

SDAccel Development Environment Tutorial

Getting Started

UG1021 (v2016.2) August 31, 2016

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
08/31/2016	2016.2	Tutorials updated to use Makefile methodology.
05/13/2016	2016.1	No technical changes.

Table of Contents

Revision History	2
Table of Contents	3
Chapter 1: Introduction	4
Objectives	4
Design Overview	4
Design Files.....	5
Chapter 2: Compiling and Optimizing the Application	6
Compiling and Running the Baseline Application	7
System Estimate Report	7
Compiling and Running the Application	8
Appendix A: Additional Resources and Legal Notices	10
Xilinx Resources	10
Solution Centers	10
Please Read: Important Legal Notices.....	10

Introduction

This document describes how to use the SDAccel™ development environment to compile and optimize an example design and then download and run the design on acceleration boards based on either the Kintex® UltraScale, Virtex®-7, or Kintex-7 FPGA.

The SDAccel development environment for OpenCL™, C, and C++ applications enables concurrent programming of the system processor and the FPGA logic and requires no RTL design experience. The example design is captured as a host program written in C or C++ and a set of computation kernels expressed in C, C++, or the OpenCL C language.

Objectives

This tutorial:

- Introduces the use of the SDAccel™ development environment to create OpenCL™ programs for Kintex® UltraScale, Virtex®-7, or Kintex-7 FPGAs.
- Provides a specific procedure for compiling and optimizing an algorithm for acceleration cards featuring Kintex UltraScale, Virtex-7, or Kintex-7 FPGAs.

After completing this tutorial, you will be able to compile and optimize a Smith-Waterman sequence alignment algorithm.

Design Overview

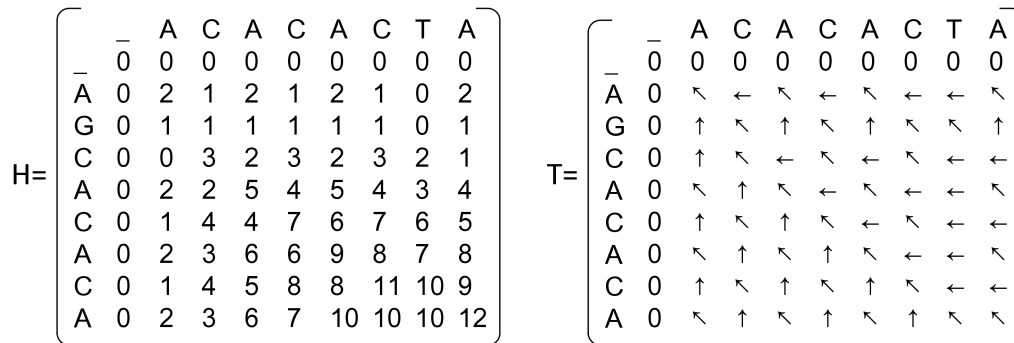
The example design is based on the Smith-Waterman algorithm which is a database search algorithm developed by T.F. Smith and M.S. Waterman, and is based on the earlier Needleman and Wunsch algorithm. The objective of the Smith-Waterman algorithm is to take two sequences of data of arbitrary length and determine the best possible alignment between these sequences. The alignment is determined by scoring matches and mismatches in character-by-character traversal of both sequences. Based on the resulting cost matrix, the Smith-Waterman algorithm determines the alignment and maximum alignment length of a sequence pair. The mathematical foundation behind Smith-Waterman is given by the function

$$H_{ij} = \max\{H_{i-1, j-1} + s(a_i, b_j); H_{i-k, j} - W_k; H_{i, j-1} - W_1; 0\}$$

An example of this algorithm shown in the following figure:

Figure 1–1: Smith-Waterman Algorithm

- Sequence 1 = ACACACTA
- Sequence 2 = AGCACACA
- $S(a,b)=+2$ if $a = b$ (match), -1 if $a \neq b$ (mismatch)
- $W_i=-i$ /



X14998-090415

For the pair of sequences shown, the sequence alignment is determined by the two resulting matrices from the Smith-Waterman algorithm. The cost matrix, denoted as matrix H, provides the endpoint of the aligned sequence. The alignment endpoint is the maximum value of matrix H.

Matrix T, which is defined as the traversal matrix, defines how the algorithm needs to trace back through matrix H until the first point in the aligned sequence is reached. The first point in alignment is point at which the back tracing methodology encounters a cell with a value of 0. The alignment results for the sequence are:

- Sequence 1 = A-CACACTA
- Sequence 2 = AGCACAC-A

Design Files

Example design files are included with the SDAccel Development Environment. Copy the Getting Started example from `<SDACCEL_INSTALLATION_PATH>/examples/getting_started` to your working directory.

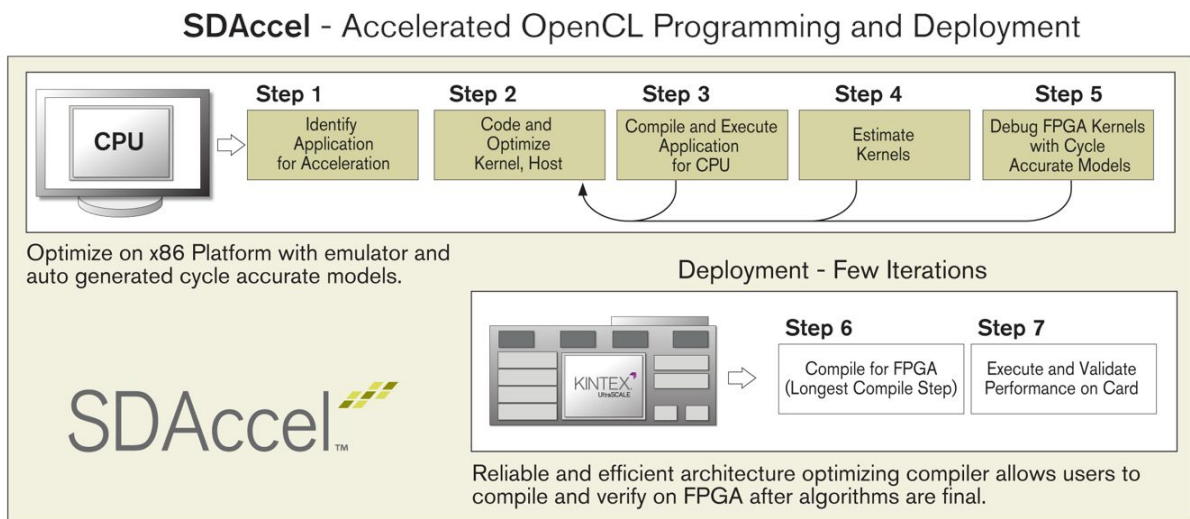


IMPORTANT: SDAccel software and board installation must be completed before continuing the tutorial. Please see SDAccel Development Environment User Guide: Installation and Licensing, (UG1020) for more details.

Compiling and Optimizing the Application

The SDAccel™ development environment enables a programmer to quickly iterate through changes in an OpenCL™ host code and kernels to arrive at an optimized version targeted at a specific board. The code and optimize iteration loop shown in the following figure captures the design methodology behind OpenCL after the programmer has completed the functionality of the application.

Figure 2–1: SDAccel Application Programming and Optimization Design Flow



To compile an application using the SDAccel flow compiler:

1. Compile the application without any hints to the compiler and analyze the resulting performance. This step assumes that the programmer has decided on the functionality of the target application and has run the program on a CPU to check for correctness.
2. Optimize the application by adding attributes to the kernel code. A list of supported attributes is available in the *SDAccel Development Environment User Guide: Features and Development Flows* ([UG1023](#)).
3. Instantiate multiple copies of a kernel in the FPGA. The SDAccel flow compiler can compile versions of the same application with 1 or N copies of a kernel running on the board. The application programmer determines how many copies of a kernel to run in parallel on the board, and provides this information as part of the SDAccel command script.
4. Run the application on the board.

The files used in compiling the application locally are located at:

```
<user directory>/getting_started
```

Compiling and Running the Baseline Application

The SDAccel™ development environment can be driven in three modes: GUI, Makefile with Xilinx OpenCL™ Compiler (xocc) compiler, and script mode. This tutorial uses Makefile with xocc for compiling and running applications.

To generate the baseline compilation of the application using the SDAccel environment, execute the following commands:

```
cd baseline
make -f sdaccel.mk estimate
run_cpu_em
```

The specified Makefile targets carry out the following tasks:

1. Define the files for the host code application.
2. Define the kernel and associated source files.
3. Define the device that the kernel is targeting.
4. Generate the system estimate report for the kernel.
5. Compile the application for CPU emulation.
6. Run the CPU emulation of the host code and the kernel.

Upon successful execution of the application, the following output appears.

```
INFO: [XOCC 60-585] Compiling for software emulation target
INFO: [XOCC 60-423] Target device: xilinx:adm-pcie-7v3:1ddr:3.0
INFO: [XOCC 60-242] Creating kernel: 'smithwaterman'
INFO: [XOCC 60-594] Finished kernel compilation
INFO: [XOCC 60-586] Created bin_smithwaterman_cpu_emu.xclbin
XCL_EMULATION_MODE=true ./baseline -k bin_smithwaterman_cpu_emu.xclbin

Input sequence1: TAGGCAAGACCACTTTAGCATGGTCTACAACGCCTAGACCTTTGGCAAAGCAGATCGGCCCGCCCATCACTAGTGGGACTATCC
Input sequence2: TAATGGGAACACCTGCTGCAATCGGATCGTTGCAGCGGTAATGTGTGCGGTATATGCGAGTAGGGTAATCCAAACGTCCCATTGC

Platform = Xilinx
Device = xilinx:adm-pcie-7v3:1ddr:3.0
OpenCL Version = 1.0
Loading bin_smithwaterman_cpu_emu.xclbin
Global size = 1
Local size = 1

Align sequence1: T-A-GGCAAGACCACT-TTAGC-AT-GG-TC--TACAACGCCTAGACCT-T-T-GGCA-AAGCAGA-T-CGG----CC---CG-CCCAT
Align sequence2: TAATGGGAACA-C-CTGCT-GCAATCGGATCGTTGCAGCG-GTA-A--TGTGTCGGTATATGC-GAGTAGGGTAATCCAAACGTCCCATT

OpenCL kernel time: 0 sec
PASSED TEST
```

System Estimate Report

The system estimate report shows the FPGA resource utilization as well as the start interval for the hardware block implementing the Smith-Waterman algorithm. The most important number for performance is the start interval, which determines the number of clock cycles between consecutive executions of a kernel. The range shown in the report is a good indication that this kernel can be further optimized.

Following is the system estimate report, `system_estimate.txt` for this application:

```

=====
Version:      sdaccel v2016.2
              (64-bit)
Build:        SW Build (by ) on
Copyright:    Copyright 1986-2016 Xilinx, Inc. All Rights Reserved.
=====

-----
Design Name:   _xocc_compile_kernel_bin.dir
Target Device: xilinx:adm-pcie-7v3:1ddr:3.0
Target Clock:  200MHz
-----

-----
Kernel Summary

Total number of kernels: 1

+-----+-----+-----+-----+-----+
| Kernel Name      | Type   | Target                | OpenCL Library | Compute Units |
+-----+-----+-----+-----+-----+
| smithwaterman    | clc    | fpga0:OCL_REGION_0   | xcl_xocc       | 1              |
+-----+-----+-----+-----+-----+

-----
OpenCL Binary = xcl_xocc

Kernels mapped to = clc_region

Timing Information (MHz)
+-----+-----+-----+-----+-----+
| Compute Unit     | Kernel Name      | Target Frequency     | Estimated Frequency |
+-----+-----+-----+-----+-----+
| smithwaterman_1 | smithwaterman    | 200                   | 204.499             |
+-----+-----+-----+-----+-----+

Latency Information (clock cycles)
+-----+-----+-----+-----+-----+-----+
| Compute Unit     | Kernel Name      | Start Interval       | Best Case           | Avg Case            | Worst Case          |
+-----+-----+-----+-----+-----+-----+
| smithwaterman_1 | smithwaterman    | 29467                 | 29466                | 29466                | 29466                |
+-----+-----+-----+-----+-----+-----+

Area Information
+-----+-----+-----+-----+-----+-----+
| Compute Unit     | Kernel Name      | FF                    | LUT                  | DSP                 | BRAM                 |
+-----+-----+-----+-----+-----+-----+
| smithwaterman_1 | smithwaterman    | 2857                  | 4295                  | 1                   | 11                   |
+-----+-----+-----+-----+-----+-----+

```

Compiling and Running the Application

After the application performance has been tuned, it is time to compile the application to run on the selected board.

1. To compile the application to run on the board, execute the following:

```
make -f sdaccel.mk xbin_hw
```



IMPORTANT: *The compile time to generate binaries for the FPGA board will take at least 30 minutes.*

After the SDAccel™ development environment finishes the compilation process, the output binary `bin_smithwaterman_hw.xclbin` is generated in the current working directory.

2. Follow the instructions below to run the host application and FPGA binary on a system with the target FPGA acceleration card:



IMPORTANT: *Programming of the flash on the FPGA card and installation of driver, firmware and runtime libraries for the target DSA must be performed before completing this step. Please see SDAccel Development Environment User Guide: Installation and Licensing, ([UG1020](#)) for more details.*

- a. Source the `setup.sh` (Bash) or `setup.csh` (Csh/Tcsh) script file generated by the `xbinst` utility. See *SDAccel Development Environment User Guide: Installation and Licensing*, ([UG1020](#)) for more details. The following is the command line example for Bash:

```
source setup.sh
```

- b. Run the application:

```
./baseline -k bin_smithwaterman_hw.xclbin
```

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

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 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 www.xilinx.com/legal.htm#tos.

© Copyright 2016 Xilinx, Inc. Xilinx, the Xilinx logo, Artix, ISE, Kintex, Spartan, Virtex, Vivado, Zynq, and other designated brands included herein are trademarks of Xilinx in the United States and other countries. OpenCL and the OpenCL logo are trademarks of Apple Inc. used by permission by Khronos. PCI, PCIe and PCI Express are trademarks of PCI-SIG and used under license. All other trademarks are the property of their respective owners.

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.