SDAccel Development Environment

Tutorial

UG1021 (2015.3) October 22, 2015
# Revision History

The following table shows the revision history for this document.

<table>
<thead>
<tr>
<th>Date</th>
<th>Version</th>
<th>Revision</th>
</tr>
</thead>
<tbody>
<tr>
<td>10/22/2015</td>
<td>2015.3</td>
<td>Restructured tutorial. Added information for the Alpha Data ADM-PCIE-KU3 card.</td>
</tr>
<tr>
<td>06/29/2015</td>
<td>2015.1.2</td>
<td>Updates to firmware location and settings</td>
</tr>
<tr>
<td>05/26/2015</td>
<td>2015.1</td>
<td>DRC Check updates + more</td>
</tr>
</tbody>
</table>
Table of Contents

Revision History .............................................................................................................. 2
Table of Contents ............................................................................................................. 3
Chapter 1: Introduction ................................................................................................. 4
  Objectives ....................................................................................................................... 4
  Design Overview ............................................................................................................. 4
  Hardware Requirements ............................................................................................... 6
  Software Requirements ................................................................................................. 6
  Design Files .................................................................................................................... 6
Chapter 2: Lab 1: Setting Up the System ................................................................... 7
  Lab 1a: Installing the Alpha Data ADM-PCIE-KU3 Card ........................................ 7
  Lab 1b: Installing the Alpha Data ADM-PCIE-7V3 Card ........................................... 17
  Lab 1c: Installing the Pico Computing M-505-K325T/EX400 Card ......................... 26
Chapter 3: Lab 2: Compiling and Optimizing the Application ............................... 29
  Step 1: Compiling and Running the Baseline Application .................................... 30
  Step 2: Optimizing the Kernel Using Attributes ..................................................... 30
  Step 3: Compiling and Running the Optimized Application ................................. 33
Appendix A: Additional Resources and Legal Notices ........................................... 35
  Xilinx Resources .......................................................................................................... 35
  Solution Centers ............................................................................................................ 35
  Please Read: Important Legal Notices ....................................................................... 35
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 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

\[ H_{ij} = \max(H_{i-1, j-1} + s(ai, bj); H_{i-1, k} - W_k; H_{i, j-1} - W_1; 0) \]

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 = b (mismatch)
- W_i = -i

\[
\begin{bmatrix}
  \_ & A & C & A & C & T & A \\
  0 & 0 & 0 & 0 & 0 & 0 & 0 \\
  A & 2 & 1 & 2 & 1 & 2 & 1 & 0 & 2 \\
  G & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 \\
  C & 0 & 3 & 2 & 3 & 2 & 3 & 2 & 1 \\
  A & 0 & 2 & 2 & 5 & 4 & 5 & 4 & 3 & 4 \\
  C & 0 & 1 & 4 & 4 & 7 & 6 & 7 & 6 & 5 \\
  A & 0 & 2 & 3 & 6 & 6 & 9 & 8 & 7 & 8 \\
  C & 0 & 1 & 4 & 5 & 8 & 8 & 11 & 10 & 9 \\
  A & 0 & 2 & 3 & 6 & 7 & 10 & 10 & 10 & 12 \\
\end{bmatrix}
\]

\[
\begin{bmatrix}
  \_ & A & C & A & C & T & A \\
  0 & 0 & 0 & 0 & 0 & 0 & 0 \\
  A & 0 & \_ & \_ & \_ & \_ & \_ & \_ \\
  G & 0 & \_ & \_ & \_ & \_ & \_ & \_ & \_ & \_ & \_ & \_ & \_ & \_ & \_ & \_ \\
Chapter 1: Introduction

Hardware Requirements

- Acceleration Card. Use one of the following:
  - Alpha Data ADM-PCIE-KU3/KU060E card. Card is based on the Kintex® UltraScale XCKU060T-2FFVA1156E FPGA.
  - Alpha Data ADM-PCIE-7V3/VX690T-2 card. Card is based on the Virtex®-7 XC7VX690T-2FFG1157C FPGA.
  - Pico Computing card consisting of an EX-400 backplane card with an M-505-K325T module. The module is based on the Kintex®-7 XC7K325T-2FFG1157C FPGA.
- Host computer: Desktop computer for hosting the acceleration card. The host computer must provide:
  - Motherboard with a PCIe Gen3 X16 slot
  - 16 GB DDR3 Memory
  - 500 GB Hard drive
  - DVD drive
  - Power supply capable of supplying 75W to the Pico Computing M-505-K325T/EX400 card. Power is supplied by a cable connection between the host computer power supply and the card.
  - A monitor, keyboard, and mouse
- Programming computer: Laptop or desktop computer running Vivado Design Suite 2015.3 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 Platform Cable USB II [DS593])

Software Requirements

- Vivado Design Suite v2015.3
- SDAccel Development Environment v2015.3
- RedHat Enterprise Linux or CentOS 6.4-6.7 64-bit
- Linux Driver (available in the example design files included with the SDAccel Development Environment here: <Solution directory>/pkg/pcie/runtime/platforms/<target DSA>/driver)

Design Files

Example design files are included with the SDAccel Development Environment v2015.3
Chapter 2

Lab 1: Setting Up the System

Follow the installation instructions appropriate for your acceleration card:

- Lab 1a: Installing the Alpha Data ADM-PCIE-KU3 Card
- Lab 1b: Installing the Alpha Data ADM-PCIE-7V3 Card
- Lab 1c: Installing the Pico Computing M-505-K325T/EX400 Card

Lab 1a: Installing the Alpha Data ADM-PCIE-KU3 Card

The ADM-PCIE-KU3/KU060E card is a high-performance reconfigurable computing card for data center applications. It features:

- Kintex® UltraScale™ XCKU060T-2FFVA1156E FPGA
- 16 GB of DDR3 memory

*Figure 2–1: ADM-PCIE-KU3/KU060E card*
Step 1: Setting up the Card and Computer

1. Locate DIP switch SW1 on the back side of the board near the center of top edge and set it as shown here:

![Figure 2–2: DIP Switch SW1](image)

2. Install the ADM-PCIE-KU3 card into an open PCIe slot in the host computer.

**IMPORTANT:** Follow manufacturer recommendations to ensure proper mounting and adequate cooling.

Step 2: Programming the Base Platform

All applications compiled by the SDAccel™ compiler 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 before the user application is loaded. To program the firmware program:
Chapter 2: Lab 1: Setting Up the System

1. Connect the Alpha Data ADM-PCIE-KU3 card to the programming computer with an installation of Vivado® Design Suite as shown here:

   **Figure 2–3: Programming Connections**

2. On the control computer, in the Vivado Design Suite Welcome page, select **Open Hardware Manager**.

   **Figure 2–4: Open Hardware Manager Icon**

3. Select **Open a New Hardware Target**.

   **Figure 2–5: Open a New Hardware Target Link**
4. Click **Next** in the Open Hardware Target window.

**Figure 2–6: Open New Hardware Target Window**
5. Select **Local server** in the **Connect to** field and click **Next**.

*Figure 2–7: Hardware Server Settings*
6. In the Open New Hardware Target window, select `xilinx_tcf` and click **Next**.
7. In the Open Hardware Target Summary window, click **Finish**.

*Figure 2–9: Open Hardware Target Summary*
8. Right-click the FPGA (\texttt{xku060.0}) and select \textbf{Add Configuration Memory Device}.

\textit{Figure 2–10: Add Configuration Memory Device}

![Figure 2–10: Add Configuration Memory Device](image)

9. Select \texttt{mt28gu01gaax1e-bpi-x16} as the configuration memory.

\textit{Figure 2–11: Select Configuration Memory}

![Figure 2–11: Select Configuration Memory](image)
10. Click **OK** to program the configuration memory.

*Figure 2–12: Program Configuration Memory*

![Program Configuration Memory Completed](image)

11. Select, enter, or verify the values in the fields shown in the Program Configuration Memory Device window, and click **OK** to start programming the memory.

*Figure 2–13: Program Configuration Memory Device Window*

![Program Configuration Memory Device](image)
12. After the memory has been configured, right-click the FPGA (xku060_0) and select **Boot From Configuration Device**.

![Figure 2–14: Boot from Configuration Memory Device](image)

13. Reboot the host computer.

**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.

**Step 3: Deploying SDAccel Applications**

Each SDAccel™ application is compiled targeting a specific acceleration card. A deployment script is generated for each application that will:

1. Copy the necessary runtime files to the target directory specified with the `-d` switch.
2. Compile and install Linux kernel device drivers unless the `-k no` switch is specified.
3. Install the firmware files to the Linux firmware area.
4. Generate a `setup.sh` or `setup.csh` environment script which can be used to set up the correct environment.

Follow the instructions below to run the deployment script:

1. Navigate to: `<Solution directory>/pkg/pcie/`
2. As a user with root or sudo level access, run `./install.sh -d <install_dir>`
Lab 1b: Installing the Alpha Data ADM-PCIE-7V3 Card

The ADM-PCIE-7V3/VX690T-2 card is a high-performance reconfigurable computing card for data center applications. It features:

- Virtex®-7 XC7VX690T-2FFG1157C FPGA
- 16 GB of DDR3 memory

*Figure 2–15: ADM-PCIE-7V3/VX690T-2 card*
Step 1: Setting up the Card and Computer

1. Locate DIP switch SW1 on the back side of the board near the card bracket and set it as shown here:

![Figure 2–16: DIP Switch SW1](image)

2. Install the ADM-PCIE-7V3 card into an open PCIe slot in the host computer.

**IMPORTANT:** Follow manufacturer recommendations to ensure proper mounting and adequate cooling.

Step 2: Programming the Base Platform

All applications compiled by the SDAccel™ compiler 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 before the user application is loaded. To program the firmware program:
Chapter 2: Lab 1: Setting Up the System

1. Connect the Alpha Data ADM-PCIE-7V3 card to the programming computer with an installation of Vivado® Design Suite as shown here:

   **Figure 2–17: Programming Connections**

   ![Programming Connections Diagram]

   USB cable standard-A plug to standard B plug
   JTAG Connector
   Card Installed in Host Computer
   Programming Computer

2. On the control computer, in the Vivado Design Suite Welcome page, select **Open Hardware Manager**.

   **Figure 2–18: Open Hardware Manager Icon**

   ![Open Hardware Manager Icon]

3. Select **Open a New Hardware Target**.

   **Figure 2–19: Open a New Hardware Target Link**

   ![Hardware Manager - unconnected]

   No hardware target is open. Open recent target Open a new hardware target
4. Click **Next** in the Open Hardware Target window.

---

**Figure 2–20: Open New Hardware Target Window**
5. Select **Local server** in the **Connect to** field and click **Next**.

*Figure 2–21: Hardware Server Settings*
6. In the Open New Hardware Target window, select `xilinx_tcf` and click **Next**.

*Figure 2–22: Select Hardware Target*
7. In the Open Hardware Target Summary window, click **Finish**.

*Figure 2–23: Open Hardware Target Summary*

8. Right-click the FPGA (**xc7vx690T_0**) and select **Add Configuration Memory Device**.

*Figure 2–24: Add Configuration Memory Device*
9. Select **mt28gu01gaax1e-bpi-x16** as the configuration memory.

*Figure 2–25: Select Configuration Memory*

10. Click **OK** to program the configuration memory.

*Figure 2–26: Program Configuration Memory*
11. Select, enter, or verify the values in the fields shown in the Program Configuration Memory Device window, and click OK to start programming the memory.

*Figure 2–27: Program Configuration Memory Device Window*

12. After the memory has been configured, right-click the FPGA (xc7vx690T_0) and select **Boot From Configuration Device**.

*Figure 2–28: Boot from Configuration Memory Device*
13. Reboot the host computer.

**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.

### Step 3: Deploying SDAccel Applications

Each SDAccel™ application is compiled targeting a specific acceleration card. A deployment script is generated for each application that will:

1. Copy the necessary runtime files to the target directory specified with the `-d` switch.
2. Compile and install Linux kernel device drivers unless the `-k no` switch is specified.
3. Install the firmware files to the Linux firmware area.
4. Generate a `setup.sh` or `setup.csh` environment script which can be used to set up the correct environment.

Follow the instructions below to run the deployment script:

1. **Navigate to:** `<Solution directory>/pkg/pcie/`
2. **As a user with root or sudo level access,** run `. /install.sh -d <install_dir>`

### Lab 1c: Installing the Pico Computing M-505-K325T/EX400 Card

The Pico Computing platform for OpenCL™ applications supported by the SDAccel™ development environment consists of the EX400 PCIe™ backplane card and an MX505 FPGA acceleration module mounted onto the backplane card. The backplane card provides PCIe connectivity and power infrastructure to run up to four FPGA-based acceleration modules. For proper operation, the backplane card requires an additional PCIe power cable directly connected to the host computer power supply.

The EX400 PCIe backplane card is shown in the following figure:
The MX505 FPGA acceleration module uses one Kintex®-7 XC7K325T-2FFG1157C FPGA. OpenCL kernels are mapped to the FPGA logic of the module. The current release of the SDAccel development environment supports only one MX505 module connected to the EX400 backplane.

**Step 1: Setting up the Card and Computer**

1. Mount the MX505 FPGA acceleration module onto the EX400 PCIe™ backplane card. Follow manufacturer instructions.

2. Install the Pico Computing card into an open PCIe slot in the host computer. Follow manufacturer instructions.

   **IMPORTANT:** Follow manufacturer recommendations to ensure proper mounting and adequate cooling.

3. Install the power cable from the host computer power supply (the ATX 6-Pin auxiliary power connector) to the 6-pin PCIe connector on the EX400 backplane card.

**Step 2: Programming the Base Platform**

The Pico Computing M-505-K325T card comes preconfigured from the factory. For this reason, no programming is required.
Step 3: Deploying SDAccel Applications

Each SDAccel™ application is compiled targeting a specific acceleration card. A deployment script is generated for each application that will:

1. Copy the necessary runtime files to the target directory specified with the `-d` switch.
2. Compile and install Linux kernel device drivers unless the `-k no` switch is specified.
3. Install the firmware files to the Linux firmware area.
4. Generate a `setup.sh` or `setup.csh` environment script which can be used to set up the correct environment.

Follow the instructions below to run the deployment script:

1. **Navigate to:** `<Solution directory>/pkg/pcie/`
2. As a user with root or sudo level access, run `. /install.sh -d <install_dir>"
Lab 2: Compiling and Optimizing the Application

The SDAccel™ development environment enables a programmer to quickly iterate through changes in an OpenCL™ API 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 3–1: 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* (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:

- For Alpha Data: `<user directory>/getting_started/alpha_data`
- For Pico: `<user_directory>/getting_started/pico`
Chapter 3: Lab 2: Compiling and Optimizing the Application

Step 1: Compiling and Running the Baseline Application

The SDAccel™ development environment 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.

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 60-380] 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: TAGGCAAGACCACCTTAGCTCTACAAACGCTAGCCTTTTGCCAAAGCAGATCGGCCCGCCCATCACT \ AGTGGGACTATCC
Input sequence2: TAATGGGAACACCTGCTGCAATCGGATCGTTGCAGCGGTAATGTGTCGGTATATGC-GAGTAGGGTAATCCA \ AACGTCCCATGTC

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

Align sequence1: T-A-GGCAAGACCACCTTAGCTCTACAAACGCTAGCCTTTTGCCAAAGCAGATCGGCCCGCCCATCACT \ GG----CC---CG-CCCAT
Align sequence2: TAATGGGAACACCTGCTGCAATCGGATCGTTGCAGCGGTAATGTGTCGGTATATGC-GAGTAGGGTAATCCA \ AACGTCCCATGTC

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

Step 2: Optimizing the Kernel Using Attributes

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

**NOTE:** For more information on the system estimate report, see System Estimate Report.
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:
   - If using the Alpha Data ADM-PCIE-7V3/VX690T-2 card: cd <user_directory>/getting_started/alpha_data
   - If using the Pico Computing EX-400 backplane card with the M-505-K325T module: cd <user_directory>/getting_started/pico

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.

```
Design Name: pipelined_solution
Target Platform: 1ddr
Target Board: Target Clock: 200MHz

Kernel Summary
Total number of kernels: 1

<table>
<thead>
<tr>
<th>Kernel Name</th>
<th>Type</th>
<th>Target</th>
<th>OpenCL Library</th>
<th>Compute Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>smithwaterman</td>
<td>clc</td>
<td>fpga0:OCL_REGION_0</td>
<td>test</td>
<td>1</td>
</tr>
</tbody>
</table>

OpenCL Binary - test
Kernels mapped to - clc_region

Timing Information (MHz)
<table>
<thead>
<tr>
<th>Compute Unit</th>
<th>Kernel Name</th>
<th>Target Frequency</th>
<th>Estimated Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>K1</td>
<td>smithwaterman</td>
<td>239.808</td>
<td>233.645</td>
</tr>
</tbody>
</table>

Latency Information (clock cycles)
<table>
<thead>
<tr>
<th>Compute Unit</th>
<th>Kernel Name</th>
<th>Start Interval</th>
<th>Best Case</th>
<th>Avg Case</th>
<th>Worst Case</th>
</tr>
</thead>
<tbody>
<tr>
<td>K1</td>
<td>smithwaterman</td>
<td>21815</td>
<td>21814</td>
<td>21814</td>
<td>21814</td>
</tr>
</tbody>
</table>

Area Information
<table>
<thead>
<tr>
<th>Compute Unit</th>
<th>Kernel Name</th>
<th>FF</th>
<th>LUT</th>
<th>DSP</th>
<th>BRAM</th>
</tr>
</thead>
<tbody>
<tr>
<td>K1</td>
<td>smithwaterman</td>
<td>2072</td>
<td>3157</td>
<td>1</td>
<td>9</td>
</tr>
</tbody>
</table>
```
## 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.

<table>
<thead>
<tr>
<th>Design Name: baseline_solution</th>
<th>Target Platform: 1ddr</th>
<th>Target Board:</th>
<th>Target Clock: 200MHz</th>
</tr>
</thead>
</table>

### 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)**

<table>
<thead>
<tr>
<th>Compute Unit</th>
<th>Kernel Name</th>
<th>Target Frequency</th>
<th>Estimated Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>K1</td>
<td>smithwaterman</td>
<td>239.808</td>
<td>273.973</td>
</tr>
</tbody>
</table>

**Latency Information (clock cycles)**

<table>
<thead>
<tr>
<th>Compute Unit</th>
<th>Kernel Name</th>
<th>Start Interval</th>
<th>Best Case</th>
<th>Avg Case</th>
<th>Worst Case</th>
</tr>
</thead>
<tbody>
<tr>
<td>K1</td>
<td>smithwaterman</td>
<td>166254 - 209094</td>
<td>187673</td>
<td>209093</td>
<td></td>
</tr>
</tbody>
</table>

**Area Information**

<table>
<thead>
<tr>
<th>Compute Unit</th>
<th>Kernel Name</th>
<th>FF</th>
<th>LUT</th>
<th>DSP</th>
<th>BRAM</th>
</tr>
</thead>
<tbody>
<tr>
<td>K1</td>
<td>smithwaterman</td>
<td>1771</td>
<td>2911</td>
<td>1</td>
<td>9</td>
</tr>
</tbody>
</table>
Chapter 3: Lab 2: Compiling and Optimizing the Application

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.

```c
#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 vall = northwest + match;
    if (vall > val) {
        val = vall;
        dir = NORTH_WEST;
    }
```

Step 3: 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/pkg/pcie`

2. If the target board is available on the system where the SDAccel environment is installed, do the following:
   a. Go into the `board_compilation/pkg/pcie` directory:
      ```
      cd board_compilation/pkg/pcie
      ```
   b. Run the deployment script `install.sh`:
      ```
      ./install.sh <install_dir>
      ```
   c. Source the generated `setup.sh` or `setup.csh` script file to get the system ready to run an application on the card:
      ```
      source setupenv.sh
      ```
   d. Run the application:
      ```
      ./board_compilation_project.exe -k test.xclbin
      ```
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

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 2015 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.