SDx Development Environment

Release Notes, Installation, and Licensing Guide

UG1238 (v2017.4) December 20, 2017

XILINX
ALL PROGRAMMABLE™

Table of Contents

Revision History **5**

Chapter 1: Release Notes and Supported Hardware **7**
SDSoC - SDAccel Development Environment Common Features 7
SDSoC Development Environment Features 8
SDAccel Development Environment Features 10

Chapter 2: Introduction to the SDx Environments **23**
SDSoC Overview 23
SDAccel Overview 24
Hardware Requirements 24
Software Requirements 25
About the SDSoC Installation 25
About the SDAccel Installation 26

Chapter 3: Obtaining a License on the Xilinx Licensing Site **31**

Chapter 4: Installing the SDx Environments **33**
Preparing to Install the Tools 33
Installing SDSoC and SDAccel 33

Appendix A: Additional Resources and Legal Notices **43**
References 43
Documentation Navigator and Design Hubs 44
Please Read: Important Legal Notices 45

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
12/20/2017	2017.4	Revised to reflect changes for 2017.4 SDx software.
08/16/2017	2017.2	Revised to reflect changes for 2017.2 SDx software.
06/20/2017	2017.1	Revised to reflect changes for 2017.1 SDx software.

XILINX
ALL PROGRAMMABLE™

Chapter 1

Release Notes and Supported Hardware

SDSoC - SDAccel Development Environment Common Features

SDSoC™ and SDAccel™ are now fully integrated with Vivado® Design Suite, no longer requiring a separate Vivado Design Suite image. The SDx™ Eclipse UI (SDx GUI) has been enhanced to increase user productivity. A background build no longer blocks additional GUI actions, and the new project creation wizard has been streamlined.

The 2017.4 release uses 5.x DSA platforms. These platforms enable dynamic configuration of the platforms.

The SDxSDx compiler automatically optimizes away interface logic associated with unused DDR at linking time. Logic for capturing profiling and debug information may be added dynamically during the compilation process. Previously, this logic was required to already exist in the DSA. Dynamic platforms make better use of the available resources on the FPGA and improve compilation times by up to 45%.

Many of the new features provided in this release require the use of 5.x DSA. See SDAccel Migration Summary to update existing projects to to use the new 5.x DSAs and the associated features.

SDSoC Development Environment Features

SDSoC Development Environment Changes for 2017.4

The following are changes to existing features in the SDSoC™ Development Environment in this release:

- SDx GUI - The project creation and platform creation process has been updated. This now provides additional functionality to provide for:
  - The ability to create platform projects from within the SDx GUI.
  - The ability to create system projects for multiple SDSoC applications.
- SDSoC `sds++` system compiler enhancements.
  - Advanced FPGA control of Vivado through the `--xp<Vivado tcl command>` option, which allows you to specify Vivado parameters and properties, including Tcl commands and flow pre or post Tcl scripts (the `-vpl-ini <file>` option, can be used to specify a file containing multiple `<parameter_value>` options).
  - Accelerator connectivity to streaming IO from FPGA pins can now be specifed with `#pragma SDS data sys_port`
  - Zynq and Zynq Ultrascale+ base platforms now employ a clock wizard to support up to seven phase aligned clocks.
- System projects for multiple SDSoC applications.
- Streaming IO ports may now be implemented using a simple pragma, `sys_port`.
- Programmable clocks now specified by a clock wizard.
  - The ZCU102 platform now supports up to seven clocks.
- Support for SDSoC platform creation
  - The SDSoC Platform Utility has been replaced by a first-class platform project type in the SDx GUI.
  - The `sdsoc:::` namespaced Tcl APIs to declare the hardware platform interface (`.hpfm` file) have been replaced by Vivado IP Integrator Tcl APIs
  - The SDx GUI now generates a Tcl journal file for batch scripting of platform builds and improved revision control.
  - A platform can now include its own IP cache.
  - The platform hardware handoff from Vivado to the SDx GUI platform project is now generated with the `write_dsa` command.

- A migration guide for SDSoC platforms built with prior releases is contained in Appendix D: Migrating SDSoC Platforms of *SDSoC Environment Platform Development Guide* (UG1146)

SDSoC Development Environment What's New for 2017.4

The following SDSoC™ Development Environment updates are included in this release:

- SDx™ GUI.
  - A new Platform Project type for creating an SDSoC platform after exporting a design support archive (DSA) from Vivado.
  - A new System Project type that includes multiple sub-projects that target a platform.
  - Enhanced support to select hardware functions from XFAST source code libraries for OpenCV and image processing applications.
- A new utility `sdx_pack` for packaging RTL accelerators into a C callable library.
- Concurrent invocation of Vivado HLS to compile accelerator functions that reside in separate files.
- Additional sds++ enhancements
  - Enhanced dataflow analysis, resource sharing, system port load balancing, and automatic barrier synchronization (wait insertion) to improve system performance in accelerator pipelines.
  - Support for multiple array members within a struct/class argument.
- FreeRTOS updated to v9.0.1, using the Xilinx SDK port (freertos901_xilinx).
- Cortex-R5 compiler support updated to gcc 6.2.1.
- Improved error handling and error messages.
- Zybo and microzed platforms are now available only from the board vendors.
- Bug fixes.

SDAccel Development Environment Features

SDAccel Development Environment Changes for 2017.4

The following are changes to existing features in the SDAccel™ development environment in this release:

- New 5.x version DSAs ("dynamic" DSAs), which are now configurable at link time to free up resources for kernels, reduce hardware compile times, and improve timing closure in Vivado.
  - In dynamic platforms, logic in the DSA, which is not required, such as the logic used to implement an unused DDR memory is optimized away during compilation.
  - When using 4.x or earlier DSAs, Software and Hardware Emulation require the prepending of the 4.x DSA libraries to the `LD_LIBRARY_PATH`.
  - When using 4.x or earlier DSAs, only design examples from the 2017.2 or earlier Xilinx Github may be used.
    - All 2017.4 SDAccel Github examples will only work with 5.x DSA versions.
- xocc compiler
  - xocc compiler for OpenCL kernel compilation has been upgraded to LLVM 3.9.
  - Starting 2017.4, two advanced transformations were added that should help getting consistently high global memory bandwidth utilization in OpenCL kernels. This is achieved in two complementary ways:
    - Widening / Vectorization of the datapath of the memory interface.
    - Enhanced burst inference on the array access to the memory interface.

    When both those transformation are triggered, speed-ups by as much as 5X can be achieved (usually, up to saturation of the memory bandwidth).
- Xilinx SDAccel Runtime
  - Use of multiple xclbin files.
    - Cases where the host code uses multiple Xclbin files must now free the current Xclbin using `clReleaseProgram()` and `clReleaseKernel()` before proceeding to load subsequent Xclbin files.
  - Updated memory selection.

    - When using multiple DDR memories, the selection is now made at compile time using the `xocc` command line switch `-sp` and the `map_connect` option is now deprecated.
    - Existing host code, which specifies DDR banks using `clCreateBuffer`, should be updated to use the new command line switches.
  - Shared library
    - When you use HLS math library in SDx host code, you must add additional linkage information to your Makefile to find the dynamic library. For example:

```
$(XILINX_VIVADO)/lnx64/lib/csim -lhlsmc++-GCC46
```

  - Enhanced xbsak features
    - The command line option `dmatest` now performs a DMA test on all the DDR banks used in the application and specified in the `xocc` command line.
    - The command line option `boot` now forces a re-enumeration of the PCIe® bus and bus re-scanning.
    - The command line option `scan` is enhanced to provide more details about the OS.
    - The command line option `classis` is depreciated and no longer supported.
    - The command line option `query` is enhanced to provide more details on each CU, such as the name, the kernel type, index, address and status.
- Profile Features
  - Kernel Instrumentation is now controlled with the `xocc` compile option `-profile_kernel`
    - The compile time option allow the insertion of additional hardware logic to enable the generation of profiling data. This is enabled by default when using the IDE and may be disabled.
  - The Profile Summary Report and Timeline Trace Report are now generated using the `sdx_analyze` utility. This replaces the existing `sda2protobuf` and `sda2wdb` utilities.
- Debug Features
  - A new `xgdb` utility is provided for Application Debug. This is required to peform Debug when using the command line interface.
    - This ensures any existing GDB debug features are not changed by SDAccel.
    - Application debug in the SDx GUI remains unchanged.
- RTL Kernels
  - Ensure that all AXI outputs of RTL kernels are driven. Driverless outputs on RTL kernels may lead to `xocc` failures after synthesis step with messages like the following one.

```
Designs upgraded from V4.x DSAs to V5.x DSAs may be more sensitive to
driverless output errors.
```

SDAccel Development Environment What's New for 2017.4

The following SDAccel™ Development Environment updates are included in this release.

5.x DSAs and features

Table 1: **xilinx_vcu1525_dynamic_5_0**

Area	SLR 0	SLR 1	SLR 2
General information			
SLR description	Bottom of device; dedicated to dynamic region.	Middle of device; shared by dynamic and static region resources.	Top of device; dedicated to dynamic region.
Dynamic region pblock name	`pfm_top_i_dynamic_region_pblock_dynamic_SLR0`	`pfm_top_i_dynamic_region_pblock_dynamic_SLR1`	`pfm_top_i_dynamic_region_pblock_dynamic_SLR2`
Compute unit placement syntax1	set_property CONFIG.SLR_ASSIGNMENTS SLR0 [`get_bd_cells <cu_name>`]	set_property CONFIG.SLR_ASSIGNMENTS SLR1 [`get_bd_cells <cu_name>`]	set_property CONFIG.SLR_ASSIGNMENTS SLR2 [`get_bd_cells <cu_name>`]
Global memory resources available in dynamic region			
Memory channels; system port name	bank0 (16GB DDR4)	bank1 (16GB DDR4, in static region) bank2 (16GB DDR4, in dynamic region)	bank3 (16GB DDR4)
Approximate available fabric resources in dynamic region2			
CLB LUT	388K	199K	388K
CLB Register	776K	399K	776K
Block RAM Tile	720	420	720
URAM	320	160	320
DSP	2280	1320	2280

1. Dynamic platforms will by default place a kernel in the same SLR as the memory bank that it accesses. Details on how this may be controlled are provided in the User-specified SLR assignments for Kernels section of the *SDAccel Environment User Guide* (UG1023)
2. Approximately 20K CLB LUTs and 20K CLB Registers are required for each mapped memory channel (except for bank1, in static region). A minimum of 12K CLB LUTs and 18K CLB Registers are also required for the SmartConnect network, with additional resources required for each mapped memory channel and each compute unit.

Table 2: **xilinx_kcu1500_dynamic_5_0**

Area	SLR 0	SLR 1
SLR description	Bottom of device; shared by dynamic and static region resources.	Top of device; dedicated to dynamic region.

Table 2: **xilinx_kcu1500_dynamic_5_0** *(cont'd)*

Area	SLR 0	SLR 1
Dynamic region pblock name	pfm_top_i_dynamic_region_ pblock_dynamic_SLR0	pfm_top_i_dynamic_region_ pblock_dynamic_SLR1
Compute unit placement syntax1	set_property CONFIG.SLR_ASSIGNMENTS SLR0 [get_bd_cells <cu_name>]	set_property CONFIG.SLR_ASSIGNMENTS SLR1 [get_bd_cells <cu_name>]
Memory channels; system port name	bank0 (4GB DDR4) bank1 (4GB DDR4)	bank2 (4GB DDR4) bank3 (4GB DDR4)
CLB LUT	264K	325K
CLB Register	529K	651K
Block RAM Tile	876	1080
DSP	2217	2760

1. Dynamic platforms will by default place a kernel in the same SLR as the memory bank that it accesses. Details on how this may be controlled are provided in the User-specified SLR assignments for Kernels section of *SDAccel Environment User Guide* (UG1023).
2. Approximately 20K CLB LUTs and 20K CLB Registers are required for each mapped memory channel. A minimum of 12K CLB LUTs and 18K CLB Registers are also required for the SmartConnect network, with additional resources required for each mapped memory channel and each compute unit.

The 2017.4 release supports 4.X DSA as backward compatibility with the `xocc` interfaces the same as 2017.2. For 5.X DSA, please use the `xocc` option changes as provided in the SDAccel Migration Summary.

You need to repackage any existing RTL kernel using 2017.4 irrespective of DSA for running in 2017.4.

Starting 2018.2, `xocc` will only support 5.X DSA (and the corresponding `xocc` options). 4.X DSA will be deprecated in 2018.2.

SDx™ GUI

- The Vivado® IDE may be launched directly from the SDx GUI
  - This allows experienced hardware designers, or those familiar with the Vivado Design Environment, to perform implementation changes to the hardware (detailed timing closure etc.) and save the results.

    In addition, a pre-synthesized or a pre-implemented Vivado Design Checkpoint (`.dcp`) file can be brought in from the launched Vivado® session, and directly used in the SDx session to complete the remaining flow without having to start from the beginning.

Changes made during the Vivado session are also automatically captured in SDx for subsequent runs.

- The RTL Kernel Wizard may now be launched directly from the SDx GUI.
- RTL Kernel Wizard has been enhanced to support additional types of packaging options, including pre-compiled kernels and netlist (`.dcp`) based kernels.
- Dataflow is an important feature in `xocc` (for both C/C++ as well as OpenCL kernels). Several DRC (with extended documentation) related to C/C++/OpenCL designs with dataflow have been added.
- DRC window highlighting key DRC in the user C/C++/OpenCL code is available at the bottom of the SDx GUI next to the **Console** tab.

Kernel Performance Enhancements

- Improved data transfer rates are now provided through the following means.
  - Automatic memory coalescing and widening. The automatic widening of the data transfer may be disabled by adding the nounroll pragma to for-loops
  - Manually specified memory coalescing using the new Xilinx OpenCL attribute `xcl_zero_global_work_offset`, which may be used when `clEnqueueNDRangeKernel` is used without `global_work_offset`.
  - Xilinx highly recommends that you use a correctly specified `global_work_offset`.
- The work group size is now automatically inferred based on OpenCL semantics.
  - Xilinx highly recommends that you use a correctly specified `global_work_offset`.

Whole-function vectorization is provided through the vec_type_hint attribute on NDRange.

- It is highly recommended to use `vec_type_hint` for improved performance.
- Whole-function vectorization may increase the size of the hardware implementation.

Sub-functions are now automatically inlined to improve performance

This may be disabled using the `noinline` pragma and OpenCL attribute.

Xilinx SDAccel Runtime

- Support is now provided for the OpenCL API `clCreateSubDevices`.
  - Sub-devices may be created for each Compute Unit (CU) allowing for multiple independent command queues for each CU.
  - Each sub-device may include only one CU.
- Improved Linux Driver support

- Drivers now use the Linux DMA_BUF framework allowing data sharing across all Linux devices.
- Enables the exporting of device data (temp, current, etc.) via Linux SysFS framework.

RTL Kernel Enhancements

- Support for compile time parameterization of RTL kernels via the `xocc` command line.
- A new `xocc` command line option allows a single RTL Kernel (`.xo` file) to be instantiated as multiple kernel instances. In addition, these separate instances can be queued independently from each other.
- RTL kernels can now be pre-compiled, to reduce SDx compile flow time by not having to do synthesis under SDx.
- RTL Kernels may be created from a Xilinx checkpoint (`.dcp`) file.
- Encryption support is now provided for RTL Kernels.

XOCC Enhancements

- `--ini_file` switch can be used to pass a set of advanced `--xp style` switches to xocc using a single file (similar to use of `xocc.ini` file).
- `--report_dir` switch allows report files generated under SDx runs to be copied to a separate directory for easy access.
- `--log_dir` switch allows log files generated under SDx runs to be copied to a separate directory for easy access.
- `--temp_dir` switch allows a user specified directory to be used for generation of temporary files.
- `--interactive` switch allows Vivado to be launched from within the `xocc` environment, with the right project loaded.
- `--reuse_synth` switch allows a pre-synthesized Vivado Design Checkpoint (.dcp) file to be brought in and used directly in SDx flow to complete implementation and xclbin generation.
- `--reuse_impl` switch allows a pre-implemented and timing closed Vivado Design Checkpoint (`.dcp`) file to be brought in and used directly in SDx flow to do `xclbin` generation.
- `--remote_ip_cache` switch allows usage of a user specified IP cache location. This will improve iterative SDx flow run times.
- `--user_ip_repo_paths` switch allows usage of additional read only IP cache locations, as well as custom IP definitions to be used in SDx.
- `--no_ip_cache` switch can be used to turn off all usages of IP caches. This is generally not recommended other than debugging purposes.

Profile Features

- Profile instrumentation of kernels is now enabled through `xocc` compile option `-profile_kernel`.
- Profile reports generated using `sdx_analyze` utility.
- Profile Summary Report Enhancements:
  - Data Transfer table now displays information on a compute unit/port basis including kernel arguments and DDR bank.
  - Compute Unit Table now reports clock frequency per Compute Unit.

Debug Features

- Kernel Debug is now supported through GDB and TCF in Hardware Emulation. This provides the ability to:
  - Start and stop at intermediate points in the execution of the kernels.
  - Inspect both kernel arguments and global memory.
- Application Debug: The following new gdb extensions provide enhanced debug information:
  - `xprint kernel`: Displays all NDRange events that are pending and their arguments
  - `xprint all`: Displays all valid OpenCL objects
  - `xstatus all`: Provides visibility into the IPs instantiated on the platform
  - A new `xocc` command option `-dk` to insert a Light Weight Protocol Checker IP into the system to debug AXI protocol violations.

Emulation Features

- `XCL_EMULATION_MODE`: The `XCL_EMULATION_MODE` environment variable now requires a value of `sw_emu` or `hw_emu`. Setting `XCL_EMULATION_MODE` environment variable to `sw_emu` changes the application execution to software emulation mode, `hw_emu` enables hardware emulation mode. Unset the `XCL_EMULATION_MODE` variable to disable emulation.
- Memory checks on out-bounds-accesses and invalid read or write operations (writing to a read only device or vice versa) are provided during Software Emulation.

SDAccel Migration Summary

The following table specifies changes to existing flows and scripts which are required when using the 2017.4 release.

Table 3: **SDAccel Migration Summary**

Area	2017.4 Behavior	Required Update for 2017.4
DSA	5.x DSA are the primary supported DSA for 2017.4. You can use existing 4.X DSA from previous release, with the 2017.2 xocc options	Xilinx strongly recommends that you update to 5.x DSA Platforms, specify the platform should be using the `-platform` option.
	Now use "_" character as separator instead of ":".	Platform names now use the character "_" as a separator. This requires updating the DSA name in the `xocc` command option `--platform`.
xocc options	Single step compilation no longer supported.	Kernels must be compiled using `-compile` (`-c`) option and then linked using the `-link` (`-l`) option as two separate `xocc` commands. When using the SDx™ GUI, it automatically generates the correct makefile with this style of `xocc` switches.
	If using multiple DDR banks, the `--xp map_connect` option has been removed and replaced with the `-sp` options with a much simpler syntax.	The new `xocc` command option `-sp` is now used to specify system ports. The following shows how an existing `map_connect` option may be changed for 2017.4. (2017.4) `--xp misc:map_connect=add.kernel.krnl_idct_1.M_AXI_GMEM.core.OCL_REGION_0.M00_AXI` (2017.4) `--sp krnl_idct_1.m_axi_gmem:bank0`
	The location of include files has changed.	If using Xilinx `include` files, change the `include` pathname from `-I$(XILINX_SDACCEL)/Vivado_HLS/include/` to `-I$(XILINX_SDACCEL)/include`

Table 3: **SDAccel Migration Summary** *(cont'd)*

Area	2017.4 Behavior	Required Update for 2017.4
Host Code	Compiling host code on an Ubuntu OS.	When compiling host code on an Ubuntu OS, you must explicitly use –std=c++14. Note that system header file, `sys/cdefs.h` is located in the directory `/usr/include/x86_64-linux-gnu` and not in the `/usr/include` directory. This may require an update to your `include` paths.
	If using more than one `xclbin` file, the first must be released before second is loaded.	For multiple `xclbin` files you must now free the current `xclbin` using `clReleaseProgram()` and `clReleaseKernel()` before proceeding to load subsequent `xclbin` files.
	Multiple kernels accessing the same DDR.	If there are independent kernels that access the same DDR, and there is a cyclic dependency between them, it may lead to a deadlock. You need to break the dependency between the kernels, either by accessing different DDRs for access, or by ensuring that there is synchronization between them. For example, if there are two kernels, each accessing the DDR bank, for a total of 4 read and write operations (two reads and two writes each), you should use one DDR for the reads for the two kernels and one DDR for doing the writes for the same kernel.
	Accessing memory size beyond what was linked in `xocc` might have worked.	Host code will only have access to memory size that is available to kernel and will be dependent on no banks being used in design. Previous designs that accessed memory size beyond allocated will not work as `xocc` will optimize away DDR interfaces that are not being set using `-p`.
	Shared library	When you use HLS math library in SDx host code, you must add additional linkage information to your Makefile to find the dynamic library. For example: `$(XILINX_VIVADO)/lnx64/lib/csim -lhlsmc++-GCC46`

Table 3: **SDAccel Migration Summary** *(cont'd)*

Area	2017.4 Behavior	Required Update for 2017.4
C/C++ OpenCL Kernel Code	Pipes must now use all lower case names.	The variable used in pipes must now all be lower cases. For example, the following cannot use 'pipe int inFifo', and must use all lower case. `pipe int infifo _attribute_((xcl_reqd_pipe_depth(16)));`
	The OpenCL 2.0 image functions were supported by default. These now require a compiler option.	Use of OpenCL 2.0 image functions generates an error, and require you to enable 3.1 compiler using the `xocc` command option `--xp param:compiler.version=31`.
	Enhance memory coalescing to improve memory bandwidth and speed up kernel execution	Loop vectorization may automatically happen to improve memory bandwidth. Sometimes this will increase resource usage. You can turn this off by using the following at the loop impacted. `#pragma nounroll`
	The xcl_dependence attribute requires using the 3.1 compiler version.	Use of the `xcl_depedence` attribute generates an error. You will have to either remove the attribute or enable 3.1 compiler using the `xocc` command `option--xp param:compiler.version=31`.
	Default data width of OpenCL Kernel interface.	The default data width of the OpenCL Kernel AXIM interface has changed to 512 bits. This should improve the performance of kernels but may impact timing closure on some designs.

	C and C++ kernels which use the interface bundle option.	The bundle option, as shown below for AXI and AXI-Lite interface must use all lower case names for the bundle name. In both cases below, the name "bar" is all lower case.
		`#pragma interface s_axilite port=foo bundle=bar`
		`#pragma interface m_axi port=foo bundle=bar`

Table 3: **SDAccel Migration Summary** *(cont'd)*

Area	2017.4 Behavior	Required Update for 2017.4
RTL Kernel	RTL Kernel XO file Packaged Using RTL Kernel Wizard	Rerun the RTL Kernel Wizard and choose same options as from 2017.4_sdx. In the Vivado Project, re-introduce your RTL adapting to top level AXI interface port list. Then repackage the kernel to generate a new XO file. This will ensure the RTL kernel has all the latest meta data required for 2017.4 tool-flow and runtime.

	RTL Kernel Packaged by user without using RTL Kernel Wizard	RTL kernels manually packaged, must be re-packaged to work with 2017.4.
		The following commands added to inject required new meta data:
		`set_property sdx_kernel true [ipx::current_core]`
		`set_property sdx_kernel_type rtl [ipx::current_core]`

		Every IP must have primary clock named `ap_clk` which drives the IP and all AXI interfaces on the IP. It can optionally have a secondary clock, but this must be named `ap_clk_2`. The former has associated reset port `ap_rst_n` and when the latter exists, it must have a corresponding reset port `ap_rst_n_2`.
		If you are manually packaging RTL Kernel (not using RTL Kernel Wizard), port interface names has to be consistent between kernel.xml and component.xml (case sensitive).
	RTL kernel port names in `component.xml` and `kernel.xml` have to match case	RTL kernel port names in component.xml and `kernel.xml` must match case.
	RTL kernel with pipes	• Within the `kernel.xml`, stream pipes names must be all lower case. No mixed or upper case will be supported. • Global BRAM pipe connections are not supported.
	Hand edited RTL kernel.xml	Every declared port (port/@name) must be mapped from, i.e., equal to some kernel argument (arg/@port

Table 3: **SDAccel Migration Summary** *(cont'd)*

Area	2017.4 Behavior	Required Update for 2017.4
Profiling	Profiling hardware may be dynamically added to any DSA and is no longer required to be pre-built in the platform. This requires an update to the `xocc` command options.	Profiling is now performed as two-step process of enabling the collection of profiling data and the saving the data. Enable profile instrumentation by compiling the kernels with the `–profile_kernel` option. Set `profile=true` in the `sdaccel.ini` file to collect profile data.
	`sda2protobuf` was used to generate profile summary report.	Replaced by `sdx_analyze profile <options>`.
	`sda2wdb` was used to generate application timeline trace report.	Replaced by `sdx_analyze trace <options>`.
	`sdxsyschk` utility to check Xilinx PCIe platform status .	Deprecated and replaced by `xbsak scan` command.

XILINX
ALL PROGRAMMABLE™

Chapter 2

Introduction to the SDx Environments

The 2017.4 SDx™ Environment software release consists of the SDSoC™ Development Environment for Zynq® UltraScale+™ MPSoC and Zynq®-7000 AP SoC families, and the SDAccel™ Development Environment for Data Center and PCIe®based accelerator systems. These environments share a common installer, but are licensed individually. All SDx Environments include the Vivado® Design Suite for programming the target devices and for developing custom hardware platforms.

SDSoC Overview

The SDSoC™ (Software-Defined System On Chip) environment is an Eclipse-based Integrated Development Environment (IDE) for implementing heterogeneous embedded systems using Zynq®-7000 AP SoCand Zynq UltraScale+ MPSoCplatforms. The SDSoC environment provides an embedded C/C++ application development experience with an easy-to-use Eclipse IDE, and comprehensive design tools for heterogeneous Zynq-7000 AP SoC and Zynq UltraScale+ MPSoC development to software engineers and system architects.

The SDSoC environment includes a full-system optimizing C/C++ compiler that provides automated software acceleration in programmable logic combined with automated system connectivity generation. The application programming model within the SDSoC environment should be intuitive to software engineers. An application is written as C/C++ code, with the programmer identifying a target platform and a subset of the functions within the application to be compiled into hardware. The SDSoC system compiler then compiles the application into hardware and software to realize the complete embedded system implemented on a Zynq device, including a complete boot image with firmware, operating system, and application executable.

The SDSoC environment abstracts hardware through increasing layers of software abstraction that includes cross-compilation and linking of C/C++ functions into programmable logic fabric as well as the ARM CPUs within a Zynq device. Based on a user specification of program functions to run in programmable hardware, the SDSoC environment performs program analysis, task scheduling and binding onto programmable logic and embedded CPUs, as well as hardware and software code generation that automatically orchestrates communication and cooperation among hardware and software components.

The SDSoC Environment also supports OpenCL™ applications on Xilinx provided base development platforms, with OpenCL kernels that target programmable logic in Zynq and Zynq UltraScale+ MPSoC devices.

SDAccel Overview

SDAccel™ is a development environment for OpenCL™ applications targeting Xilinx FPGA based accelerator cards. This environment enables concurrent programming of the in-system processor and the FPGA fabric without the need for extensive FPGA design experience. The application is captured as a host program written in OpenCL C and a set of computation kernels expressed in C, C++, OpenCL C, or RTL.

Hardware Requirements

SDSoC Hardware Requirements

The 2017.4 SDSoC™ environment release includes support for the following development boards:

- ZC702, ZC706, and ZedBoard development boards featuring the Zynq®-7000 AP SoC
- ZCU102 development board featuring the Zynq® UltraScale+™ MPSoC.

Additional platforms are available from partners. Also, the SDSoC Platform Utility enables you to target any custom Zynq and Zynq UltraScale+ MPSoC board. For more information, visit the SDSoC Developer Zone: https://www.xilinx.com/products/design-tools/software-zone/sdsoc.html.

You also need a mini-USB cable to observe the UART output from the board.

SDAccel Hardware Requirements

The SDAccel™ environment requires the following hardware:

- Acceleration Card. Use one of the following:
  - Xilinx Kintex UltraScale FPGA KCU1500 Reconfigurable Acceleration card based on XCKU115-FLVB2104-2-E FPGA.
  - Xilinx Virtex UltraScale+ FPGA VCU1525 Reconfigurable Acceleration card based on XCVU9P-L2FSGD2104E FPGA.

- Host computer: Desktop computer for hosting the acceleration card. The host computer must provide the following.
  - Motherboard with a PCIe Gen3 X8 slot
  - 16 GB RAM
  - 100GB free disk space
  - DVD drive
- Programming computer: Laptop or desktop computer running the supplied Vivado Design Suite 2017.4 for programming the FPGA.
- Xilinx® Platform Cable USB 2, part number HW-USB-II-G for connecting the programming computer to the acceleration card. See the *Platform Cable USB II Data Sheet*, (DS593).
- Additional platforms are available from partners. For more information, visit the SDAccel Developer Zone: https://www.xilinx.com/products/design-tools/software-zone/sdaccel.html.

Software Requirements

The SDx™ Development Environment runs on both Linux and Windows operating systems. The supported operating systems are listed below.

- Windows 7 and 7 SP1 Professional (64-bit) (SDSoC only)
- Windows 10 Professional (64-bit) (SDSoC only)
- Linux Support
  - Red Hat Enterprise Workstation/Server 7.2-7.3 (64-bit)
  - Red Hat Enterprise Workstation 6.7, 6.8, and 6.9 (64-bit), (SDSoC only)
  - Red Hat Enterprise Workstation 6.9 (64-bit)
  - CentOS 6.8, CentOS 7.3 (64-bit) (SDAccel)
  - Ubuntu Linux 16.04.2 LTS (64-bit)

About the SDSoC Installation

The installation of SDSoC™ includes the following:

- SDSoC environment, including an Eclipse/CDT-based GUI, high-level system compiler, and ARM GNU toolchain

- Vivado® Design Suite System Edition, with Vivado High-Level Synthesis (HLS) and the Xilinx® Software Development Kit (SDK)

The SDSoC environment includes the same GNU ARM toolchain included with the Xilinx Software Development Kit (SDK), which also provides additional tools used by the SDSoC environment. The SDSoC environment setup script sets PATH variables to use this toolchain.

More information about the SDSoC installation:

- The provided toolchains contain 32-bit executables, requiring your Linux host OS installation to include 32-bit compatibility libraries.
- RHEL 6 and 7 64-bit x86 Linux installations might not include the 32-bit compatibility libraries, and might need to be added separately; see https://access.redhat.com/site/solutions/36238.
- On RHEL, 32-bit compatibility libraries can be installed by becoming a superuser (or root) with root access privileges and running the yum install glibc.i686 command.
- On Ubuntu, 32-bit compatibility libraries can be installed by becoming a superuser (or root) with root access privileges and running the following commands. Refer to SDSoC Development Environment Features for additional information.
-
```
sudo dpkg --add-architecture i386
sudo apt-get update
sudo apt-get install libc6:i386 libncurses5:i386 libstdc++6:i386
sudo apt-get install g++-multilib
sudo apt-get install libgtk2.0-0:i386 dpkg-dev:i386
sudo ln -s /usr/bin/make /usr/bin/gmake
```
- The version of the toolchain can be displayed by running the `arm-linux-gnueabihf-g++ -v` command.
- The last line of the output printed in the shell window should be GCC version 4.9.2 20140904 (pre-release)(crosstool-NG linaro-1.13.1-4.9-2014.09 - Linaro GCC 4.9-2014.09).

About the SDAccel Installation

The SDAccel™ Environment runs on the Linux operating systems only with no support for Windows. It supports RedHat Enterprise Linux, CentOS 6.9 and 7.3 64-bit, as well as Ubuntu 16.04 64-bit.

CentOS/RHEL 7.3 (6.9 also supported) Package List

You can install the EPEL repository using the instructions at https://fedoraproject.org/wiki/EPEL. In addtion, the following packages should be installed with the `yum` install command.

- `ocl-icd`

- ocl-icd-devel
- opencl-headers
- kernel-headers
- kernel-devel
- gcc-c++
- gcc
- gdb
- make
- opencv
- libjpeg-turbo-devel
- libpng12-devel
- python(gdb capability in SDAccel works with python 2.7.5)
- git version 1.8.3.1 or higher
- unzip
- dmidecode
- redhat-lsb
- kernel-headers-$(uname -r)
- compat-libtiff3
- libstdc++-static
- libtiff-devel
- pciutils
- strace
- perl
- boost-devel
- boost-filesystem
- gnuplot
- cmake
- lm_sensors

Ubuntu 16.04 Package List

The following packages should be installed with `apt-get install` command.

- `ocl-icd-libopencl1`
- `opencl-headers`
- `ocl-icd-opencl-dev`
- `linux-headers`
- `linux-libc-dev`
- `g++`
- `gcc`
- `gdb`
- `make`
- `libopencv-core`
- `libjpeg-dev`
- `libpeng-dev`
- `python`
- `git version 1.8.3.1 or higher`
- `dmidecode`
- `lsb`
- `unzip`
- `linux-headers-$(uname -r)`
- `libstdc++ -static`
- `opencv`
- `libtiff5-dev`
- `pciutils`
- `strace`
- `perl`
- `libboost-dev`
- `libboost-filesystem-dev`
- `gnuplot`
- `cmake`

- `lm-sensors`

Obtaining a License on the Xilinx Licensing Site

This section describes the steps to obtain a license for the SDx™ development environment.

1. Sign in to the Xilinx® licensing website: https://www.xilinx.com/getproduct. See the following figure.

   If this is your first time generating a license for the SDAccel™ - Xilinx OpenCL™ Design Environment, contact your Xilinx representative to enable your access to the SDAccel licensing website.

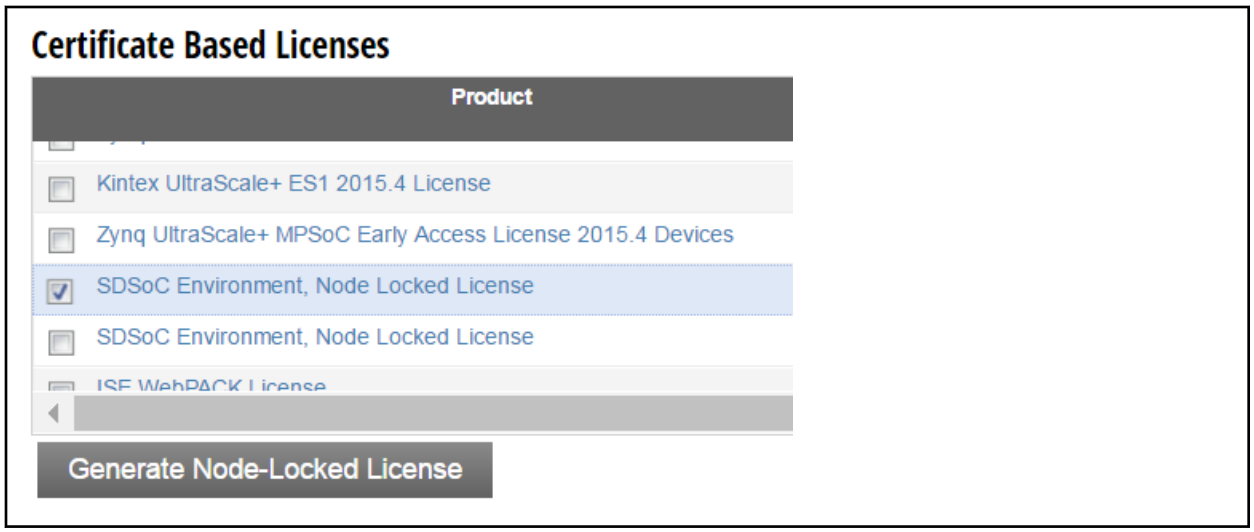
   SDSoC comes with a 60-day evaluation license, so you should be able to see it in your available license list.

2. In the account drop-down menu, select **XILINX - SDSoC Environment** or **SDAccel Environment**.

***Note*:** This only shows up if you have purchased or redeemed an SDSoC™ or SDAccel license.

IMPORTANT!: *If you are interested in SDSoC, you should also see a "SDSoC 60-day evaluation license" for the first time use.*

3. From the Certificate Based Licenses menu, select **SDSoC Environment, Node-Locked License** or **SDAccel Environment, Node-Locked License**.



4. Click **Generate node-locked license**.
5. Enter a Host ID in the **License Generation** screen and click **Next**.
6. Verify that the Host ID for the license is correct and click **Next**.
7. Accept the licensing agreement by clicking **Accept**.

   You will receive an email from xilinx.notification@entitlenow.com with the license file.
8. Set the `XILINXD_LICENSE_FILE` environment variable to point to the location of the license file on your system.

XILINX
ALL PROGRAMMABLE™

Chapter 4

Installing the SDx Environments

This chapter explains the installation process for either the SDSoC™ environment or the SDAccel™ environment.

Preparing to Install the Tools

***Note*:** Before starting installation, you must complete the following steps.

1. Make sure your system meets the requirements described in the following topics:
   - Software Requirements
   - SDSoC Hardware Requirements
   - SDAccel Hardware Requirements

   .
2. Disable anti-virus software to reduce installation time.
3. Close all open programs before you begin installation.

Installing SDSoC and SDAccel

You have two options for installation of SDSoC™ and SDAccel™. Both installation types are available on the Xilinx® Downloads Website.

***Note*:** There are separate installers for SDSoC and SDAccel. When you launch the installer for the product you want to use, the devices are preselected for you.

Using the Web Installer

Using the web installer is recommended.

Using the web installer you can pick and choose what you would like to install up front and that is the only data that will need to get downloaded for installation. Also, in the case of a network failure, you can resume from where you last stopped, instead of starting from the beginning again.

***Note*:** The following devices are pre-selected in the individual installers: - For the SDAccel™-specific web installer, 7 Series, UltraScale™, and UltraScale+™ are pre-selected. - For the SDSoC-specific web installer, Zynq®-7000 and UltraScale+ MPSoC are pre-selected. - For the combined SDx SFD (single file download) image, no devices are pre-selected.

Downloading and Installing the Full Installation File

If you downloaded the full product installation, decompress the file and run xsetup (for Linux) or xsetup.exe (for Windows, not available for SDAccel™) to launch the installation.

If you downloaded the web installer client, launch the downloaded file.You are prompted to log in and use your regular Xilinx login credentials to continue with the installation process.

Xilinx recommends the use of 7-zip or WinZip (v.15.0 or newer) to decompress the downloaded `tar.gz` file.

- The **Download and Install Now** choice allows you to select specific tools and device families on following screens, downloads only the files required to install those selections, and then installs them for you. After entering your login credentials, you can select between a traditional web-based installation or a full image download.
- The **Download Full Image** requires you to select a download destination and to choose whether you want a Windows only, Linux only, or an install that supports both operating systems. There are no further options to choose with the **Download Full Image** selection, and installation needs to be done separately by running the xsetup application from the download directory.

Batch Mode Installation Flow

The installer can run in an unattended batch process. To run unattended, a standard edition and install location must be specified or a configuration file must be present which tells the installer the install location and which of the tools, devices and options you wish to install. The installer has a mode in which it can generate a reference option file for you based on common configurations, which you can further edit to customize your installation.

Xilinx recommends that you generate this reference for each new quarterly release, so that new devices, tools, options or other changes will be accounted for in your options file.

To begin using batch mode, open a command shell and change to the directory where you have stored your extracted installer.

For Windows, open the command window with administrator privileges and run the xsetup.bat file, found in the `\bin` directory, and not `xsetup.exe` with the options below.

Generating a Configuration File

1. Run: `xsetup -b ConfigGen`

   This will put you in an interactive mode where you will see the following menu. Choose the SDx™ IDE for SDSoC™ and SDAccel™ development environments edition.
2. After you make a selection, you will be prompted with the location/filename for your configuration file and the interactive mode exits.

   Below is a sample configuration file:

```
Edition=SDx Development Environments

# Path where Xilinx software will be installed.
Destination=/opt/Xilinx

# Choose the Products/Devices the you would like to install.
Modules=Built-in Platforms and associated devices for SDSoC:1,Zynq
UltraScale+ MPSoC:1,Virtex UltraScale+ HBM ES:0,Zynq-7000:1,Kintex
UltraScale+ ES:0,Kintex UltraScale+:1,ARM Cortex-A53:1,Spartan-7:1,ARM
Cortex-A9:1,ARM Cortex R5:1,Virtex UltraScale+ ES:0,System Generator
for DSP:0,Artix-7:1,Built-in Platforms and associated devices for
SDAccel:1,DocNav:1,Kintex-7:1,Virtex UltraScale+:1,Model
Composer:0,Zynq UltraScale+ RFSoC ES:0,Kintex UltraScale:1,Engineering
Sample Devices for Custom Platforms:0,Virtex UltraScale:1,Zynq
UltraScale+ MPSoC ES:0,MicroBlaze:1,Virtex-7:1

# Choose the post install scripts you'd like to run as part of the
finalization step. Please note that some of these scripts may require
user interaction during runtime.
InstallOptions=Acquire or Manage a License Key:0,Enable WebTalk for SDK
to send usage statistics to Xilinx:1,Enable WebTalk for Vivado to send
usage statistics to Xilinx (Always enabled for WebPACK license):1

## Shortcuts and File associations ##
# Choose whether Start menu/Application menu shortcuts will be created
or not.
CreateProgramGroupShortcuts=1

# Choose the name of the Start menu/Application menu shortcut. This
setting will be ignored if you choose NOT to create shortcuts.
ProgramGroupFolder=Xilinx Design Tools

# Choose whether shortcuts will be created for All users or just the
Current user. Shortcuts can be created for all users only if you run
the installer as administrator.
CreateShortcutsForAllUsers=0
```

```
# Choose whether shortcuts will be created on the desktop or not.
CreateDesktopShortcuts=1

# Choose whether file associations will be created or not.
CreateFileAssociation=1
```

Each option in the configuration file matches a corresponding option in the GUI. A value of `1` means that option is selected; a value of `0` means the option is unselected.

***Note*:** In this configuration file, by default there are no devices selected for installation (all devices have a value of 0). You MUST update a device to a value of 1 in order to install that device.

Running the Installer

Now that you have edited your configuration file to reflect your installation preferences, you are ready to run the installer. As part of the command line installer, you must indicate your acceptance of the Xilinx End-User License Agreement Xilinx End-User License Agreement and Third Party End-User License Agreement Third Party End-User License Agreement, and confirm you understand the WebTalk Terms and Conditions.

WebTalk Terms and Conditions

The WebTalk Terms and Conditions, which you must agree to when running the installer, reads as follows:

By indicating `I AGREE`, I also confirm that I have read Section 13 of the terms and conditions above concerning WebTalk and have been afforded the opportunity to read the WebTalk FAQ posted at https://www.xilinx.com/webtalk. I understand that I am able to disable WebTalk later if certain criteria described in Section 13(c) apply. If they don't apply, I can disable WebTalk by uninstalling the Software or using the Software on a machine not connected to the internet. If I fail to satisfy the applicable criteria or if I fail to take the applicable steps to prevent such transmission of information, I agree to allow Xilinx to collect the information described in Section 13(a) for the purposes described in Section 13(b).

When using the command line, use the command-line switch, `-a` or `--agree`, to indicate your agreement to each of the above. If one of the above is left out of the list, or the agree switch is not specified, the installer exits with an error and does not install.

Batch Mode Installation

This is an example of the command line for a typical new installation using a configuration file.

```
xsetup --agree XilinxEULA,3rdPartyEULA,WebTalkTerms
--batch Install --config install_config.txt
```

If you wish to use one of Xilinx's default Edition configurations, you do not have to specify the `--config` option, but since the destination directory is included in the configuration file, you will be required to specify this on the command line.

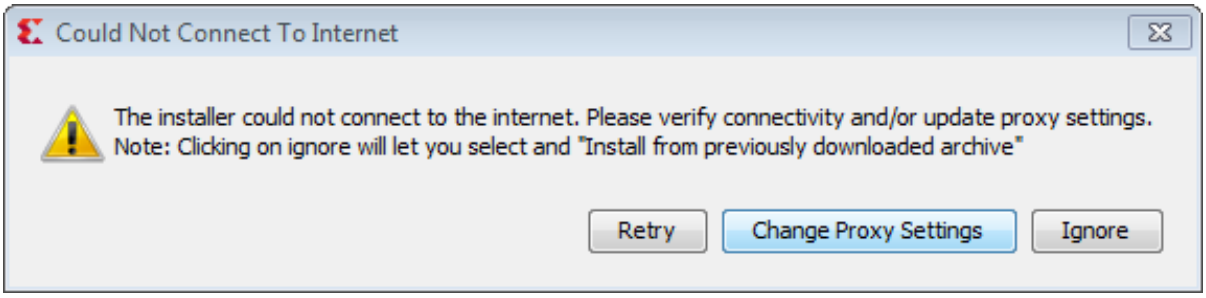
```
xsetup --agree 3rdPartyEULA,WebTalkTerms,XilinxEULA
--batch Install --edition "SDx Development Environments" --location
"C:\Xilinx"
```

The above command uses the default configuration options for the edition specified. To see the default configuration options, use the `-b ConfigGen` mode as described above. The batch mode of the SDx installer can also perform uninstallation and upgrades (adding additional tools and devices). For the full list of the options in the installer batch mode run `xsetup -h or xsetup --help`.

Verifying Connectivity

The installer connects to the Internet through the system proxy settings in Windows. These settings can be found under **Control Panel** > **Network and Internet** > **Internet Options**. For Linux users, the installer uses Firefox browser proxy settings (when explicitly set) to determine connectivity.

Figure 1: **Vivado Design Suite Installation - Connectivity**



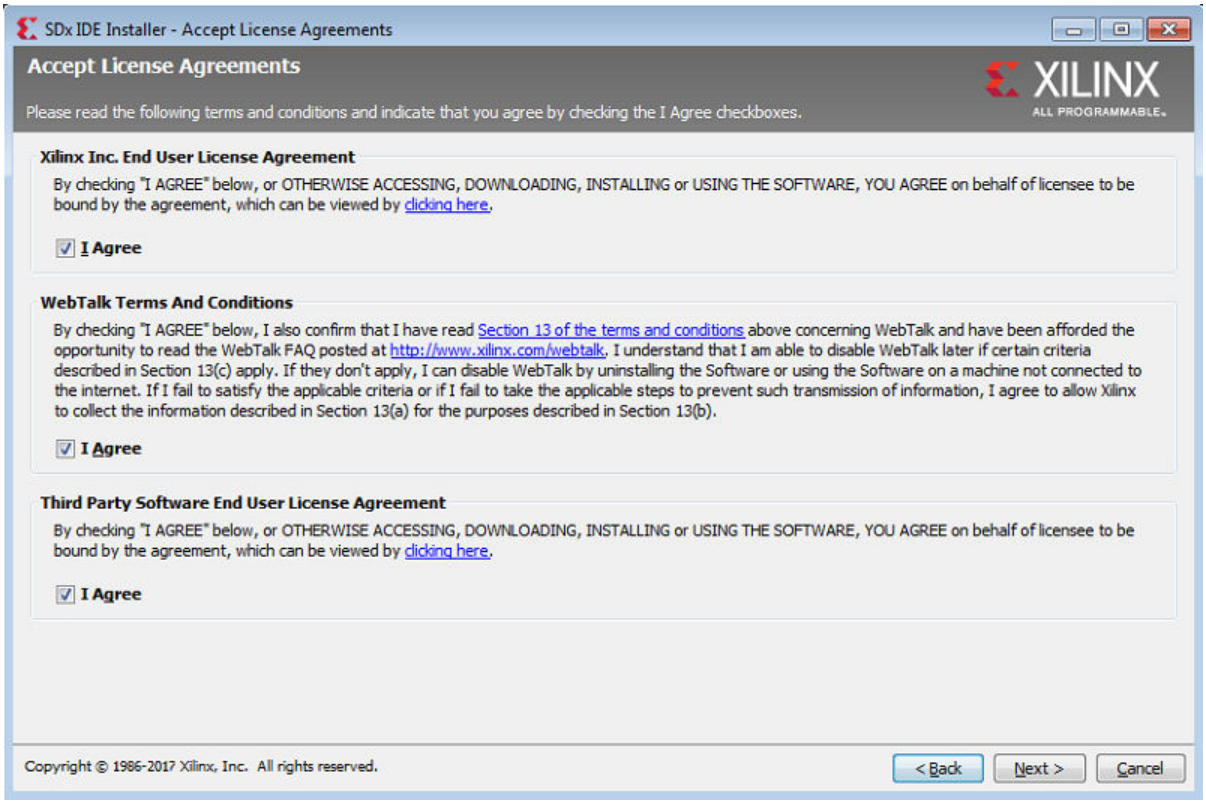
If there are connectivity issues, verify the following:

1. If you are using alternate proxy settings to the ones referred to, select the **Manual Proxy Configuration** option to specify the settings.
2. Check whether your company firewall requires a proxy authentication with a user name and password. If so, select the **Manual Proxy Configuration** option in the dialog box above.
3. For Linux users, if either the **Use System settings** or the **Auto detect settings** option is selected in the Firefox browser, you must manually set the proxy in installer.

Accepting License Agreements

Carefully read the license agreements before continuing with the installation. If you do not agree to the terms and conditions, cancel the installation and contact Xilinx.

Figure 2: **License Agreements**

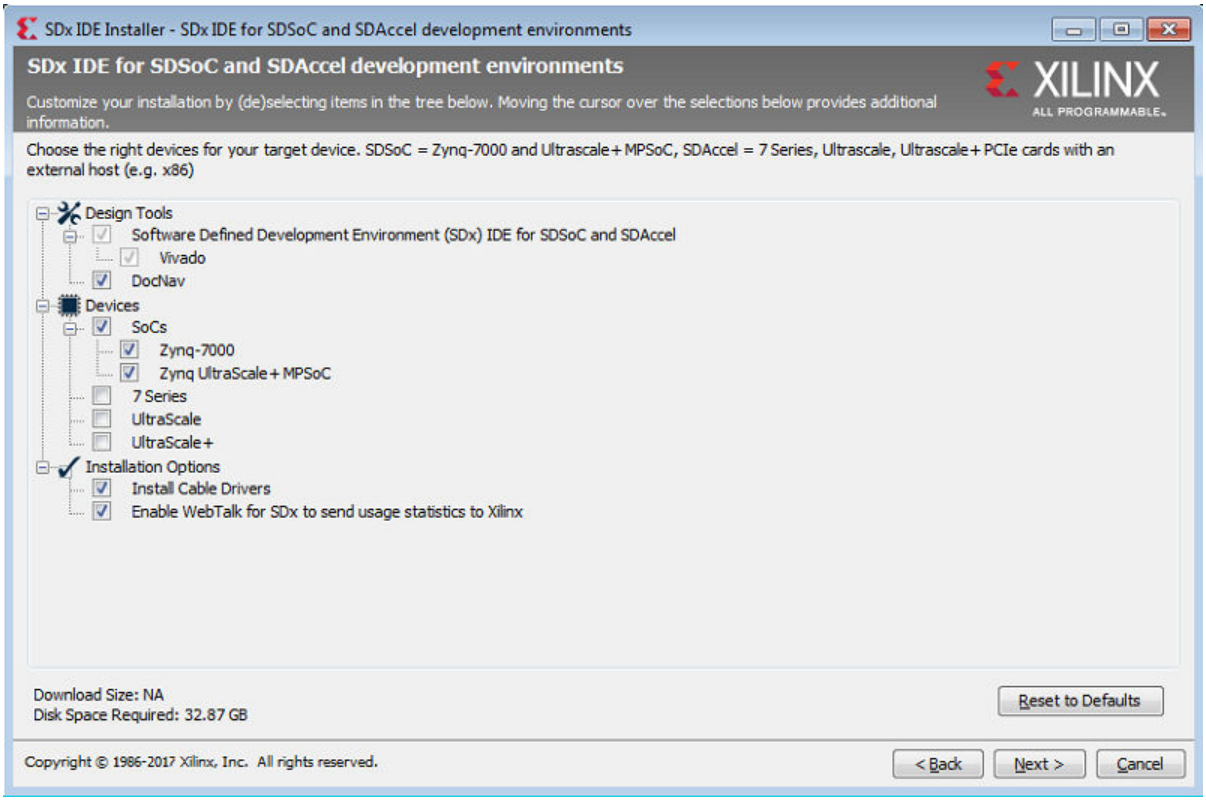


Selecting Tool and Device Options

Customize the installation by choosing the design tools, device families and installation options. Selecting only what you need helps to minimize the time taken to download and install the product. You will be able to add to this installation later by clicking **Add Design Tools or Devices** from either the operating system Start Menu or the **Vivado** > **Help** menu.

When you launch the installer for the product you want to use, the devices are preselected for you.

Figure 3: **Design Tools and Device Options**



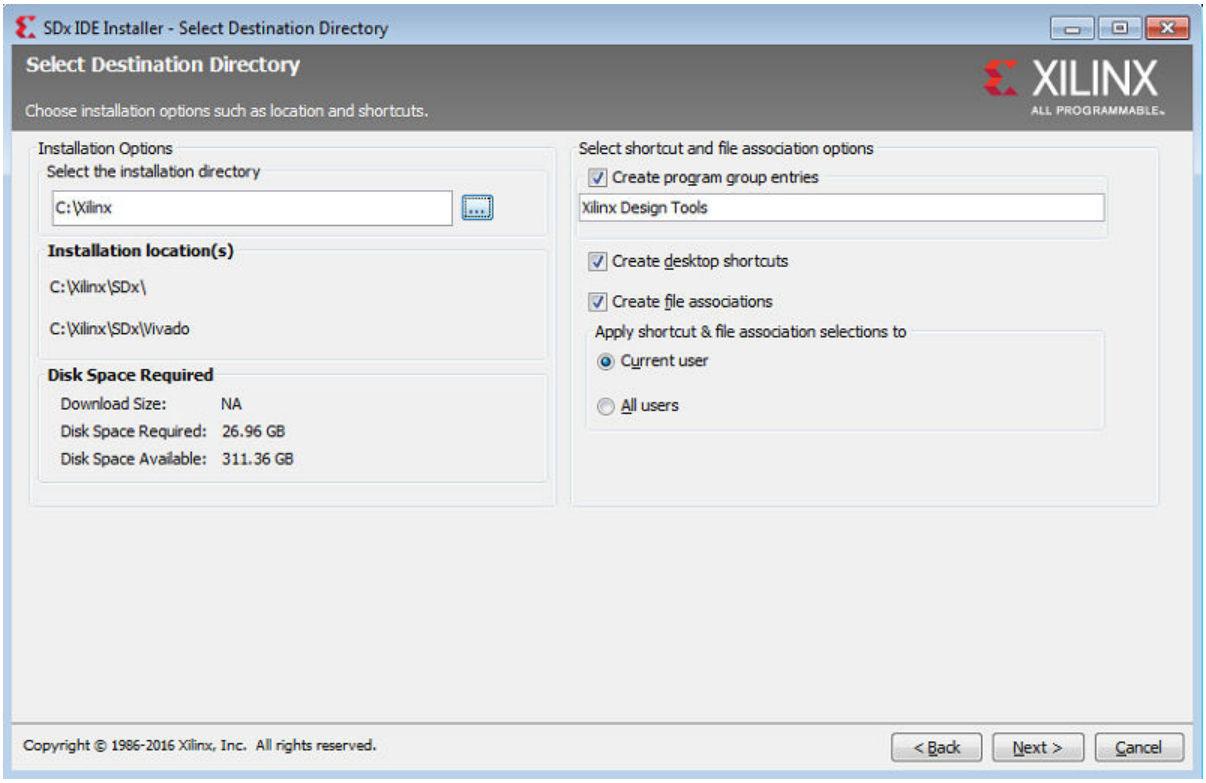
Setting Destination Directory and Installation Options

Define the installation directory for the software, as shown in the following figure.

***Note*:** The installation directory name must not contain any spaces in any part of the directory path.

You can customize the creation of the program group entries (Start Menu) and the creation of desktop shortcuts. The shortcut creation and file association options can be applied to the current user or all users.

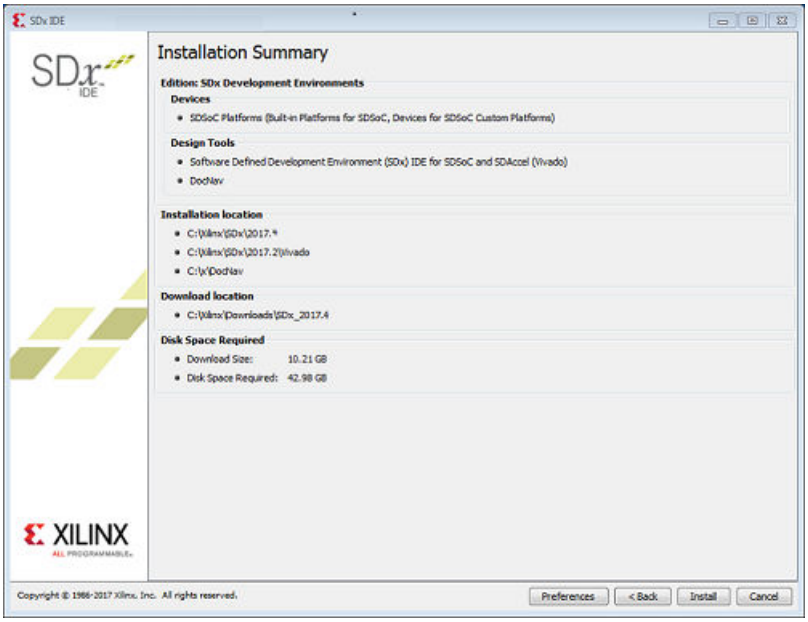
Figure 4: **Destination Directory and Installation Options**



Reviewing the Installation Details

Review the installation details shown in the **Installation Summary** screen.

Figure 5: **Installation Summary**



When you click **Install**, the installation process takes several minutes to complete.

Setting Up the Environment to Run SDx

1. To set up the environment to run SDx, source the file below so that `sdx` command is in the `PATH`:

```
C Shell: source <SDX_INSTALL_DIR>/settings64.csh
Bash: source <SDX_INSTALL_DIR>/settings64.sh
```

XILINX
ALL PROGRAMMABLE™

Appendix A

Additional Resources and Legal Notices

Xilinx Resources

For support resources such as Answers, Documentation, Downloads, and Forums, see Xilinx Support.

Solution Centers

See the Xilinx Solution Centers for support on devices, software tools, and intellectual property at all stages of the design cycle. Topics include design assistance, advisories, and troubleshooting tips.

References

These documents provide supplemental material useful with this webhelp:

SDAccel Documents

1. *SDAccel Environment User Guide* (UG1023)
2. *SDAccel Environment Optimization Guide* (UG1207)
3. *SDAccel Environment Tutorial: Introduction* (UG1021)
4. *SDAccel Environment Platform Development Guide* (UG1164)

SDSoC Documents

1. *SDSoC Environment User Guide* (UG1027)
2. *SDSoC Environment Optimization Guide* (UG1235)
3. *SDSoC Environment Tutorial: Introduction* (UG1028)
4. *SDSoC Environment Platform Development Guide* (UG1146)

Additional Documents

1. *SDx Pragma Reference Guide* (UG1253)
2. *Xilinx OpenCV User Guide* (UG1233)
3. *Platform Cable USB II Data Sheet* (DS593)

More Resources

1. Xilinx® licensing website: https://www.xilinx.com/getproduct
2. SDSoC Developer Zone: https://www.xilinx.com/products/design-tools/software-zone/sdsoc.html.
3. SDAccel Developer Zone: https://www.xilinx.com/products/design-tools/software-zone/sdaccel.html
4. *Xilinx End-User License Agreement* (UG763)
5. *Third Party End-User License Agreement* (UG1254)

Documentation Navigator and Design Hubs

Xilinx® Documentation Navigator provides access to Xilinx documents, videos, and support resources, which you can filter and search to find information. To open the Xilinx Documentation Navigator (DocNav):

- From the Vivado® IDE, select **Help > Documentation and Tutorials**.
- On Windows, select **Start > All Programs > Xilinx Design Tools > DocNav**.
- At the Linux command prompt, enter `docnav`.

Xilinx Design Hubs provide links to documentation organized by design tasks and other topics, which you can use to learn key concepts and address frequently asked questions. To access the Design Hubs:

- In the Xilinx Documentation Navigator, click the **Design Hubs View** tab.
- On the Xilinx website, see the Design Hubs page.

For more information on Documentation Navigator, see the Documentation Navigator page on the Xilinx website.

Please Read: Important Legal Notices

The information disclosed to you hereunder (the "Materials") is provided solely for the selection and use of Xilinx products. To the maximum extent permitted by applicable law: (1) Materials are made available "AS IS" and with all faults, Xilinx hereby DISCLAIMS ALL WARRANTIES AND CONDITIONS, EXPRESS, IMPLIED, OR STATUTORY, INCLUDING BUT NOT LIMITED TO WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, OR FITNESS FOR ANY PARTICULAR PURPOSE; and (2) Xilinx shall not be liable (whether in contract or tort, including negligence, or under any other theory of liability) for any loss or damage of any kind or nature related to, arising under, or in connection with, the Materials (including your use of the Materials), including for any direct, indirect, special, incidental, or consequential loss or damage (including loss of data, profits, goodwill, or any type of loss or damage suffered as a result of any action brought by a third party) even if such damage or loss was reasonably foreseeable or Xilinx had been advised of the possibility of the same. Xilinx assumes no obligation to correct any errors contained in the Materials or to notify you of updates to the Materials or to product specifications. You may not reproduce, modify, distribute, or publicly display the Materials without prior written consent. Certain products are subject to the terms and conditions of Xilinx's limited warranty, please refer to Xilinx's Terms of Sale which can be viewed at https://www.xilinx.com/legal.htm#tos; IP cores may be subject to warranty and support terms contained in a license issued to you by Xilinx. Xilinx products are not designed or intended to be fail-safe or for use in any application requiring fail-safe performance; you assume sole risk and liability for use of Xilinx products in such critical applications, please refer to Xilinx's Terms of Sale which can be viewed at https://www.xilinx.com/legal.htm#tos.

AUTOMOTIVE APPLICATIONS DISCLAIMER

AUTOMOTIVE PRODUCTS (IDENTIFIED AS "XA" IN THE PART NUMBER) ARE NOT WARRANTED FOR USE IN THE DEPLOYMENT OF AIRBAGS OR FOR USE IN APPLICATIONS THAT AFFECT CONTROL OF A VEHICLE ("SAFETY APPLICATION") UNLESS THERE IS A SAFETY CONCEPT OR REDUNDANCY FEATURE CONSISTENT WITH THE ISO 26262 AUTOMOTIVE SAFETY STANDARD ("SAFETY DESIGN"). CUSTOMER SHALL, PRIOR TO USING OR DISTRIBUTING ANY SYSTEMS THAT INCORPORATE PRODUCTS, THOROUGHLY TEST SUCH SYSTEMS FOR SAFETY PURPOSES. USE OF PRODUCTS IN A SAFETY APPLICATION WITHOUT A SAFETY DESIGN IS FULLY AT THE RISK OF CUSTOMER, SUBJECT ONLY TO APPLICABLE LAWS AND REGULATIONS GOVERNING LIMITATIONS ON PRODUCT LIABILITY.