SDAccel Development Environment

Tutorial

UG1021 (v2015.1) September 15, 2015

XILINX
ALL PROGRAMMABLE™

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
09/15/2015	2015.1.5	Updates to lab instructions
06/29/2015	2015.1.2	Updates to firmware location and settings
05/26/2015	2015.1	DRC Check updates + more

Table of Contents

Introduction

SDAccel™ is a development environment for OpenCL applications targeting PCIe® based Virtex®-7, and Kintex®-7 FPGA accelerator cards. This environment enables concurrent programming of the system processor and the FPGA logic without the need for RTL design experience. The application is captured as a host program written in C/C++ and a set of computation kernels expressed in C, C++, or the OpenCL C language.

Objectives

This tutorial:

- Show you how to take advantage of OpenCL programs into a Xilinx FPGA.
- Provide specifics on how to compile and optimize an algorithm on a Xilinx FPGA.

After completing this tutorial, you will be able to:

- Compile and optimize a Smith-Waterman Sequence Alignment Algorithm.

Getting Started

Setup Requirements

Before you start this tutorial, make sure you have and understand the hardware and software components needed to perform the labs included in this tutorial as listed below.

Software Requirements

To compile OpenCL programs into Xilinx FPGAs, you must have the following tools installed:

- Xilinx SDAccel 2015.1

Contact your Xilinx representative for instructions on how to download the tools, and how to obtain licensing.

Hardware Requirements

- Memory: 16GB
- Hard Drive: 500GB
- Chasis: 1 PCIe Gen3 x8 or x16 slot
- Operating System RedHat Enterprise Linux or CentOS 6.4-6.6 64-bit
- Acceleration Card: Alpha Data ADM-PCIE-7v3 or Pico MX505-EX400 card

Design Files

You can find all of the example files for SDAccel located at:

```
<sdaccel installation directory>/SDAccel/2015.1/examples
```

The example for this tutorial is located at:

```
<sdaccel installation directory>/SDAccel/2015.1/examples/getting_started
```

Smith-Waterman Sequence Alignment Algorithm

In this tutorial, you will use SDAccel to compile the Smith-Waterman algorithm onto a Xilinx FPGA. The Smith-Waterman algorithm is a database search algorithm developed by T.F. Smith and M.S. Waterman, and is based on an earlier algorithm named Needleman and Wunsch after its original creators. The key 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 of two sequences 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 Figure 1.

- Sequence 1 = ACACACTA
- Sequence 2 = AGCACACA
- $s(a, b) = +2 \text{ if } a = b (\text{match}), \; -1 \text{ if } a \neq b \text{ (mismatch)}$
- $W_i = -i$

$$H = \begin{pmatrix} & - & A & C & A & C & A & C & T & A \\ - & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ A & 0 & 2 & 1 & 2 & 1 & 2 & 1 & 0 & 2 \\ G & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 \\ C & 0 & 0 & 3 & 2 & 3 & 2 & 3 & 2 & 1 \\ A & 0 & 2 & 2 & 5 & 4 & 5 & 4 & 3 & 4 \\ C & 0 & 1 & 4 & 4 & 7 & 6 & 7 & 6 & 5 \\ A & 0 & 2 & 3 & 6 & 6 & 9 & 8 & 7 & 8 \\ C & 0 & 1 & 4 & 5 & 8 & 8 & 11 & 10 & 9 \\ A & 0 & 2 & 3 & 6 & 7 & 10 & 10 & 10 & 12 \end{pmatrix} \quad T = \begin{pmatrix} & - & A & C & A & C & A & C & T & A \\ - & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ A & 0 & \nwarrow & \leftarrow & \nwarrow & \leftarrow & \nwarrow & \leftarrow & \leftarrow & \nwarrow \\ G & 0 & \uparrow & \nwarrow & \uparrow & \nwarrow & \uparrow & \nwarrow & \nwarrow & \uparrow \\ C & 0 & \uparrow & \nwarrow & \leftarrow & \nwarrow & \leftarrow & \nwarrow & \leftarrow & \leftarrow \\ A & 0 & \nwarrow & \uparrow & \nwarrow & \leftarrow & \nwarrow & \leftarrow & \leftarrow & \nwarrow \\ C & 0 & \uparrow & \nwarrow & \uparrow & \nwarrow & \leftarrow & \nwarrow & \leftarrow & \leftarrow \\ A & 0 & \nwarrow & \uparrow & \nwarrow & \uparrow & \nwarrow & \leftarrow & \leftarrow & \nwarrow \\ C & 0 & \uparrow & \nwarrow & \uparrow & \nwarrow & \uparrow & \nwarrow & \leftarrow & \leftarrow \\ A & 0 & \nwarrow & \uparrow & \nwarrow & \uparrow & \nwarrow & \uparrow & \nwarrow & \nwarrow \end{pmatrix}$$

Figure 1: **Smith-Waterman Example**

For the pair of sequences shown in Figure 1, the sequence alignment is determined by the two resulting matrices from the Smith-Waterman algorithm. The cost matrix, denoted as matrix H in Figure 1, 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 in Figure 1 are:

Sequence 1 = `A-CACACTA`

Sequence 2 = `AGCACAC-A`

Lab 1: Compile and Optimize an Algorithm

SDAccel enables a programmer to quickly iterate through changes in an OpenCL application to arrive at an optimized version targeted at a specific board. The code and optimize iteration loop shown in Figure 1 captures the design methodology behind OpenCL once the programmer has completed the functionality of the application.

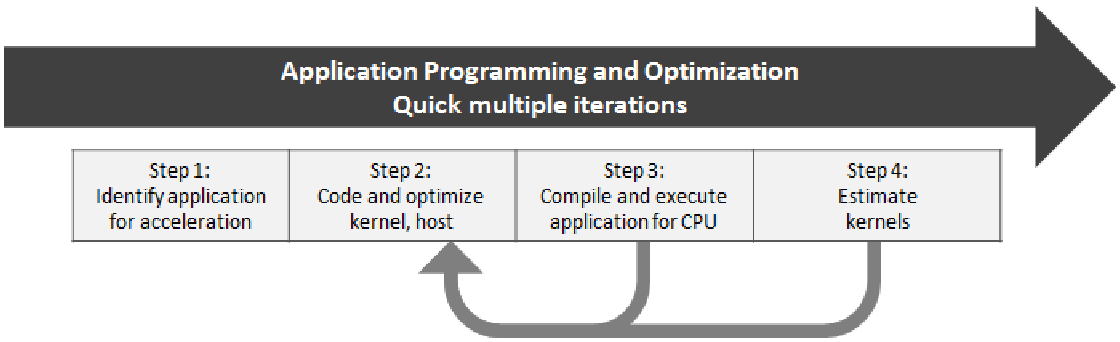

Figure 1: **SDAccel Application Programming and Optimization Design Flow**

The main steps in application compilation using SDAccel are:

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* (UG1023).
3. Instantiate multiple copies of a kernel in the FPGA. SDAccel 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:

- For Alpha Data: `<user directory>/getting_started/alpha_data`
- For Pico: `<user_directory>/getting_started/pico`

Step 1: Compiling the Baseline Version of an Application

SDAccel is a command line tool that is driven by a command script. The command script for basic compilation of the getting started example is `baseline.tcl`. The commands in `baseline.tcl` carry out the following tasks:

1. Define the directory where all tool output will be stored.
2. Select the device where the application will 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.

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

```
sdaccel baseline.tcl
```

Upon successful execution of the application, the following output will be as shown in Figure 2.

```
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:
INFO: [SDAccel 60-174] /proj/baseline_solution/impl/sim/cpu_em/baseline_solution.exe -d acc -k test.xclbin


Input sequence1: TAGGCAAGACCACTTTAGCATGGTCTACAACGCCTAGACCTTTGGCAAAGCAGATCGGCCCGCCCATCACTAGTGGGACTATCC
Input sequence2: TAATGGGAACACCTGCTGCAATCGGATCGTTGCAGCGGTAATGTGTCGGTATATGCGAGTAGGGTAATCCAAACGTCCCATTGC

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-CGG----CC---CG-CCCAT
Align sequence2: TAATGGGAACA-C-CTGCT-GCAATCGGATCGTTGCAGCG-GTA-A--TGTGTCGGTATATGC-GAGTAGGGTAATCCAAACGTCCCAT

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

Figure 2: **Getting Started Application Output**

Before determining which attribute to apply to improve performance, you must review the system estimate report. The system estimate report which is located at:

```
baseline_project/rpt
```

For the baseline version of the this application, the system estimate report is as shown in Figure 3.

```
-------------------------------------------------------------------------------
Design Name:      baseline_solution
Target Platform:  1ddr
Target Board:
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         | 239.808               | 273.973               |
+---------------------+-----------------------+-----------------------+-----------------------+

Latency Information (clock cycles)
+-------------------+------------------+------------------+-------------+-------------+-------------+
| Compute Unit      | Kernel Name      | Start Interval   | Best Case   | Avg Case    | Worst Case  |
+-------------------+------------------+------------------+-------------+-------------+-------------+
| K1                | smithwaterman    | 166254 ~ 209094  | 166253      | 187673      | 209093      |
+-------------------+------------------+------------------+-------------+-------------+-------------+

Area Information
+---------------------+---------------------+------------+------------+------------+------------+
| Compute Unit        | Kernel Name         | FF         | LUT        | DSP        | BRAM       |
+---------------------+---------------------+------------+------------+------------+------------+
| K1                  | smithwaterman       | 1771       | 2911       | 1          | 9          |
+---------------------+---------------------+------------+------------+------------+------------+
```

Figure 3: **System Estimate Report**

The system estimate report of Figure 4 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.

Step 2: 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 SDAccel are described in detail in the *SDAccel Development Environment User Guide* (UG1023). For the purposes of this guide, only the pipeline attribute is introduced.

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. The modified code showing the loop pipeline attribute is as shown in Figure 4.

```
  // Loops transferring data from DDR to BRAM memories are transformed into
  // burst memory transfers over an AXI master interface once the loop is pipelined
// Get the first sequence from DDR
  __attribute__((xcl_pipeline_loop))
  for (int i = 1; i < N; i++) {
      localS1[i] = s1[i + seq_id*N];
  }

  // Get the second sequence from DDR
  __attribute__((xcl_pipeline_loop))
  for (int i = 1; i < N; i++) {
      localS2[i] = s2[i + seq_id*N];
  }

  // Get the search matrix from DDR
  __attribute__((xcl_pipeline_loop))
  for (int i = 0; i < N * N; i++) {
      localMatrix[i] = matrix[i + seq_id*N*N];
  }


  // Compute the similarity matrix for the sequences
  __attribute__((xcl_pipeline_loop))
  for (int index = N; index < N * N; index++)
  {
          int dir = CENTER;
          int val = 0;
          int j = index % N;

          // Skip the first column, which is not part of the similarity
          // matrix cost function
          if (j == 0) {
              west = 0;
              northwest = 0;
              continue;
          }

          // Determine if element j of sequence 1 matches element i of sequence 2
          int i = index / N;
          north = localMatrix[index - N];
          const int match = (localS1[j] == localS2[i]) ? MATCH : MISS_MATCH;

          // Determine the new cost value at the current point in the similarity matrix
          // Update the direction of motion to trace the sequence alignment
          int val1 = northwest + match;
          if (val1 > val) {
              val = val1;
              dir = NORTH_WEST;
          }
```

Figure 4: **Application Code with Loop Pipeline Attribute**

The command script for compiling the application after the addition of the code attribute is identical to the baseline script. Since this optimization is defined in the application source code, there is no need for additional commands in SDAccel. To compile this version of the application, execute:

```
sdaccel pipelined.tcl
```

The system estimate report for this compilation run, shown in Figure 4, demonstrates the results of the pipeline attribute. After this optimization is applied, the resulting compute unit is seven times faster than the baseline run.

```
-------------------------------------------------------------------------------
Design Name:      pipelined_solution
Target Platform:  1ddr
Target Board:
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         | 239.808                   | 233.645                  |
+----------------------+-----------------------+---------------------------+--------------------------+

Latency Information (clock cycles)
+---------------------+--------------------+--------------------+---------------+-------------+---------------+
| Compute Unit        | Kernel Name        | Start Interval     | Best Case     | Avg Case    | Worst Case    |
+---------------------+--------------------+--------------------+---------------+-------------+---------------+
| K1                  | smithwaterman      | 21815              | 21814         | 21814       | 21814         |
+---------------------+--------------------+--------------------+---------------+-------------+---------------+

Area Information
+-----------------------+-----------------------+------------+------------+------------+------------+
| Compute Unit          | Kernel Name           | FF         | LUT        | DSP        | BRAM       |
+-----------------------+-----------------------+------------+------------+------------+------------+
| K1                    | smithwaterman         | 2072       | 3157       | 1          | 9          |
+-----------------------+-----------------------+------------+------------+------------+------------+
```

Figure 5: **System Estimate Report After Optimization**

Step 3: Compiling to Run on the Board

Once 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.

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

```
sdaccel board_compile.tcl
```

The compile time to generate binaries for the FPGA board will take at least 30 minutes. Once SDAccel has finished the compilation process, the output binaries are placed in the following directory:

```
board_compilation/pkg/pcie
```

If the target board is available on the system where SDAccel is installed,

1. Copy the `setupenv.sh` script to `board_compilation/pkg/pcie`.

   ```
   cp setupenv.sh board_compilation/pkg/pcie
   ```

2. Go to the `board_compilation/pkg/pcie` directory.

   ```
   cd board_compilation/pkg/pcie
   ```

3. Open `setupenv.sh` in a text editor and edit it to match the local SDAccel installation.
4. Source the `setupenv.sh` script file to get the system ready to run an application on the card.

   ```
   source setupenv.sh
   ```

5. Run the application:

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

Appendix A

Installing the Alpha Data ADM-PCIE-7V3 Card

The ADM-PCIE-7V3 board from Alpha Data is an FPGA co-processing card that can be used for OpenCL applications compiled by SDAccel. The main characteristics of this card for a co-processing accelerator solution are:

- One Xilinx Virtex®-7 690T FPGA device
- 16 GB of DDR3

The card is shown in Figure A-1.

Figure A-1: **Alpha Data ADM-PCIE-7V3**

Server and Workstation Requirements

The server/workstation requirements to run the Alpha Data ADM-PCIE-7V3 card are:

- RedHat Enterprise Linux or CentOS 6.4-6.6 64-bit.
- Available PCIe gen3 x16 slot at a minimum power rating of 75W.
- Driver located at:

  `<Solution directory>/pkg/pcie/runtime/platforms/<target DSA>/driver`

Card Settings

The Alpha Data card has a DIP switch in the back. For proper operation, the switch, which is shown in Table A-1, must be set in the following positions.

Table A-1: Alpha Data DIP Switch Positions

Position 1	Position 2	Position 3	Position 4
ON	ON	ON	OFF

Base Platform Programming

All applications compiled by SDAccel for the Alpha Data card are compiled against a specific device. A device is a combination of interfaces and infrastructure components on the card, which are required for proper execution of the user program. The base device program or firmware is different for all devices. This program must be loaded onto the FPGA card before the user application is loaded. The device base program file, `Xilinx_adm-pcie-7v3_1ddr-xdma_1_0.mcs`, is located at:

```
pkg/pcie/runtime/platforms/xilinx_adm-pcie-7v3_1ddr_1_0/driver/
```

The steps to program the firmware program are:

1. Connect a JTAG cable to the Alpha Data card and the other end of the cable to a computer with Vivado Design Suite 2015.1
2. Start Vivado Design Suite 2015.1
3. Select Open Hardware Manager (see Figure A-2).

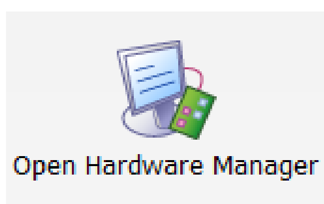

Figure A-2: **Open Hardware Manager**

4. Select **Open a New Hardware Target** (see Figure A-3).

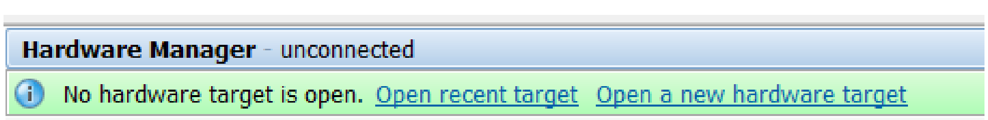

Figure A-3: **Hardware Manager**

5. Click **Next** on the Open Hardware Target Panel (see Figure A-4).

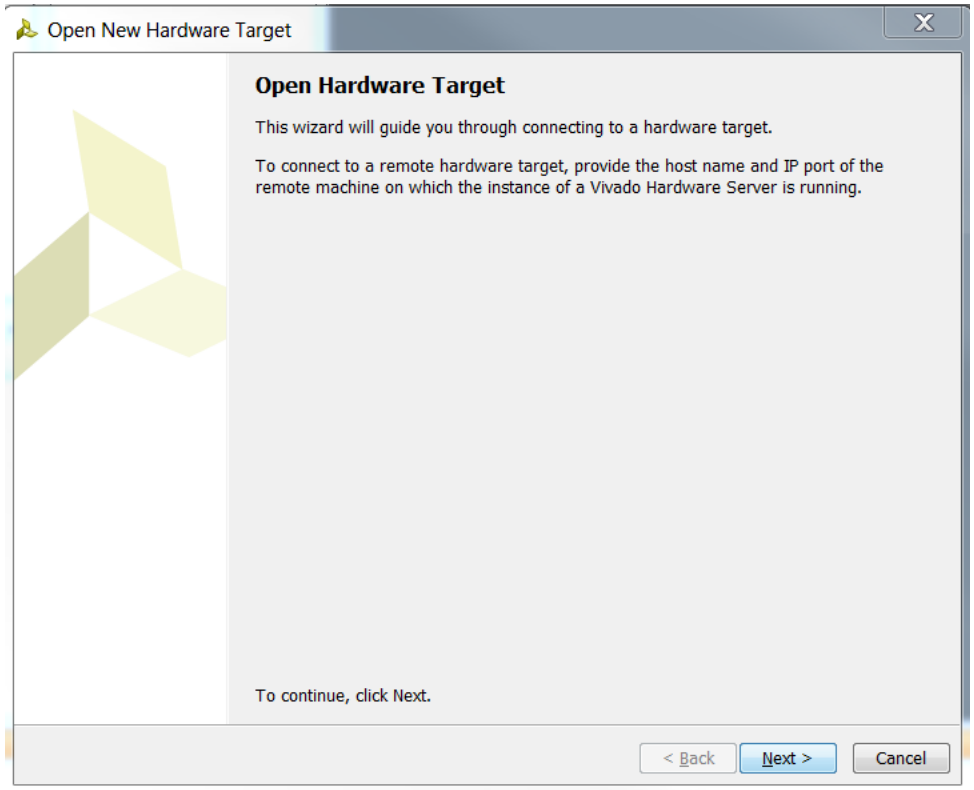

Figure A-4: **Open Hardware Target**

6. On the **Connect to** line, select **Local server** in the drop-down menu and click **Next** (see Figure A-5).

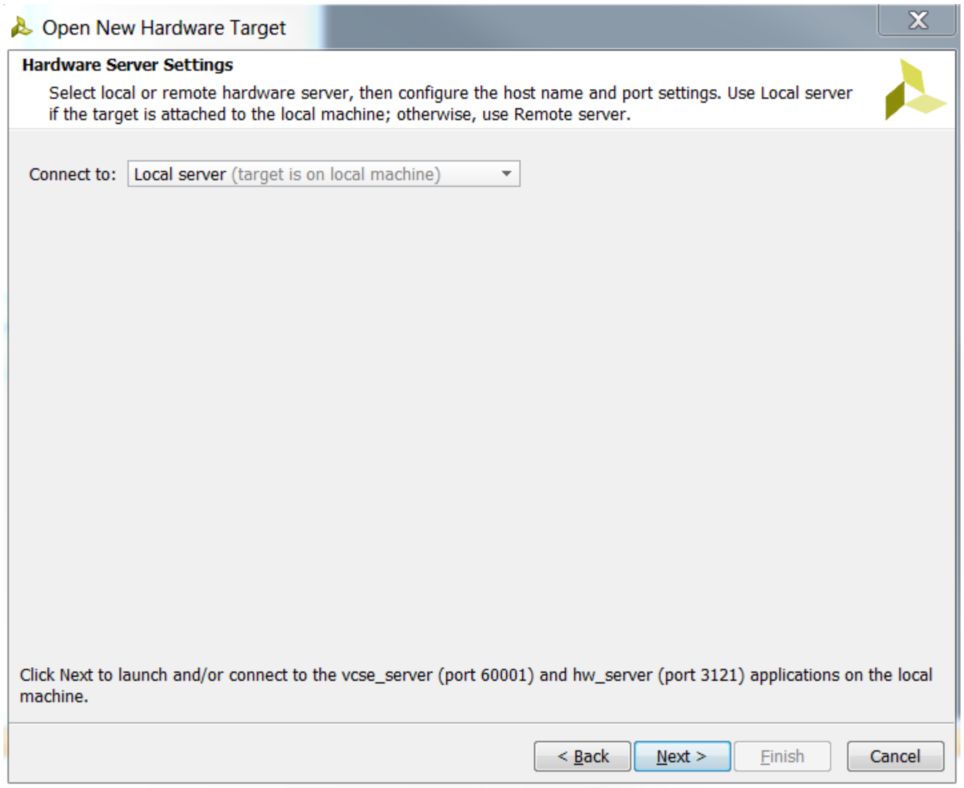

Figure A-5: **Open New Hardware Target**

7. Select **xilinx_tcf** for the hardware target and click **Next** (see Figure A-6).

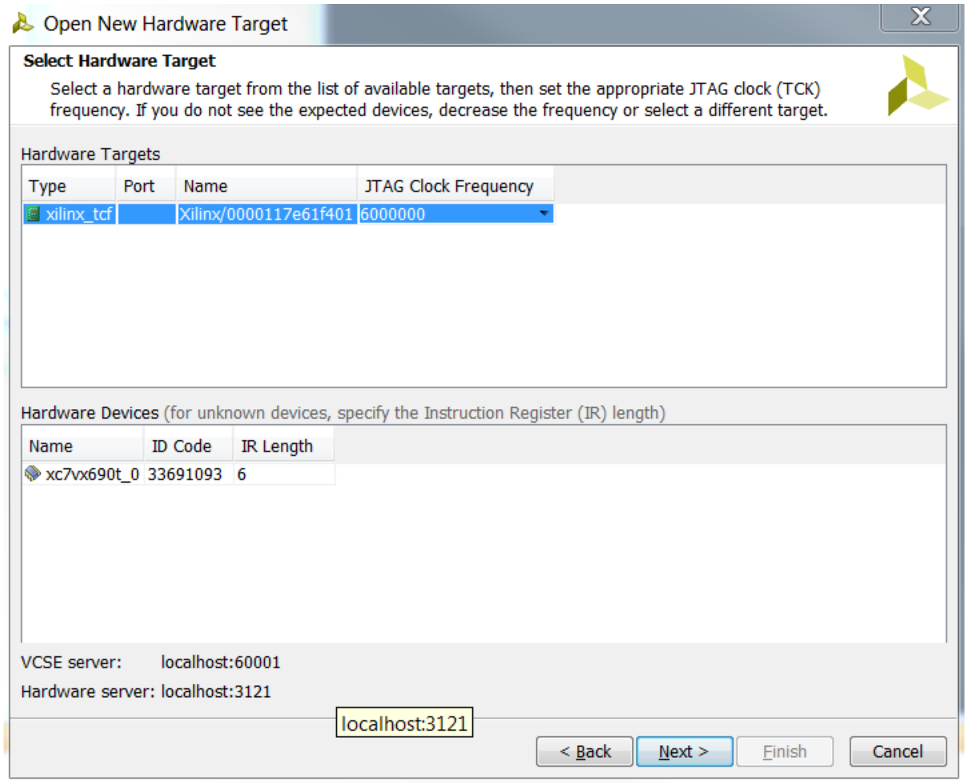

Figure A-6: **Select Hardware Target**

8. Click **Finish** on the Open Hardware Target Summary panel (see Figure A-7).

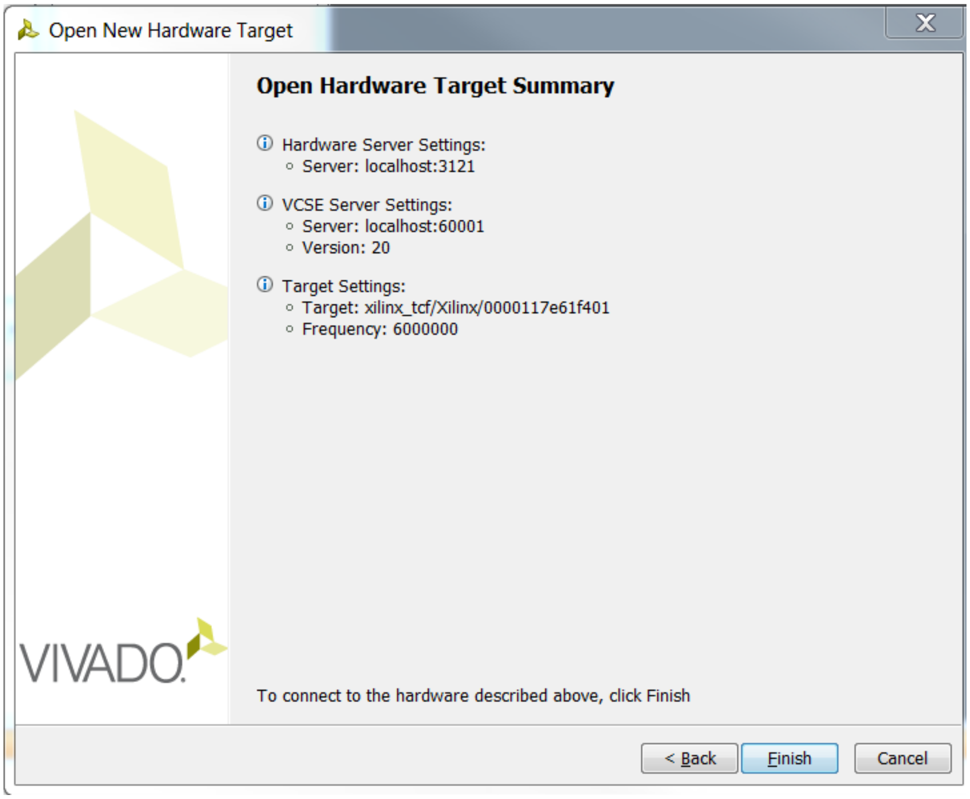

Figure A-7: **Open New Hardware Target**

9. Right-click the FPGA device and select **Add Configuration Memory Device** (see Figure A-8).

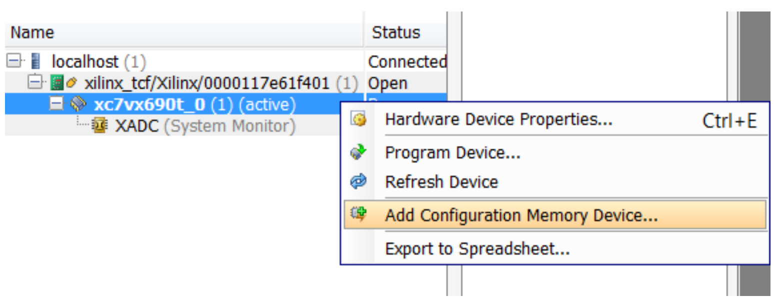

Figure A-8: **Add Configuration Memory Device**

10. Select **mt28gu01gaax1e-bpi-x16** as the configuration memory (see Figure A-9).

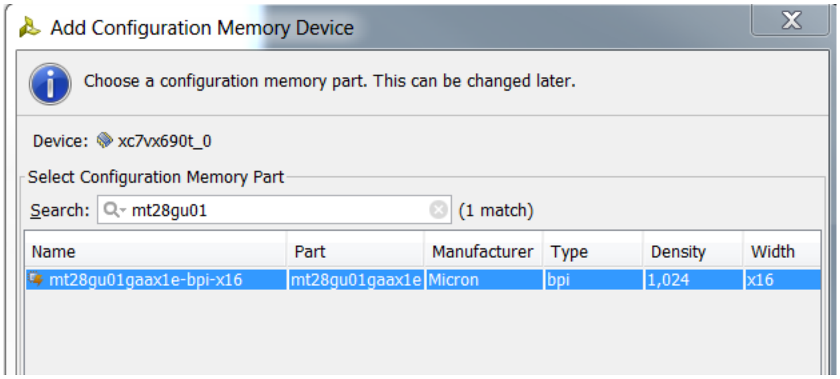

Figure A-9: **Selecting the Device**

11. Click **OK** to add the firmware file for the memory (see Figure A-10).

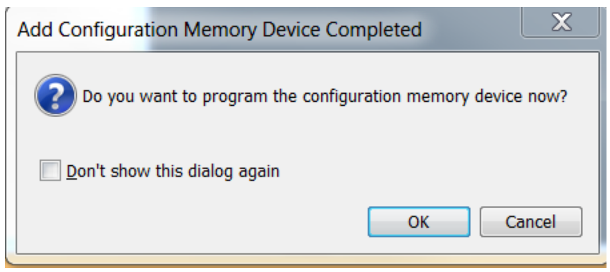

Figure A-10: **Add Configuration Memory Device Completed**

12. Select the configuration file for the platform. This is the MCS file in the firmware directory. In addition, set the following settings, as shown in Figure A-11:

- RS Pins: **None**
- Select **Erase**
- Select **Program**
- Un-select **Verify**

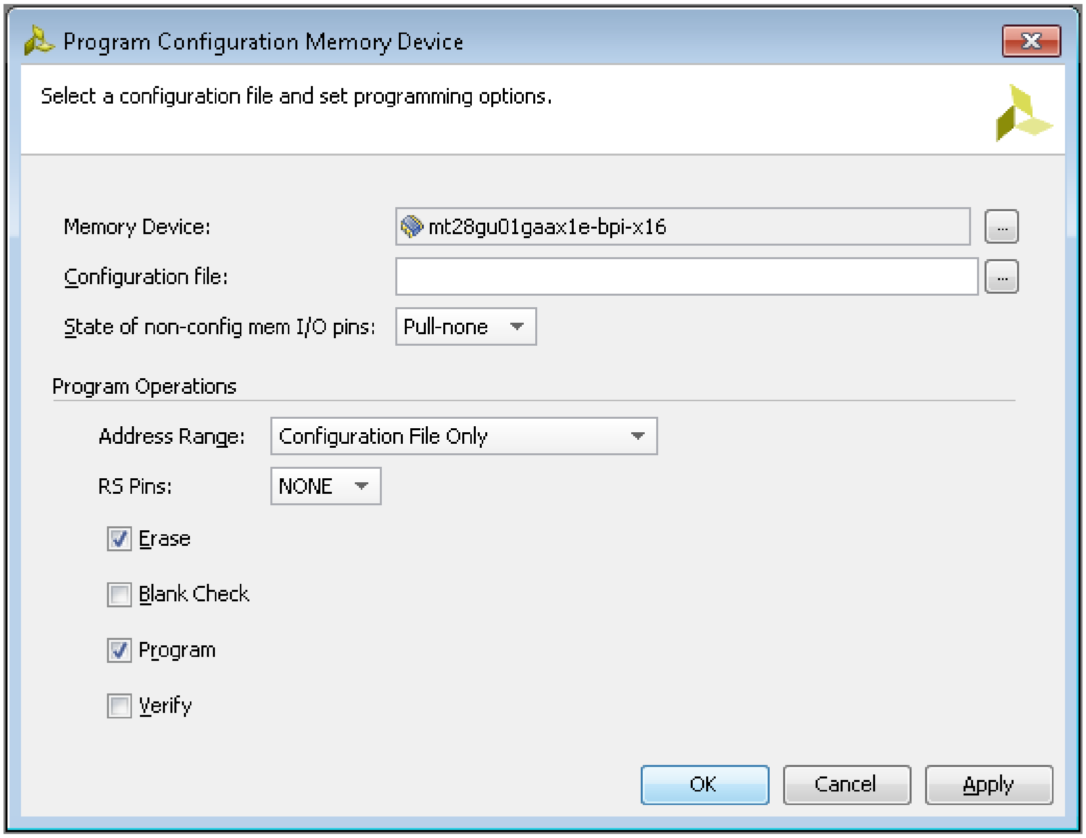

Figure A-11: **Configuration Settings**

13. Click **OK** to start programming the memory.

14. After the memory has been configured, right-click the FPGA device and select **Boot Device** (see Figure A-12).

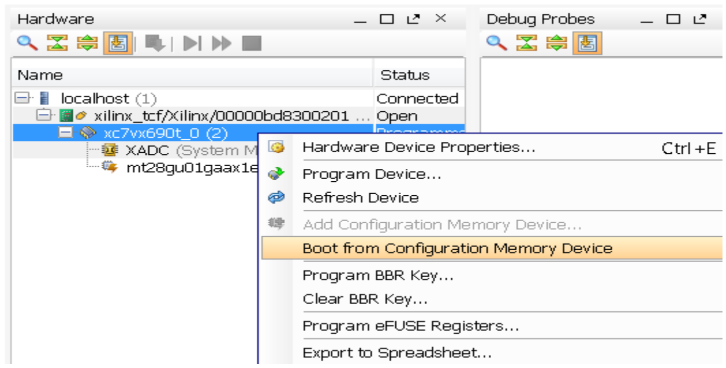

Figure A-12: **Select Boot Device**

15. Click **OK** to confirm booting the device (see Figure A-13).

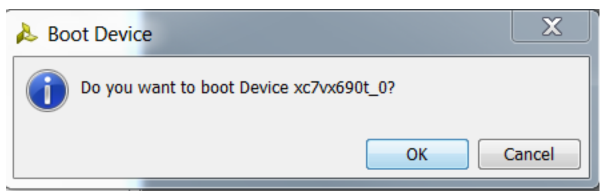

Figure A-13: **Boot Device**

16. Reboot the machine where the Alpha Data card is installed.

IMPORTANT: *Programming of the device firmware is required only once per device. All applications targeting the same device can share a single programming instance of the card firmware.*

Linux Driver Installation

The driver for the Alpha Data card can be found in the `pkg` directory of applications compiled for this card. After application compilation, the driver will be located at:

```
<Solution directory>/pkg/pcie/runtime/platforms/<target DSA>/driver
```

The steps to install the driver are:

1. Unzip `driver.zip`

```
unzip driver.zip
```

2. Execute the make command:

```
make
```

3. As a user with root or sudo level access:

```
cp 10-xdma.rules /etc/udev/rules.d
insmod  xdma.ko
```

The command sequence above installs and loads the driver for the Alpha Data card for the current Linux session. The driver will not be automatically loaded the next time the machine is rebooted unless changes are made to the Linux boot configuration. Refer to the Linux manuals for the Linux version on your machine for the proper way to modify Linux boot configuration files.

IMPORTANT: *The driver for the Alpha Data card is common across all applications and platforms targeting this card. It only needs to be installed once on the target system.*

Installing the Pico MX505-EX400 Card

The Pico Computing platform for OpenCL applications supported by SDAccel consists of two components. The first component is a PCIe backplane shown in Figure B-1. This backplane provides the PCIe connectivity and power infrastructure to run the FPGA based acceleration modules. The backplane can support a maximum of four FPGA acceleration modules. For proper operation, this backplane cards requires an additional PCIe power cable directly connected to the system power supply.

Figure B-1 shows the Pico Computing EX400 PCIe card.

Figure B-1 **Pico Computing EX400 PCIe Backplane**

The second component of the solution is the MX505 FPGA acceleration module. The module, shown in Figure B-2, uses one Xilinx Kintex®-7 325 FPGA. OpenCL kernels are mapped to the FPGA logic of the module. The current release of SDAccel supports only one MX505 module connected to the EX400 backplane.

Figure B-2 **Pico Computing MX505 FPGA Acceleration Module**

Server and Workstation Requirements

The server/workstation requirements to run the Pico Computing MX505-EX400 card are as follows:

- RedHat Enterprise Linux or CentOS 6.4-6.6 64-bit
- Available PCIe gen2 slot

Linux Driver Installation

The driver for the Pico Computing card can be found in the `pkg` directory of applications compiled for this card. After application compilation, the driver will be located at:

```
pkg/pcie/runtime/platforms/xilinx_pico_m505_pico_1_0/driver/
```

The steps to install the driver area:

1. Unzip `driver.zip`:

   ```
   unzip driver.zip
   ```

2. Change directory to `pico/kernel`:

   ```
   cd pico/kernel
   ```

3. Execute the make command:

   ```
   make
   ```

4. As a user with root or sudo level access:

   ```
   cp 10-pico.rules /etc/udev/rules.d
   insmod pico.ko
   ```

The command sequence above installs and loads the driver for the Pico card for the current Linux session. The driver will be not be automatically loaded the next time the machine is rebooted unless changes are made to the Linux boot configuration. Refer to the Linux manuals for the Linux version on the machine for the proper way to modify Linux boot configuration files.

IMPORTANT: *The driver for the Pico card is common across all applications and platforms targeting this card. It only needs to be installed once on the target system.*

Legal Notices

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