resolution of the oscillator is 0.0233 Hz. For an output frequency of 6.25 MHz from this oscillator, you should apply an input phase increment of:

$(6.25 \times 10^6/100 \times 10^6) \times 2^{32} = 268435456$

The NCO MegaCore function automatically calculates this value, using the specified parameters. IP Toolbench also sets the value of the phase increment in all testbenches and vector source files it generates.

Similarly, the generated output frequency, f_{FM} for a given frequency modulation increment, $\square_{FM}$ is determined by the equation:

$f_{FM} = \square_{FM} f_{clk}/2^F$ Hz

where F is the modulator resolution

The angular precision of an NCO is the phase angle precision before the polar-to-cartesian transformation. The magnitude precision is the precision to which the sine and/or cosine of that phase angle can be represented. The effects of reduction or augmentation of the angular, magnitude, accumulator precision on the synthesized waveform vary across NCO architectures and for different f_o/f_{clk} ratios.

You can view these effects in the NCO time and frequency domain graphs as you change the NCO IP core parameters.

3.1. NCO IP Core Architectures

The NCO MegaCore function supports large ROM, small ROM, CORDIC, and multiplier-based architectures.

3.1.1. Large ROM Architecture

Use the large ROM architecture if your design requires very high speed sinusoidal waveforms and your design can use large quantities of internal memory.

In this architecture, the ROM stores the full 360 degrees of both the sine and cosine waveforms. The output of the phase accumulator addresses the ROM.

The internal memory holds all possible output values for a given angular and magnitude precision. The generated waveform has the highest spectral purity for that parameter set (assuming no dithering). The large ROM architecture also uses the fewest logic elements (LEs) for a given set of precision parameters.

3.1.2. Small ROM Architecture

.To reduce memory usage (but increase LE usage) and increase output frequency, use the small ROM architecture.

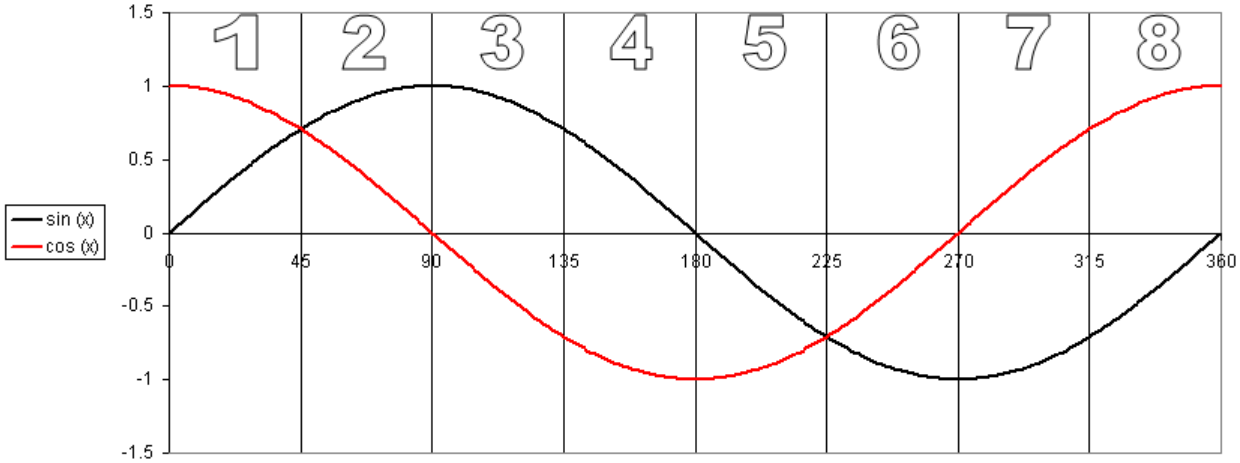
In a small ROM architecture, the device memory only stores 45 degrees of the sine and cosine waveforms. All other output values are derived from these values based on the position of the rotating phasor on the unit circle.

Table 6. Derivation of Output Values

Position in Unit Circle	Range for Phase x	sin(x)	cos(x)
1	0 <= x < π/4	sin(x)	cos(x)
2	π/4 <= x < π/2	cos(π/2-x)	sin(π/2-x)
3	π/2 <= x < 3π/4	cos(x-π/2)	-sin(x-π/2)
4	3π/4 <= x < π	sin(π-x)	-cos(π-x)
5	π <= x < 5π/4	-sin(x-π)	-cos(x-π)
6	5π/4 <= x < 3π/2	-cos(3π/2-x)	-sin(3π/2-x)
7	3π/2 <= x < 7π/4	-cos(x-3π/2)	sin(x-3π/2)
8	7π/4 <= x < 2π	-sin(2π-x)	cos(2π-x)

A small ROM implementation is more likely to have periodic value repetition, so the resulting waveform's SFDR is lower than that of the large ROM architecture. However, you can often mitigate this reduction in SFDR by using phase dithering.

Figure 9. Derivation of output Values



Related Information

Phase Dithering on page 21

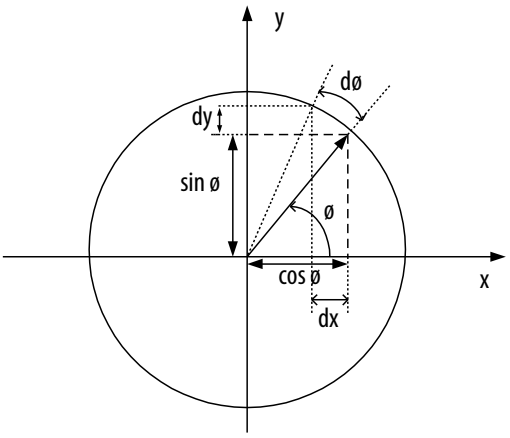
3.1.3. CORDIC Architecture

The CORDIC algorithm, which can calculate trigonometric functions such as sine and cosine, provides a high-performance solution for very-high precision oscillators in systems where internal memory is at a premium.

The CORDIC algorithm is based on the concept of complex phasor rotation by multiplication of the phase angle by successively smaller constants. In digital hardware, the multiplication is by powers of two only. Therefore, the algorithm can be implemented efficiently by a series of simple binary shift and additions/subtractions.

In an NCO, the CORDIC algorithm computes the sine and cosine of an input phase value by iteratively shifting the phase angle to approximate the cartesian coordinate values for the input angle. At the end of the CORDIC iteration, the x and y coordinates for a given angle represent the cosine and sine of that angle, respectively.

Figure 10. CORDIC Rotation for Sine & Cosine Calculation



With the NCO MegaCore function, you can select parallel (unrolled) or serial (iterative) CORDIC architectures:

- You an use the parallel CORDIC architecture to create a very high-performance, high-precision oscillator—implemented entirely in logic elements—with a throughput of one output sample per clock cycle. With this architecture, there is a new output value every clock cycle.
- The serial CORDIC architecture uses fewer resources than the parallel CORDIC architecture. However, its throughput is reduced by a factor equal to the magnitude precision. For example, if you select a magnitude precision of *N* bits in the NCO MegaCore function, the output sample rate and the Nyquist frequency is reduced by a factor of *N*. This architecture is implemented entirely in logic elements and is useful if your design requires low frequency, high precision waveforms. With this architecture, the adder stages are stored internally and a new output value is produced every *N* clock cycles.

3.1.4. Multiplier-Based Architecture

The multiplier-based architecture uses multipliers to reduce memory usage. You can choose to implement the multipliers in either:

- Logic elements (Cyclone series) or combinational ALUTs (Stratix series).
- Dedicated multiplier circuitry (for example, dedicated DSP blocks) (Stratix or Arria series).

Note: When you specify a dual output multiplier-based NCO, the IP core provides an option to output a sample every two clock cycles. This setting reduces the throughput by a factor of two and halves the resources required by the waveform generation unit.

Table 7. Architecture Comparison

Architecture	Advantages
Large ROM	Good for high speed and when a large quantity of internal memory is available. Gives the highest spectral purity and uses the fewest logic elements for a given parameterization.
Small ROM	Good for high output frequencies with reduced internal memory usage when a lower SFDR is acceptable.
CORDIC	High performance solution when internal memory is at a premium. The serial CORDIC architecture uses fewer resources than parallel although the throughput is reduced.
Multiplier-Based	Reduced memory usage by implementing multipliers in logic elements or dedicated circuitry.

3.2. Multichannel NCOs

The NCO IP core allows you to implement multichannel NCOs. You can generate multiple sinusoids of independent frequency and phase t at a very low cost in additional resources. The waveforms have an output sample-rate of f_{clk}/M where M is the number of channels. You can select 1 to 8 channels.

Multichannel implementations are available for all single-cycle generation algorithms. The input phase increment, frequency modulation value and phase modulation input are input sequentially to the NCO with the input values corresponding to channel 0 first and channel (M–1) last. The inputs to channel 0 should be input on the rising clock edge immediately following the de-assertion of the NCO reset.

On the output side, the first output sample for channel 0 is output concurrent with the assertion of `out_valid` and the remaining outputs for channels 1 to (M–1) are output sequentially.

If you select a multichannel implementation, the NCO MegaCore function generates VHDL and Verilog HDL testbenches that time-division-multiplex the inputs into a single stream and demultiplex the output streams into their respective downsampled channelized outputs.

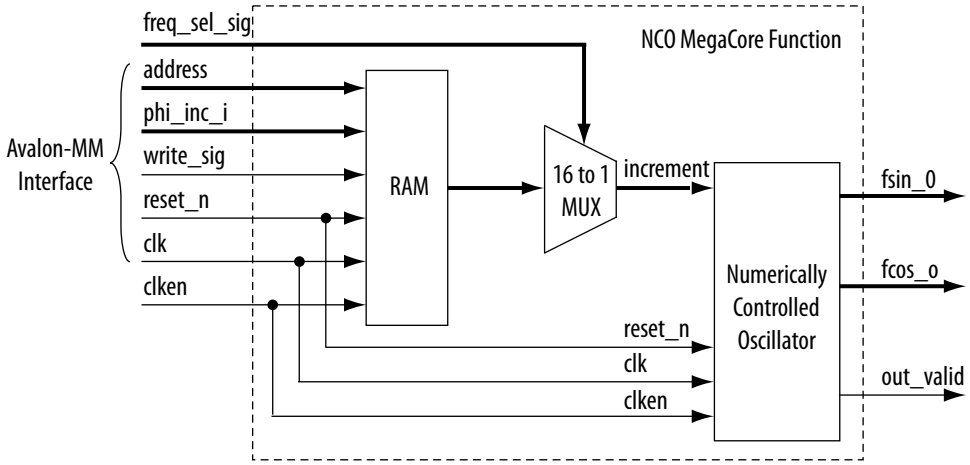
3.3. Frequency Hopping

The NCO IP core supports frequency hopping (except the serial CORDIC architecture). Frequency hopping allows control and configuration of the NCO IP core at run time so that carriers with different frequencies can be generated and held for a specified period of time at specified slot intervals.

The IP core supports multiple phase increment registers that you can load using an Avalon-MM bus. You select the phase increment register using an external hardware signal; changes on this signal take effect on the next clock cycle. The maximum number of phase increment registers is 16.

Note: During frequency hopping, the phase of the carrier should not experience discontinuous change. Discontinuous carrier phase changes may cause spectral emission problems.

Figure 11. Frequency Hopping Block Diagram



The RAM stores all hopping frequencies. The RAM size is <width>×<depth>, where <width> is the number of bits required to specify the phase accumulator value to the precision you select in the parameter editor, and <depth> is the number of bands you select in the parameter editor.

3.4. Phase Dithering

All digital sinusoidal synthesizers suffer from the effects of finite precision, which manifests itself as spurs in the spectral representation of the output sinusoid. Because of angular precision limitations, the derived phase of the oscillator tends to be periodic in time and contributes to the presence of spurious frequencies. You can reduce the noise at these frequencies by introducing a random signal of suitable variance into the derived phase, thereby reducing the likelihood of identical values over time. Adding noise into the data path raises the overall noise level within the oscillator, but tends to reduce the noise localization and can provide significant improvement in SFDR.

The extent to which you can reduce spur levels is dependent on many factors. The likelihood of repetition of derived phase values and resulting spurs, for a given angular precision, is closely linked to the ratio of the clock frequency to the desired output frequency. An integral ratio clearly results in high-level spurious frequencies, while an irrational relationship is less likely to result in highly correlated noise at harmonic frequencies.

The Altera NCO IP core allows you to finely tune the variance of the dither sequence for your chosen algorithm, specified precision, and clock frequency to output frequency ratio, and dynamically view the effects on the output spectrum graphically.

3.5. Frequency Modulation

You can add an optional frequency modulator to your custom NCO variation. You can use the frequency modulator to vary the oscillator output frequency about a center frequency set by the input phase increment. This option is useful for applications in which the output frequency is tuned relative to a free-running frequency, for example in all-digital phase-lock-loops.

You can also use the frequency modulation input to switch the output frequency directly.

You can set the frequency modulation resolution input in the IP core. The specified value must be less than or equal to the phase accumulator precision.

The NCO IP core also provides an option to increase the modulator pipeline level; however, the effect of the increase on the performance of the NCO IP core varies across NCO architectures and variations.

3.6. Phase Modulation

You can use the NCO IP core to add an optional phase modulator to your variation, allowing dynamic phase shifting of the NCO output waveforms. This option is particularly useful if you want an initial phase offset in the output sinusoid.

You can also use the option to implement efficient phase shift keying (PSK) modulators in which the input to the phase modulator varies according to a data stream. You set the resolution and pipeline level of the phase modulator in the NCO wizard. The input resolution must be greater than or equal to the specified angular precision.

3.7. NCO IP Core Parameters

The wizard only allows you to select legal combinations of parameters, and warns you of any invalid configurations.

3.7.1. Architecture Parameters

Table 8. Architecture Parameters

Parameter	Value	Description
Generation Algorithm	Small ROM, Large ROM, CORDIC, Multiplier-Based	Select the required algorithm.
Outputs	Dual Output, Single Output	Select whether to use a dual or single output.
Device Family Target	—	Displays the target device family. The target device family is preselected by the value specified in the Quartus II or DSP Builder software. The HDL that is generated for your variation may be incorrect if you change the device family target in this wizard.
Number of Channels	1–8	Select the number of channels when you want to implement a multichannel NCO.
Number of Bands	1–16	Select a number of bands greater than 1 to enable frequency hopping. Frequency hopping is not supported in the serial CORDIC architecture.

continued...

Parameter	Value	Description
Use dedicated multipliers	On or off	When the multiplier-based algorithm is selected on the Parameters page, turn on to use dedicated multipliers and select the number of clock cycles per output, otherwise the design uses logic elements. This option is not available if you target the Cyclone device family.
CORDIC Implementation	Parallel, Serial	When you select the CORDIC generation algorithm, you can select a parallel (one output per clock cycle) or serial (one output per 18 clock cycles) implementation.
Clock Cycles Per Output	1, 2.	When the multiplier-based algorithm is selected on the Parameters page, you can select 1 or 2 clock cycles per output.

Related Information

NCO IP Core Architectures on page 17

3.7.2. Frequency Parameters

Table 9. Frequency Parameters

Parameter	Value	Description
Phase Accumulator Precision	4 to 64	Select the required phase accumulator precision. The phase accumulator precision must be greater than or equal to the specified angular resolution.
Angular Resolution	4 to 24 or 32	Select the required angular resolution. The maximum value is 24 for small and large ROM algorithms; 32 for CORDIC and multiplier-based algorithms.
Magnitude Precision	10 to 32	Select the required magnitude precision.
Implement Phase Dithering	On or Off	Turn on to implement phase dithering.
Dither Level	Min to Max	When phase dithering is enabled you can use the slider control to adjust the dither level between its minimum and maximum values,
Clock Rate	1 to 999 MHz, kHz, Hz, mHz,	Select the clock rate using units of MegaHertz, kiloHertz, Hertz or milliHertz.
Desired Output Frequency	1 to 999 MHz, kHz, Hz, mHz,	Select the desired output frequency using units of MegaHertz, kiloHertz, Hertz or milliHertz.
Phase Increment Value	—	Displays the phase increment value calculated from the clock rate and desired output frequency.
Real Output Frequency	—	Displays the calculated value of the real output frequency.

Related Information

- Frequency Modulation on page 22
- Phase Modulation on page 22

3.7.3. Optional Ports Parameters

Table 10. Optional Ports Parameters

Parameter	Value	Description
Frequency Modulation input	On or Off	You can optionally enable the frequency modulation input.
Modulator Resolution	4 to 64,	Select the modulator resolution for the frequency modulation input.
Modulator Pipeline Level	1, 2,	Select the modulator pipeline level for the frequency modulation input.

continued...

Parameter	Value	Description
Phase Modulation Input	On or Off	You can optionally enable the phase modulation input.
Modulator Precision	4 to 32,	Select the modulator precision for the phase modulation input.
Modulator Pipeline Level	1, 2,	Select the modulator pipeline level for the phase modulation input.

3.8. NCO IP Core Interfaces and Signals

The NCO MegaCore function is an Avalon-ST source and does not support backpressure.The Avalon-MM interface allows you to control frequency hopping at run time.

Related Information

Avalon Interface Specifications
For more information about the Avalon-MM and Avalon-ST interfaces including integration with other Avalon-ST components which may support backpressure

3.8.1. Avalon-ST Interfaces in DSP IP Cores

Avalon-ST interfaces define a standard, flexible, and modular protocol for data transfers from a source interface to a sink interface.

The input interface is an Avalon-ST sink and the output interface is an Avalon-ST source. The Avalon-ST interface supports packet transfers with packets interleaved across multiple channels.

Avalon-ST interface signals can describe traditional streaming interfaces supporting a single stream of data without knowledge of channels or packet boundaries. Such interfaces typically contain data, ready, and valid signals. Avalon-ST interfaces can also support more complex protocols for burst and packet transfers with packets interleaved across multiple channels. The Avalon-ST interface inherently synchronizes multichannel designs, which allows you to achieve efficient, time-multiplexed implementations without having to implement complex control logic.

Avalon-ST interfaces support backpressure, which is a flow control mechanism where a sink can signal to a source to stop sending data. The sink typically uses backpressure to stop the flow of data when its FIFO buffers are full or when it has congestion on its output.

Related Information

Avalon Interface Specifications

3.8.2. NCO IP Core Signals

Table 11. NCO IP Core Signals

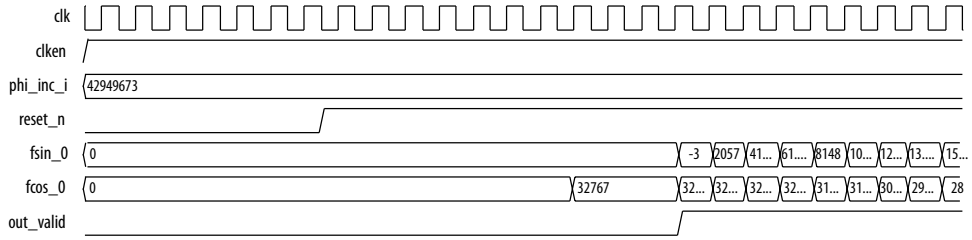
Signal	Direction	Description
`address[2:0]`	Input	Address of the 16 phase increment registers when frequency hopping is enabled.
`clk`	Input	Clock.
`clken`	Input	Active-high clock enable.

continued...

Signal	Direction	Description
freq_mod_i [F-1:0]	Input	Optional frequency modulation input. You can specify the modulator resolution F in IP Toolbench.
freq_sel[log_2N-1:0]	input	Use to select one of the phase increment registers (that is to select the hopping frequencies), when frequency hopping is enabled. N is the depth.
phase_mod_i [P-1:0]	Input	Optional phase modulation input. You can specify the modulator precision P in Ithe wizard.
phi_inc_i [A-1:0]	Input	Input phase increment. You can specify the accumulator precision A in the wizard.
reset_n	Input	Active-low asynchronous reset.
write_sig	Input	Active-high write signal when frequency hopping is enabled.
in_data	Output	In Qsys systems, this Avalon-ST-compliant data bus includes all the Avalon-ST input data signals.
fcos_o [M-1:0]	Output	Optional output cosine value (when dual output is selected). You can specify the magnitude precision M in IP Toolbench.
fsin_o [M-1:0]	Output	Output sine value. You can specify the magnitude precision M in IP Toolbench.
out_valid	Output	Data valid signal. Asserted by the MegaCore function when there is valid data to output.
out_data	Output	In Qsys systems, this Avalon-ST-compliant data bus includes all the Avalon-ST output data signals.

3.8.3. NCO IP Core Timing Diagrams

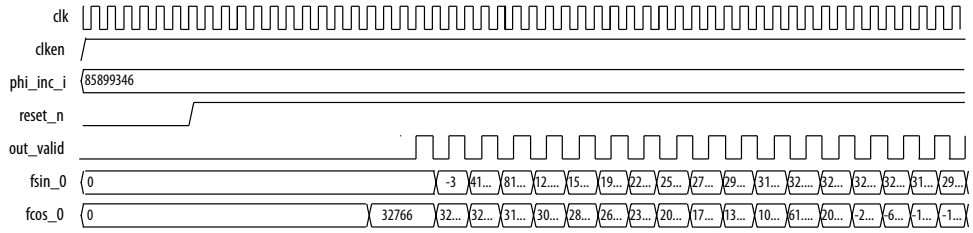
Figure 12. Single-Cycle Per Output Timing Diagram



All NCO architectures, except for serial CORDIC and multi-cycle multiplier-based architectures, output a sample every clock cycle. After the clock enable is asserted, the oscillator outputs the sinusoidal samples at a rate of one sample per clock cycle, following an initial latency of *L* clock cycles. The exact value of *L* varies across architectures and parameterizations.

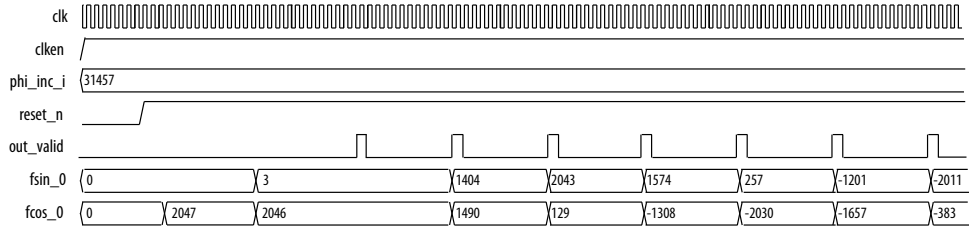
Note: For the non-single-cycle per output architectures, the optional phase and frequency modulation inputs need to be valid at the same time as the corresponding phase increment value. The values should be sampled every 2 cycles for the two-cycle multiplier-based architecture and every *N* cycles for the serial CORDIC architecture, where *N* is the magnitude precision.

Figure 13. Two-Cycle Multiplier-Based Architecture Timing Diagram



After the clock enable is asserted, the oscillator outputs the sinusoidal samples at a rate of one sample for every two clock cycles, following an initial latency of L clock cycles. The exact value of L depends on the parameters that you set.

Figure 14. Serial CORDIC Timing Diagram with N = 8



Note: The `fsin_0` and `fcos_0` values can change while `out_valid` is low.

After the clock enable is asserted, the oscillator outputs sinusoidal samples at a rate of one sample per *N* clock cycles, where *N* is the magnitude precision. The IP core has an initial latency of *L* clock cycles; the exact value of *L* depends on the parameters that you set.

Table 12. Latency Values for Different Architectures

Architecture	Variation	Latency (1), (2)		
		Base	Minimum	Maximum
Small ROM	all	7	7	13
Large ROM	all	4	4	10
Multiplier-Based	Throughput = 1, Logic cells	11	11	17
Multiplier-Based	Throughput = 1, Dedicated, Special case (3)	8	8	14

continued...

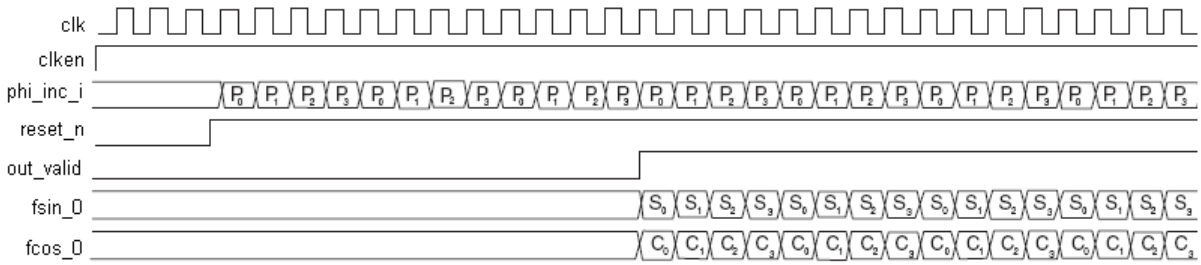
(1) Latency = base latency + dither latency+ frequency modulation pipeline + phase modulation pipeline (×N for serial CORDIC).

(2) Dither latency = 0 (dither disabled) or 2 (dither enabled).

Architecture	Variation	Latency (1), (2)		
		Base	Minimum	Maximum
Multiplier-Based	Throughput = 1, Dedicated, Not special case	10	10	16
Multiplier-Based	Throughput = 1/2	15	15	26
CORDIC	Parallel	2N + 4	20 (4)	74 (5)
CORDIC	Serial CORDIC	2N + 2	18 (6)	258 (7)

Figure 15. Multi-Channel NCO Timing Diagram with M = 4.

The IP core sequentially interleaves and loads input phase increments for each channel, P_k



The phase increment for channel 0 is the first value read in on the rising edge of the clock following the de-assertion of reset_n (assuming clken is asserted) followed by the phase increments for the next (M-1) channels. The output signal `out_valid` is asserted when the first valid sine and cosine outputs for channel 0, S_0, C_0, respectively are available.

The output values S_k and C_k corresponding to channels 1 through (M-1) are output sequentially by the NCO. The outputs are interleaved so that a new output sample for channel k is available every M cycles.

(1) Latency = base latency + dither latency+ frequency modulation pipeline + phase modulation pipeline (×N for serial CORDIC).

(2) Dither latency = 0 (dither disabled) or 2 (dither enabled).

(3) Special case: (9 <= N <= 18 && WANT_SIN_AND_COS).

(4) Minimum latency assumes N = 8.

(5) Maximum latency assumes N = 32

(6) Minimum latency assumes N = 8.

(7) Maximum latency assumes N = 32

A. NCO IP Core User Guide Document Archives

If an IP core version is not listed, the user guide for the previous IP core version applies.

IP Core Version	User Guide
14.1	NCO IP Core User Guide v14.1

5. Document Revision History

NCO IP User Guide revision history

Date	Version	Changes Made
November 2017	2017.11.06	• Deleted reference to MATLAB testbench support • Removed design example • Added support for Intel Cyclone 10 devices • Changed *small ROM architecture* description. • Corrected "deviation of output values" table position 2.
2014.12.15	14.1	• Added full support for Arria 10 and MAX 10 devices • Reordered parameters tables to match wizard
August 2014	14.0 Arria 10 Edition	• Added support for Arria 10 devices. • Added new in_data and out_data bus descriptions. • Added Arria 10 generated files description. • Removed table with generated file descriptions.
June 2014	14.0	• Removed device support for Cyclone III and Stratix III devices • Added support for MAX 10 FPGAs. • Added instructions for using IP Catalog
November 2013	13.1	• Removed support for the following devices: — Arria — Cyclone I — IHardCopy II, HardCopy III, and HardCopy IV — Stratix, Stratix II, Stratix GX, and Stratix II GX • Added full support for the following devices: — Arria V — Stratix V
November 2012	12.1	Added support for Arria V GZ devices.

Mouser Electronics

Authorized Distributor

Click to View Pricing, Inventory, Delivery & Lifecycle Information:

Altera:

IP-NCO