# SDAccel Development Environment Tutorial

## *Getting Started*

**UG1021 (2015.4) February 16, 2016**

XILINX
ALL PROGRAMMABLE™

# Revision History

The following table shows the revision history for this document.

| Date | Version | Revision |
|---|---|---|
| 02/16/2016 | 2015.4 | Moved the board and driver installation to UG1020: SDAccel Development Environment User Guide: Installation and Licensing. |
| 10/22/2015 | 2015.3 | Restructured tutorial. Added information for the Alpha Data ADM-PCIE-KU3 card. |
| 06/29/2015 | 2015.1.2 | Updates to firmware location and settings |
| 05/26/2015 | 2015.1 | DRC Check updates + more |

# Table of Contents

XILINX
ALL PROGRAMMABLE™

Chapter 1

# 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

Hij = max{Hi–1, j–1 + s(ai, bj); Hi–k, j – Wk; Hi, j–1 – W1; 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* $a = b$ *(mismatch)*
- $W_i=-i$

H=

| | _ | A | C | A | C | A | C | T | A |
|---|---|---|---|---|---|---|---|---|---|
| _ | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| A | 0 | 2 | 1 | 2 | 1 | 2 | 1 | 0 | 2 |
| G | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 |
| C | 0 | 0 | 3 | 2 | 3 | 2 | 3 | 2 | 1 |
| A | 0 | 2 | 2 | 5 | 4 | 5 | 4 | 3 | 4 |
| C | 0 | 1 | 4 | 4 | 7 | 6 | 7 | 6 | 5 |
| A | 0 | 2 | 3 | 6 | 6 | 9 | 8 | 7 | 8 |
| C | 0 | 1 | 4 | 5 | 8 | 8 | 11 | 10 | 9 |
| A | 0 | 2 | 3 | 6 | 7 | 10 | 10 | 10 | 12 |

T=

| | _ | A | C | A | C | A | C | T | A |
|---|---|---|---|---|---|---|---|---|---|
| _ | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| A | 0 | ↖ | ← | ↖ | ← | ↖ | ← | ← | ↖ |
| G | 0 | ↑ | ↖ | ↑ | ↖ | ↑ | ↖ | ↖ | ↑ |
| C | 0 | ↑ | ↖ | ← | ↖ | ← | ↖ | ← | ← |
| A | 0 | ↖ | ↑ | ↖ | ← | ↖ | ← | ← | ↖ |
| C | 0 | ↑ | ↖ | ↑ | ↖ | ← | ↖ | ← | ← |
| A | 0 | ↖ | ↑ | ↖ | ↑ | ↖ | ← | ← | ↖ |
| C | 0 | ↑ | ↖ | ↑ | ↖ | ↑ | ↖ | ← | ← |
| A | 0 | ↖ | ↑ | ↖ | ↑ | ↖ | ↑ | ↖ | ↖ |

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

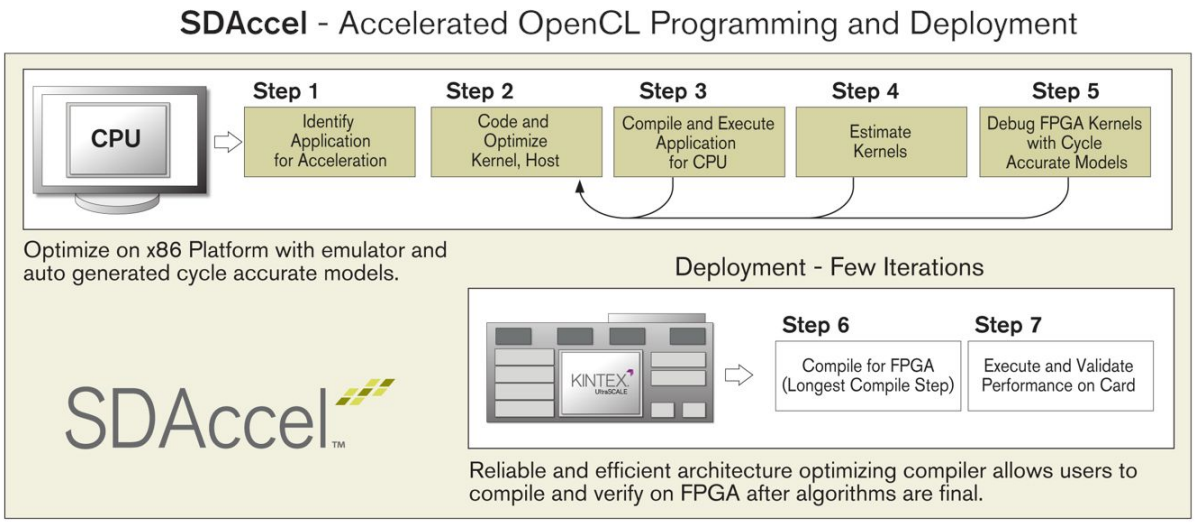
Example design files are included with the SDAccel Development Environment. Copy the Getting Started example from `<SDACCEL_INSTALLATION_PATH>/examples/getting_started` to your working directory.

**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/alpha_data
```

# Compiling and Running the Baseline Application

The SDAccel™ development environment can be driven in three modes: GUI, Makefile with Xilinx OpenCL compiler, and script mode. This tutorial uses script mode and the script for basic compilation of the Getting Started example is `baseline.tcl`.

To generate the baseline compilation of the application using the SDAccel environment, execute the following command: `sdaccel baseline.tcl`

The commands in `baseline.tcl` carry out the following tasks:

1. Define the directory where all tool output is stored.
2. Select the device where the application is run.
3. Define the files for the host code application.
4. Define the kernels and associated source files.
5. Compile the application for emulation on the development machine.
6. Execute the application in the SDAccel emulation environment.

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

```
INFO: [SDAccel 60-348] Executing cpu emulation using software accelerators...
INFO: [SDAccel 60-280] Additional args : '-d acc -k test.xclbin'
INFO: [SDAccel 60-174] Running emulation command line: /proj/baseline_solution/impl/sim/ \
   cpu_em/baseline_solution.exe -d acc -k test.xclbin

Input sequence1: TAGGCAAGACCACTTTAGCATGGTCTACAACGCCTAGACCTTTGGCAAAGCAGATCGGCCCGCCCATCACT \
   AGTGGGACTATCC
Input sequence2: TAATGGGAACACCTGCTGCAATCGGATCGTTGCAGCGGTAATGTGTCGGTATATGCGAGTAGGGTAATCCA \
   AACGTCCCATTGC

Platform = Xilinx
Device = fpga0
OpenCL Version = 1.0
Loading test.xclbin
Global size = 1
Local size = 1

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

OpenCL kernel time: 0 sec
PASSED TEST
INFO: [SDAccel 60-349] Executing cpu emulation using software accelerators...COMPLETE
```

# 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 for the baseline version of this application in `baseline_solution/rpt` directory.

```
================================================
Version:    sdaccel v2015.4 (64-bit)
Build:      SW Build 1462041 on Mon Jan 25 16:35:05 MST 2016
Copyright:  Copyright 1986-2016 Xilinx, Inc. All Rights Reserved.
================================================================================

--------------------------------------------------------------------------------
Design Name:      baseline_solution
Target Device:    xilinx:adm-pcie-7v3:1ddr:2.1
Target Clock:     200MHz
--------------------------------------------------------------------------------

--------------------------------------------------------------------------------
Kernel Summary

Total number of kernels: 1

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

--------------------------------------------------------------------------------
OpenCL Binary = test

Kernels mapped to = clc_region

Timing Information (MHz)
+----------------------+---------------------+--------------------------+--------------------------+
| Compute Unit         | Kernel Name         | Target Frequency         | Estimated Frequency      |
+----------------------+---------------------+--------------------------+--------------------------+
| K1                   | smithwaterman       | 200                      | 204.499                  |
+----------------------+---------------------+--------------------------+--------------------------+

Latency Information (clock cycles)
+----------------+-----------------+------------------+--------------+------------+-------------+
| Compute Unit   | Kernel Name     | Start Interval   | Best Case    | Avg Case   | Worst Case  |
+----------------+-----------------+------------------+--------------+------------+-------------+
| K1             | smithwaterman   | 166424 ~ 202124  | 166423       | 187843     | 202123      |
+----------------+-----------------+------------------+--------------+------------+-------------+

Area Information
+----------------------+----------------------+------------+------------+------------+-------------+
| Compute Unit         | Kernel Name          | FF         | LUT        | DSP        | BRAM        |
+----------------------+----------------------+------------+------------+------------+-------------+
| K1                   | smithwaterman        | 2095       | 3180       | 1          | 11          |
+----------------------+----------------------+------------+------------+------------+-------------+
--------------------------------------------------------------------------------
```

# Optimizing the Kernel Using Attributes

Attributes are the way a programmer can influence the compiler and optimize an application without having to change the application code. Attributes supported by the SDAccel™ development environment are described in detail in the *SDAccel Development Environment User Guide* (UG1023).

***NOTE:*** For information on pipelining, see Loop Pipelining.

The command script for compiling the application after the addition of the code attribute is identical to the baseline script. Because this optimization is defined in the application source code, there is no need for additional commands in the SDAccel environment.

1. Go to the directory that contains the example files: `cd <user directory>/getting_started/alpha_data`
2. To compile this version of the application, execute: `sdaccel pipelined.tcl`

The following system estimate report for this compilation run demonstrates the results of the pipeline attribute. After this optimization is applied, the resulting compute unit is seven times faster than the baseline run in `baseline_solution/rpt directory`.

```
================================================================
Version:    sdaccel v2015.4 (64-bit)
Build:      SW Build 1462041 on Mon Jan 25 16:35:05 MST 2016
Copyright:  Copyright 1986-2016 Xilinx, Inc. All Rights Reserved.
================================================================================

--------------------------------------------------------------------------------
Design Name:      pipelined_solution
Target Device:    xilinx:adm-pcie-7v3:1ddr:2.1
Target Clock:     200MHz
--------------------------------------------------------------------------------

--------------------------------------------------------------------------------
Kernel Summary

Total number of kernels: 1

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

--------------------------------------------------------------------------------
OpenCL Binary = test

Kernels mapped to = clc_region

Timing Information (MHz)
+--------------------+--------------------------+---------------------+------------------------+
| Compute Unit       | Kernel Name              | Target Frequency    | Estimated Frequency    |
+--------------------+--------------------------+---------------------+------------------------+
| K1                 | smithwaterman            | 200                 | 204.499                |
+--------------------+--------------------------+---------------------+------------------------+

Latency Information (clock cycles)
+----------------+---------------+-----------------+------------+-------------+---------------+
| Compute Unit   | Kernel Name   | Start Interval  | Best Case  | Avg Case    | Worst Case    |
+----------------+---------------+-----------------+------------+-------------+---------------+
| K1             | smithwaterman | 21818           | 21817      | 21817       | 21817         |
+----------------+---------------+-----------------+------------+-------------+---------------+

Area Information
+--------------------+----------------------+------------+------------+------------+-----------+
| Compute Unit       | Kernel Name          | FF         | LUT        | DSP        | BRAM      |
+--------------------+----------------------+------------+------------+------------+-----------+
| K1                 | smithwaterman        | 2428       | 3430       | 1          | 11        |
+--------------------+----------------------+------------+------------+------------+-----------+
--------------------------------------------------------------------------------
```

## Loop Pipelining

Loop pipelining is a user controlled attribute that allows the compiler to modify the scheduling of operations to enable loop iterations inside the kernel code to run in parallel on the same compute unit. Following is the modified code with the loop pipeline attribute.

```
#ifdef __xilinx__
    __attribute__((xcl_pipeline_loop))
#endif
    for (int i = 1; i < N; i++) {
        localS1[i] = s1[i];
    }
#ifdef __xilinx__
    __attribute__((xcl_pipeline_loop))
#endif
    for (int i = 1; i < N; i++) {
        localS2[i] = s2[i];
    }
#ifdef __xilinx__
    __attribute__((xcl_pipeline_loop))
#endif
    for (int i = 0; i < N * N; i++) {
        localMatrix[i] = 0;
    }

#ifdef __xilinx__
    __attribute__((xcl_pipeline_loop))
#endif
    for (short index = N; index < N * N; index++)
    {
        short dir = CENTER;
        short val = 0;
        short j = index % N;
        if (j == 0) { // Skip the first column
            west = 0;
            northwest = 0;
            continue;
        }
        short i = index / N;
        short2 temp = localMatrix[index - N];
        north = temp.x;
        const short match = (localS1[j] == localS2[i]) ? MATCH : MISS_MATCH;
        short val1 = northwest + match;

        if (val1 > val) {
            val = val1;
            dir = NORTH_WEST;
        }
```

# Compiling and Running the Optimized Application

After the application performance has been tuned, it is time to compile the application to run on the selected board. The command script for this compilation run adds the following functionality:

- The `build_system` command invokes the generation of compute unit specific hardware for execution in the FPGA logic.
- The `package_system` command readies the compiled application executables and binaries to be run on the board.

1. To compile the application to run on the board, execute the following: `sdaccel board_compile.tcl`

**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 binaries are placed in the following directory: `board_compilation_solution/pkg/pcie`

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. Go into the `board_compilation_solution/pkg/pcie` directory:

      ```
      cd board_compilation_solution/pkg/pcie
      ```

   b. 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
      ```

   c. Run the application:

      ```
      ./board_compilation_solution.exe -k test.xclbin
      ```

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

## Please Read: Important Legal Notices