Daedalus/Daedalus^{RT} User Manual

The Daedalus/Daedalus^{RT} Team csartem@liacs.nl

September 10, 2012

Contents

1	Ove	rview 5
	1.1	Automatic Parallelization
	1.2	Design Space Exploration
	1.3	Hard-Real-Time Schedulability Analysis
	1.4	System Synthesis
2	Inst	alling Daedalus/Daedalus ^{RT}
	2.1	Prerequisites
	2.2	Obtaining the Frameworks
	2.3	Installing the Daedalus Framework
	2.4	Installing the Daedalus ^{RT} Framework
	2.5	Installing the individual tools
		2.5.1 Installing the PNgen Compiler
		2.5.2 Installing pntools
		2.5.3 Installing Sesame
		2.5.4 Installing darts
		2.5.5 Installing ESPAM
3		ng Daedalus 11
	3.1	Example I: Single Application - FPGA Backend 11
		3.1.1 Using the auto-run script
		3.1.2 Running the Daedalus Framework manually
		3.1.3 Building the Project using Xilinx Tools
		3.1.4 Programming the FPGA and Running the Application
	3.2	Example II: Sobel filter - SystemC Backend
		3.2.1 Using the auto-run script
		3.2.2 Running the Timed SystemC Backend Manually 20
4	Usin	ng Daedalus ^{RT} 21
	4.1	Example I: Multiple Applications - FPGA Backend
		4.1.1 Using the auto-run script
		4.1.2 Running the Daedalus ^{RT} Framework Manually
		4.1.3 Building and Running the Generated Project
5	Con	tributing to Daedalus/Daedalus ^{RT} 26
-	5.1	PNgen
		5.1.1 Building PNgen from the git repository
	5.2	pntools
	2.2	5.2.1 Building phtools from the git repository
	5.3	darts
	5.4	ESPAM
	5.1	5.4.1 Building ESPAM from the git repository

Table 1: Revisions of this user manual				
Date	Notes			
2012-09-10	Mohamed A. Bamakhrama and Jiali Teddy Zhai	Initial release		

Table 1: Revisions of this user manual

Introduction

This document serves as a manual for Daedalus and Daedalus^{RT} *users* and *developers*. Chapter 1 gives an overview of the frameworks. Chapter 2 provides instructions on how to install them. Chapter 3 explains how to use the Daedalus framework, while Chapter 4 explains how to use the Daedalus^{RT} framework. Chapter 5 explains how to contribute to the development of the tools.

Acknowledgments

The work on Daedalus framework has been supported by the following Dutch and European organizations: Netherlands Organisation for Scientific Research (NWO), Technology Foundation STW, European Union (EU) Seventh Framework Programme (FP7), CATRENE MEDEA+, and others.

List of Contributors

Many people have contributed over the years to the Daedalus/Daedalus^{RT} frameworks. Below we provide an incomplete list of the contributors (in alphabetical order). We would like to apologize in advance from anyone whom we forgot to add his/her name.

Mohamed Bamakhrama, Ed Deprettere, Cagkan Erbas, Ji Gu, Sven van Haastregt, Kai Huang, Joris Huizer, Dmitry Nadezhkin, Hristo Nikolov, Andy Pimentel, Simon Polstra, Todor Stefanov, Ying Tao, Mark Thompson, Sven Verdoolaege, Jiali Teddy Zhai, Wei Zhong

Overview

Daedalus [1, 2] is a system-level design flow for designing multiprocessor systems-on-chip (MPSoC) running embedded streaming applications.



Figure 1.1: Overview of the Daedalus framework

There is also a variant of Daedalus, called Daedalus^{RT} [3], which is targeted for designing hard-real-time embedded streaming systems. Daedalus^{RT} derives Cyclo-Static Data Flow (CSDF) [4] graphs equivalent to PPNs used in the Daedalus framework, and then uses hard-real-time multiprocessor scheduling theory in the Design Space Exploration (DSE) phase to determine the platform size and the mapping of tasks to processors.



Figure 1.2: Overview of the Daedalus^{RT} framework

In the following sections, we will describe in details each phase and the associated tools implementing it.

1.1 Automatic Parallelization

The first phase in the flow is the automatic parallelization shown in Figure 1.1. The PNgen compiler [5] is used to derive a parallel specification of the applications in the form of Polyhedral Process Networks (PPN, [5]) from a C program. The input to the PNgen compiler is a set of C files. The top-level C file has to conform to a constrained form known as *static affine nested loop programs* (*SANLP*, [5]). This constrained form allows to determine the data dependencies, and in turn, derive a parallel specification of the program in the form of Polyhedral Process Networks (PPN, [5]). An example of a SANLP program is shown in Figure 1.3. The program shown in Figure 1.3 implements the edge detection Sobel filter. This constrained form does not apply to the set of C files implementing all functions in the top-level C file. For example for the Sobel filter, the implementation of functions readPixel, gradient, absVal and writePixel may contain arbitrary C code and they are stored in a header file called sobel_func.h.

1.2 Design Space Exploration

The second phase is the design space exploration which accepts as an input a set of PPNs (and their equivalent CSDF graphs in case of Daedalus^{RT}). Daedalus uses the Sesame framework [6] to perform the DSE. The timing guarantees provided by the resulting platform are best-effort.

1.3 Hard-Real-Time Schedulability Analysis

Once designing a hard-real-time embedded streaming systems using Daedalus^{RT} is desired, the second phase is replaced by hard-real-time schedulability analysis [3]. This phase contains *analysis model derivation* and *analysis* as shown in Figure 1.2. pntools is used to derive analysis model in the form of CSDF. darts toolbox performs analysis based on hard-real-time multiprocessor scheduling theory to derive the minimum number of processors needed to schedule the applications while guaranteeing a certain throughput, together with the mapping of tasks to processors.

1.4 System Synthesis

The last step is the system synthesis which is implemented using the ESPAM tool [7]. The inputs to this step are: 1) a set of PPNs, 2) the platform specification, and 3) the mapping specification. After that After that, ESPAM generates an MPSoC platform that can run the parallelized applications. The generated platform consists of the hardware specifications together with the parallel C code.

```
#include "sobel_func.h"
#define N 450 // image width
#define M 275 // image height
int main(void)
{
   int i, j;
   static int image[M][N];
   static int Jx[M][N];
   static int Jy[M][N];
   static int av[M][N];
   for (j=1; j <= M; j++) {</pre>
      for (i=1; i <= N; i++) {</pre>
         readPixel(&image[j][i]);
      }
   }
   for (j=2; j <= M-1; j++) {</pre>
      for (i=2; i <= N-1; i++) {</pre>
           gradient( &image[j-1][i-1], &image[j][i-1], &image[j+1][i-1],
                      &image[j-1][i+1], &image[j][i+1], &image[j+1][i+1], &Jx[j][i] );
      }
   }
   for (j=2; j <= M-1; j++) {
      for (i=2; i <= N-1; i++) {</pre>
           gradient( &image[j-1][i-1], &image[j-1][i], &image[j-1][i+1],
                     &image[j+1][i-1], &image[j+1][i], &image[j+1][i+1], &Jy[j][i] );
      }
   }
   for (j=2; j <= M-1; j++) {</pre>
      for (i=2; i <= N-1; i++) {</pre>
           absVal( &Jx[j][i], &Jy[j][i], &av[j][i] );
      }
   }
   for (j=2; j <= M-1; j++) {</pre>
      for (i=2; i <= N-1; i++) {</pre>
          writePixel( &av[j][i] );
      }
   }
   return (0);
}
```

Figure 1.3: Example of a SANLP implementing the Sobel filter

Installing Daedalus/Daedalus^{RT}

In this chapter, we explain the steps to install Daedalus, Daedalus^{RT}, and their tools.

2.1 Prerequisites

Daedalus/Daedalus^{RT} are available only on Linux platforms and tested using the Linux distributions listed in Table 2.1. Installing Daedalus/Daedalus^{RT} requires the tools shown in Table 2.2 to be installed on the system.

Distribution	Version
Kubuntu	10.4 (32-bit)
Ubuntu	12.04 (64-bit)
openSUSE	11.4 (64-bit)
Fedora	10 (32-bit)

Table 2.1: Tested Linux distributions

Table 2.2:	Tools required	l to install Daedalus/Daedalus ^{RT}	

Tested version
4.5.2
2.67
3.81
2.2.6b
1.4.14
2.5.35
2.4.1
1.6.0_22
2.7.1+
1.12

The listed versions are the ones that have been tested while developing this manual. A higher/older version might cause compilation/installation errors in some cases (especially with GCC). The Daedalus framework installation consumes around 900 MB of disk space, while the Daedalus^{RT} framework installation consumes around 1.2 GB of disk space.

2.2 Obtaining the Frameworks

The Daedalus and Daedalus^{RT} frameworks together with the individual tools of the frameworks can be downloaded for free from http://daedalus.liacs.nl/. All the releases have names of the form release_yyyymmdd, where yyyymmdd is the date of the release. Therefore, the user can easily determine the latest release. Please note that we provide support only for problems related to the latest releases of the tools.

2.3 Installing the Daedalus Framework

We provide a script that installs the whole Daedalus framework automatically on a Linux machine. To run this script, open a command-line terminal and execute the commands shown in Figure 2.1.

```
mkdir daedalus
cd daedalus
wget http://daedalus.liacs.nl/daedalus/release_yyyymmdd/setup_daedalus.sh
chmod +x setup_daedalus.sh
./setup_daedalus.sh [-s systemc-path] [-j jdk-path] [-h] [-c] [-d]
```

Figure 2.1: Commands needed to install the Daedalus framework. yyyymmdd stands for the release date

The setup script accepts several options. The full list of options and usage can be shown by passing -h option which shows a help message.

2.4 Installing the Daedalus^{RT} Framework

We provide a script that installs the whole Daedalus^{RT} framework automatically on a Linux machine. To run this script, open a command-line terminal and execute the commands shown in Figure 2.2.

```
mkdir daedalus-rt
cd daedalus-rt
wget http://daedalus.liacs.nl/daedalus-rt/release_yyyymmdd/setup_daedalus-rt.sh
chmod +x setup_daedalus-rt.sh
./setup_daedalus-rt.sh [-s systemc-path] [-j jdk-path] [-h] [-c] [-d]
```

Figure 2.2: Commands needed to install the Daedalus^{RT} framework.

The setup script accepts several options. The full list of options and usage can be shown by passing -h option which shows a help message.

2.5 Installing the individual tools

We explain here the steps to install the individual tools of the Daedalus/Daedalus^{RT} frameworks.

2.5.1 Installing the PNgen Compiler

To install the PNgen compiler, open a command-line terminal and execute the commands shown in Figure 2.3.

```
mkdir pngen
cd pngen
wget http://daedalus.liacs.nl/pngen/release_yyyymmdd/setup_pngen.sh
chmod +x setup_pngen.sh
./setup_pngen.sh [clean] [nocheck]
```

Figure 2.3: Commands needed to install PNgen.

After invoking the commands, PNgen will be installed in pngen directory. The setup script (setup_pngen.sh) takes two optional arguments: clean which cleans the pngen directory from all the generated/installed files, and nocheck which ignores invoking make check for all the packages (leads to reduced installation time). When setup_pngen.sh is executed with no options, it builds PNgen from scratch and invokes make check for all the packages.

```
mkdir pntools
cd pntools
wget http://daedalus.liacs.nl/pntools/release_yyyymmdd/setup_pntools.sh
chmod +x setup_pntools.sh
./setup_pntools.sh [pngen_path] [clean]
```

Figure 2.4: Commands needed to install pntools

2.5.2 Installing pntools

pntools is a set of tools for working with PPNs. To install pntools, open a command-line terminal and execute the commands shown in Figure 2.4.

setup_pntools.sh takes two options. The first option is pngen_path which is the full path to an existing PNgen installation. If pngen_path is not specified, then the script will install PNgen from scratch. The second option is clean which cleans the pntools directory from all the generated/installed files.

2.5.3 Installing Sesame

TBD

2.5.4 Installing darts

darts (stands for Dataflow Analysis for Hard-Real-Time Scheduling) is a tool-set for performing hard-real-time multiprocessor scheduling analysis on CSDF graphs. To install it, open a command-line terminal and execute the commands shown in Figure 2.5.

```
mkdir darts
cd darts
wget http://daedalus.liacs.nl/darts/release_yyyymmdd/setup_darts.sh
chmod +x setup_darts.sh
./setup_darts.sh [clean]
```

Figure 2.5: Commands needed to install darts.

2.5.5 Installing ESPAM

ESPAM is the system synthesis tools of Daedalus and Daedalus^{RT}. To install it, open a command-line terminal and execute the commands shown in Figure 2.6 - j option can be used to pass the path to Java JDK installation, while

```
mkdir espam
cd espam
wget http://daedalus.liacs.nl/espam/release_yyyymmdd/setup_espam.sh
chmod +x setup_espam.sh
./setup_espam.sh [-j jdk-path] [-s systemc-path] [-c] [-h]
```

Figure 2.6: Commands needed to install ESPAM

the -s option can be used to pass the path to SystemC installation.

Using Daedalus

In this chapter, we provide several examples on how to use the Daedalus framework. Throughput this chapter, we assume that the variables shown in Table 3.1 store the full paths pointing to the corresponding tools in the Daedalus framework.

Tuble 5.11. Vultubles storing the full pulls to Duculus tools		
Variable	Meaning	
\$DAEDALUS_DIR	The full-path to the directory where Daedalus is installed	
\$PN_DIR	The full-path to the directory where PNgen is installed	
\$SESAME_DIR	The full-path to the directory where Sesame is installed	
\$ESPAM_DIR	The full-path to the directory where ESPAM is installed	

Table 3.1: Variables storing the full paths to Daedalus tools

The examples in this chapter refer to Daedalus release release_20120907. The source code of the examples can be found under \$DAEDALUS_DIR/examples.

3.1 Example I: Single Application - FPGA Backend

In this example, we show a step-by-step example of implementing the Sobel filter shown in Figure 1.3 on an FPGA-based MPSoC. The MPSoC platform consists of three Xilinx MicroBlaze processors running on Xilinx ML605 FPGA board.

3.1.1 Using the auto-run script

In the directory where Daedalus is installed, a script called daedalus.sh can be found at the following location: \$DAEDALUS_DIR/scripts/daedalus.sh. This script can be used to invoke all the different components of Daedalus automatically. To run the Daedalus flow on the Sobel example, which can be found also under \$DAEDALUS_DIR/examples/sobel, execute the following command:

```
$DAEDALUS_DIR/scripts/daedalus.sh -p $DAEDALUS_DIR/examples/sobel -o fpga
```

The command in Figure 3.1.1 will generate a directory called parallel under the \$DAEDALUS_DIR/examples/sobel directory. The parallel directory contains the FPGA project in a directory called sobel. Now, proceed to Section 3.1.3 for instructions on how to build the generated project using Xilinx tools.

3.1.2 Running the Daedalus Framework manually

For the reader interested in understanding the inner workings of the Daedalus flow, we provide here a detailed description of the commands executed by the auto-run script. The user can always refer to the auto-run script to see the exact steps executed by the flow.

Running PNgen

First, copy the contents of directory \$DAEDALUS_DIR/examples/sobel/sequential into a new directory called \$DAEDALUS_DIR/examples/sobel/parallel. Inside the parallel directory, make a new directory called func_code and copy the following files into it: sobel_func.h and sobel_func.cpp. Then, to produce the PPN specifications, open a command-line terminal and execute the commands shown in Figure 3.1.

cd parallel
\$PN_DIR/bin/c2pdg -func main sobel.c
\$PN_DIR/bin/pn < sobel.yaml > sobel-pn.yaml
\$PN_DIR/bin/pn2adg --xml < sobel-pn.yaml > sobel.kpn

Figure 3.1: Commands to generate the PPN of the Sobel example

The resulting sobel.kpn file is an XML file containing the PPN representation of the application. This file represents the input to ESPAM. The resulting PPN can be visualized using the dotty tool as shown in Figure 3.2.

\$PN_DIR/bin/pn2dot < sobel-pn.yaml > sobel-pn.dot
dotty sobel-pn.dot

Figure 3.2: Visualizing the resulting PPN

Specifying Platform and Mapping files

Daedalus provides the ability to either specify the platform/mapping specifications manually, or derive them using Sesame. We start by showing the first option. We can manually specify the platform specification (sobel.pla) to be used and the assignment (sobel.map) of PPN processes to processing resources, i.e., Xilinx MicroBlaze (MB) in this case. In the next subsection, we show how to generate sobel.pla and sobel.map files automatically using Sesame. Figure 3.3 shows an example of a platform file containing three MBs interconnected via AMBA AXI crossbar. Here we explain the important attributes to be specified in the platform file and their meaning.

- MB: type of processor, Xilinx MicroBlaze (MB);
- data_memory="65536": size of local data memory in bytes, 65536 in this case;
- program_memory="65536": size of local program memory in bytes, 65536 in this case;
- <network name="CS" type="AXICrossbarSwitch">: type of interconnection; AXICrossbarSwitch denotes AMBA AXI crossbar. Otherwise, if no network is specified, Xilinx Fast Simplex Link (FSL) FIFOs are assumed to interconnect MBs. FSL FIFOs are automatically instantiated between MBs according to the given mapping file explained later on;
- <host_interface name="HOST_IF" type="ML605" interface="UART">: host interface is essentially another MB with minimum resource usage in terms of data and program memory (8 KB each). The purpose of having host interface is to communicate with host PC, e.g., after host PC transfers some input data to the DDR and then notifies host interact to start MBs specified in the platform file. It also specifies the FPGA board in use via attribute type="ML605". Besides ML605, XUPV5-LX110T board is also supported and can be used via attribute type="XUPV5-LX110T". Attribute interface="UART" indicates that the FPGA board communicates with host PC through UART.

Once we have the platform file, we explain how to specify a mapping file. Figure 3.4 shows an example of a mapping file. The two instances of gradient (PPN processes ND_1 and ND_2) are mapped to MB mb_2 and mb_3. All the other PPN processes are mapped to MB mb_1. On MB mb_1, all PPN processes are statically scheduled according to the schedule computed in PNgen.

 schedule_type: Besides statically schedule all PPN processes mapped onto the same MB, dynamic scheduling of the PPN processes is also supported. Dynamic scheduling includes round-robin and round-robin with yielding using Xilinx Xilkernel.

Deriving Platform and Mapping files using Sesame

TBD.

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE platform PUBLIC "-//LIACS//DTD ESPAM 1//EN"
"http://www.liacs.nl/~cserc/dtd/espam_1.dtd">
<platform name="myPlatform">
   <processor name="mb_1" type="MB" data_memory="65536" program_memory="65536">
     <port name="I0_1" />
   </processor>
   <processor name="mb_2" type="MB" data_memory="65536" program_memory="65536">
      <port name="IO_1" />
   </processor>
   <processor name="mb_3" type="MB" data_memory="65536" program_memory="65536">
      <port name="IO_1" />
   </processor>
   <network name="CS" type="AXICrossbarSwitch">
      <port name="IO_1" />
      <port name="I0_2" />
      <port name="I0_3" />
   </network>
   <host_interface name="HOST_IF" type="ML605" interface="UART">
   </host_interface>
   <link name="BUS1">
     <resource name="mb_1" port="I0_1" />
      <resource name="CS" port="IO_1" />
   </link>
   <link name="BUS2">
     <resource name="mb_2" port="I0_1" />
     <resource name="CS" port="IO_2" />
   </link>
    <link name="BUS3">
     <resource name="mb_3" port="I0_1" />
     <resource name="CS" port="I0_3" />
   </link>
 </platform>
```

Figure 3.3: An example of a platform file.

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE mapping PUBLIC "-//LIACS//DTD ESPAM 1//EN"
"http://www.liacs.nl/~cserc/dtd/espam_1.dtd">
<mapping name="myMapping">
   <processor name="mb_1">
       <process name="ND_0" />
       <process name="ND_3" />
       <process name="ND_4" />
   </processor>
   <processor name="mb_2">
       <process name="ND_1" />
   </processor>
   <processor name="mb_3">
       <process name="ND_2" />
   </processor>
</mapping>
```

Figure 3.4: An example of a mapping file.

Running ESPAM

To run ESPAM and generate the MPSoC implementation, open a terminal and execute the command shown in Figure 3.5.

After invoking ESPAM, a directory called sobel will be generated which contains a Xilinx Platform Stu-

```
$ESPAM_DIR/bin/espam --adg sobel.kpn --platform ../sobel.pla --mapping ../sobel.map \
--xps --libxps $ESPAM_DIR/src/espam/libXPS/ --sdk --libsdk $ESPAM_DIR/src/espam/libSDK \
--funcCodePath func_code
```

Figure 3.5: Running the system synthesis tool. \setminus is used to break the command over multiple lines

dio (XPS) project containing the MPSoC implementation. Currently, ESPAM supports generating XPS projects compatible with XPS 13.2 and later.

3.1.3 Building the Project using Xilinx Tools

After you generate the Xilinx Platform Studio (XPS) project using ESPAM through either the auto-run script or manual steps, you will find the generated project under \$DAEDALUS_DIR/examples/sobel/parallel/sobel. Open the project using Xilinx Platform Studio 13.2 or later and perform the FPGA synthesis. After the synthesis is completed and you have a bitstream, export the project to Xilinx SDK. Upon exporting the project to SDK, SDK will launch. You need to perform the following steps to import and build the software projects.

1. Upon launching SDK, you will prompted for the workspace location as shown in Figure 3.6. Click on "Browse" and set the location to be \$DAEDALUS_DIR/examples/sobel/parallel/sobel/SDK/. Then, click on "OK"



Figure 3.6: Step 1: Setting the workspace location

- 2. Now, you should see a window like the one shown in Figure 3.7.
- 3. In SDK, click on "File" > "Import" as shown in Figure 3.8
- 4. A new window will appear as shown in Figure 3.9. Select "General" ¿ "Existing Projects into Workspace". Then, click on "Next"
- 5. You will see a window like the one shown in Figure 3.10. Click on "Browse" to select the root directory which contains the SDK projects. This directory is \$DAEDALUS_DIR/examples/sobel/parallel/sobel/SDK/.
- 6. After setting the directory correctly, you should see a window like the one shown in Figure 3.11. Now, click on "Finish"
- 7. After clicking on "Finish" in the previous step, SDK will build all the imported projects. However, there are a few issues that we need to fix manually. First, the host interface project (host_if) will show an error. This is because the project contains the files needed to interface with both of AXI and PLB system. To get rid of this error, you need to delete the main_PLB.cpp file as shown in Figure 3.12.
- 8. Next, we need to set up the linker scripts and debugging/optimizations options for the P_* projects. Note: Correct ELF files will NOT be generated without performing this step. To change the optimization/debugging options, right-click on each P_* project and select "C/C++ Build Settings" as shown in Figure 3.13

😣 🖱 🗇 C/C++ - sobel_hw_platform/system.x	ml - Eclipse Platform		
File Edit Source Refactor Navigate Search	Project Run Xilinx Tools Window Help		
🗅 - 🔛 🗠 👌 🗖 🏭 📓 🚳 - 🚳	⁸ ▼ G▼ 또 한▼ 弊▼ Q▼ Q _▼ Ø		🗈 🖬 C/C++ 🏊 F 🎽
🕒 Project Explorer 😫 📄 😫 🍸 🗖 🗖	🕞 system.xml 🛿	- 0	😤 Outline 😫 🛞 Make Target 📃 🗖
▼ 🥵 sobel_hw_platform	sobel_hw_platform Hardware Platform Specification		An outline is not available.
🗃 system.xml	Design Information		
	Target FPGA Device: xc6vlx240t Created With: EDK 13.2 Created On: Tue Jul 2416:29:10 2012		
	Address Map for processor mb_1		
	fn_ctrl_P1 0:0000000 0x59000001 LMB_CTRL_CM_mb_1 0x6000000 0x400001111 DCTRL_BRAM1_mb_1 0x0000000 0x00011111 mb_1_lnct0x00000 0x1001111 mb_1_lnct0x00000 0x1001111 hotJ[mb_BC332_ulrt 0x1000000 0x1001111 DDR3_SDRAM 0x8000000 0x4111111 AXI_CTRL_CM_mb_1 0x8000000 0x8011111 AXI_CTRL_CM_mb_2 0x8000000 0x8011111 mb_1_axi2axi_connector 0x8000000 0x81111111 Address Map for processor mb_2 fn_ctrl P2 0x59000000 0x5900001 LMB_CTRL_CM_mb_2 0x8000000 0x8011111 pCTRL_BRAM1_mb_2 0x0000000 0x001111 pCTRL_BRAM1_mb_2 0x0000000 0x001111 mb_1 axi2avi_connector 0x8000000 0x8001111 pCTRL_BRAM1_mb_2 0x0000000 0x0001111 pCTRL_BRAM1_mb_2 0x0000000 0x0001111 mb_2 lnct0x1000000 0x1000111		
	Overview Source		
	🗄 Problems 🖉 Tasks Console 😫 💭 Properties 🖉 Terminal 1		
	SOKLOg [B:32:27 INFO : Processing command line option -hwspec /home/mohamed/workspace/codeple	x/daedalus/examples/sobel/parallel/sobe	l/SDK/SDK_Export/hw/system.xml.
□ *			

Figure 3.7: Step 2: The imported hardware platform in SDK

See C/C++ - sobel_hw_platform/system.xml - Eclipse Platform					
File Edit Source Refactor Navigat	e Search Project Run XilinxTools Window Help				
New	Shift+Alt+N · - 중국 방국 방국 외국 🚱 🖉 🗐 🗊 한 성국 학교 수국 수국	🗈 🖬 C/C++ 🔤 F »			
Open File					
Close	Ctrl+w stem.xml 🛛	🗄 Outline 🛿 🛞 Make Target 👘 🗖			
Close All	Shift-Circlew pel hw platform Hardware Platform Specification	An outline is not available.			
Save	CTU-5 sign Information				
Save As Save All	shifternets rget FPGA Device: xc6vix240t				
	Created With: EDK 13.2				
Revert	Created On: Tue Jul 24 16:29:10 2012				
Move	the second se				
Rename	ress Map for processor mb_1				
Refresh	F5 fin_ctrl_P1 0xf900000 0xf900000f				
Convert Line Delimiters To	LMB_CTRL_CM_mb_1 0xe000000 0xe000fff				
Print	DCTRL_BRAM1_mb_1 0x000000 0x0001fff				
	CETTEP PCTRL_BRAM1_mb_1 0x0000000 0x0001fff mb 1 intc 0xf2000000 0xf200fff				
Switch Workspace	mb_timer 0x1000000 0x100fff				
Restart	host if mb RS232 Uart 0xb1000000 0xb100fff				
Import	DDR3 SDRAM 0xa0000000 0xaffffff				
Export	AXI_CTRL_CM_mb_1 0x80010000 0x8001ffff				
Properties	Alt+Enter AXI_CTRL_CM_mb_2 0x80020000 0x8002fff				
	AXI_CTRL_CM_mb_3 0x80030000 0x8003ffff				
<u>1</u> system.xml [sobel_hw_platform]	>_1_ddr_axi2axi_connector 0xa000000 0xbffffff				
Exit	mb_1_axi2axi_connector 0x80000000 0x8111111				
	Address Map for processor mb_2.				
	fin ctrl P2 0x1000001x0 1000001				
	LMB_CTRL_CM_mb_2 0xe0000000 0xe000fff				
	DCTRL BRAM1 mb 2 0x0000000 0x0001fff				
	PCTRL_BRAM1_mb_2 0x0000000 0x0001ffff				
	mb_2_intc 0xf2000000 0xf200ffff				
	Overview Source				
	💽 Problems 🧟 Tasks 📮 Console 😫 💭 Properties 🖉 Terminal 1				
	SDKLog				
	16:32:27 INF0 : Processing command line option -hwspec /home/mohamed/workspace/codeplex/daedalus/examples/sobel/parallel/sobe	l/SDK/SDK_Export/hw/system.xml.			
L					
] 0.					

Figure 3.8: Step 3

- 9. Now, a new window will appear as shown in Figure 3.14. Click on "Optimization" and change it to Os. Then, click on Debugging and change it to None.
- 10. Now, after changing the optimization/debugging options for all the projects, you need to set up the correct linker script for each P_* project. This can be done by right-click on each P_* project and selecting "Generate Linker Script". After that, you will see a new window as the one shown in Figure 3.15. Change the location of the Stack and Heap from LMB_CTRL_CM_mb_* into PCTRL_BRAM1_mb_*_DCTRL_BRAM1_mb_*. Moreover, increase the Stack and Heap size into a reasonable value (e.g., 8 KB). Then, click on "Generate"
- 11. Now, we are done and the correct ELF files are generated

😣 🗈 Import					
Select					
Create new projects from an archive file or directory.					
<u>S</u> elect an import source:					
type filter text	(1)				
▼ 🗁 General 및 Archive File					
🖆 Existing Projects into Workspace 💦					
🗀 File System					
💷 Preferences					
▶					
 ▶					
 ▷ ▷ C/C++ ▷ ▷ Remote Systems ▷ ▷ Run/Debug ▷ ▷ Team 					
? < Back Next > Cancel	Finish				

Figure 3.9: Step 4

😣 🗊 Import	
Import Projects	
Select a directory to search for existing Eclipse projects.	
Select roo <u>t</u> directory:	B <u>r</u> owse
○ Select <u>a</u> rchive file:	Browse
Projects:	
	<u>S</u> elect All
	Deselect All
	R <u>e</u> fresh
Copy projects into workspace	
Working sets	
Add project to working sets	
Working sets:	Select
	Detectin
(?) < Back Next > Cancel	Finish
<pre></pre>	FINISN

Figure 3.10: Step 5

3.1.4 Programming the FPGA and Running the Application

Now, go back to XPS and perform the following steps:

😣 🗈 Import			
Import Projects			
A Some projects cannot be imported because they already exist in the workspace			
Select roo <u>t</u> directory: aedalus/examples/sobel/parallel/sobel/SDK Select archive file:	Browse		
Projects:	Drowsen		
BSP_host_if (/home/mohamed/workspace/codeplex/daedalus/e>	Select All		
 BSP_P_1 (/home/mohamed/workspace/codeplex/daedalus/exam BSP_P_2 (/home/mohamed/workspace/codeplex/daedalus/exam 	Deselect All		
BSP_P_3 (/home/mohamed/workspace/codeplex/daedalus/exam	Refresh		
 kost_if (/home/mohamed/workspace/codeplex/daedalus/examp P_1 (/home/mohamed/workspace/codeplex/daedalus/examples, P_2 (/home/mohamed/workspace/codeplex/daedalus/examples, P_3 (/home/mohamed/workspace/codeplex/daedalus/examples, Copy projects into workspace 			
Working sets			
Add project to working sets			
Working sets:	Select		
? < Back Next > Cancel	Finish		

Figure 3.11: Step 6

C/C++ - sobel_hw_platform/s File Edit Source Refactor Navigate	system.xml - Eclipse Platform • Search Project Run Xilinx Tools W	indow Help	
		○ - 오. · · · · · · · · · · · · · · · · · ·	🖬 🛱 C/C++ 🍙 F »
	▼ ■ 🛙 🍙 system.xml 🛿		
BSP_host_if		dware Platform Specification	An outline is not available.
BSP_P_1 BSP_P_2	Design Information		
▶ 🏙 BSP_P_3	Target FPGA Device: xc6vlx24		
▼ 💕 host_if	Created With: EDK 13.2 Created On: Tue Jul 2		
 M Includes Debug 	Created On. The Sur 2	4 10.23.10 2012	
Debug Main_AXI.cpp	Address Map for processor m	1 <u>b_1</u>	
Ramain PLB.cop	fin_ctrl_P1	0xf9000000 0xf900000f	
▶ 25 P_1	New	, te0000000 0xe000ffff (0000000 0x0001ffff	
▶ 2 P_2	<u>O</u> pen	00000000 0x0001fff	
▶ 2 P_3	Open Wit <u>h</u>	*:f2000000 0xf200ffff	
🖲 sobel_hw_platform	Copy Ctrl+	C 11010000 0x1100FFF	
🚡 system.xml		b1000000 0xb100ffff	
	Delete Delet	100000008 00000000000000000000000000000	
	Mo <u>v</u> e	11112008x0 00002008	
	Rena <u>m</u> e F	2 180030000 0x8003ffff	
	Import	ca000000xbfffffff 8000000x0	
	Export		
	Refresh F	⁵⁵ . ²	
	Index	10000001x0 0000001x ,	
	Make Targets	te0000000 0xe000ffff	
	Resource Configurations	10000000 00000000000000000000000000000	
	Clean Selected File(s)	12000000 0xf200ffff	
	Build Selected <u>F</u> ile(s)		
	Team	2 🕮 Properties 🧬 Terminal 1	♦
	Comp <u>a</u> re With	, 22 Properties & Terminal 1	
	Rep <u>l</u> ace With	vuariussoo_senubyte was not dectared in this scope	
	Properties Alt+Ente	ZVJartNs550 SendByte' was not declared in this scope XUartNs550 SendByte' was not declared in this scope	
		: 'XUartNs550 SendByte' was not declared in this scope	
		: 'XUartNs559 SendByte' was not declared in this scope	
/main PLB.cpp:159: error: 'XUartN5559 SendByte' was not declared in this scope /main PLB.cpp:34: warning: unused variable 'APAMM' /main PLB.cpp:52: warning: unused variable 'time1' make: *** (main PLB.c] Error 1			
	maker [main_reb.0] Erro		
P → P /host_if/main_PLB.cpp			

Figure 3.12: Step 7: Removing the un-used PLB file

 Associate the host interface processor (host_if_mb) with the correct ELF file which can be found under \$DAEDALUS_DIR/examples/sobel/parallel/sobel/SDK/host_if/Debug/host_if.elf. This is illustrated in Figure 3.16



Figure 3.13: Step 8: Changing the build settings

Resource Builders C/C++ Build Build Variables Discovery Options	Configuration: Debug [Active]	Build Artifact 🗟 Binary Pars	ers 🛛 Error Parsers	CManage Configurations
Environment Logging	MicroBlaze gcc assembler General	Optimization Level	Optimize for size (-Os)	*
Logging Settings Tool Chain Editor ▶ C/C++ General Project References Run/Debug Settings	 SimicroBlaze g++ compiler Symbols Warnings Optimization Debugging Profiling Directories Miscellaneous Software Platform Processor Options Software Platform Processor Options Software Software Platform Ceneral Libraries Miscellaneous Libraries Miscellaneous Viscellaneous 	Other optimization flags		

Figure 3.14: Step 9: Changing optimization and debugging options

- 2. Run "BRAM INIT" which will update the bitstream with the ELF files.
- 3. Under etc directory in the XPS project directory, rename the file download_ML605.cmd to download.cmd.
- 4. Open a terminal in the directory \$DAEDALUS_DIR/examples/sobel/serial_m1605 and run make. This will build the program that will communicate with the FPGA from the computer side using UART.
- 5. Connect and start the FPGA board. Then, run the "Download bitstream to FPGA" command in XPS.
- 6. After the FPGA programming is done, run the command serial_m1605 which you built previously.

⊗ 🗊 Generate a linker script						
Generate linker script						
Control your application's memory map.	Carlo					
Control your application's memory map. Output Settings Project: P_1 Output Script: aedalus/examples/sobel/parallel/sobel/SDK/P_1/lscript.ld Browse Modify project build settings as follows: Set generated script on all project build configurations PCTRL_BRAM1_mb_1_DCTRL DX80000000 LMB_CTRL_CM_mb_1 DX80000000 AXI_CTRL_CM_mb_15_AXI_E Ox8000000 AXI_CTRL_CM_mb_2_S_AXI_E Ox80030000 64 KB AXI_CTRL_CM_mb_3_S_AXI_E Ox80030000 64 KB	Basic Advanced Place Code Sections in: PCTRL_BRAM1_mb_1_DCTRL_BRA ▼ Place Data Sections in: PCTRL_BRAM1_mb_1_DCTRL_BRA ▼ Place Heap and Stack in: PCTRL_BRAM1_mb_1_DCTRL_BRA ▼ Heap Size: 8 KB Stack Size: 8 KB					
Fixed Section Assignments	6					
3	Cancel Generate					

Figure 3.15: Step 10: Generating the correct linker script

Project Files Image: Second and Control of	oject 🛛 💮 🖉 🖉		Bus Interfaces Ports	Addresses	(B)	Bus Interface Filters
Project plies	atform	2 22222	Name Dus Name	ID Tune ID Merries		
Must Provide and Andrew Standard Cond Implementation Controls Rise activities and the standard Cond Implementation Controls Rise activities and the standard Cond Implementation Controls Rise activities and the standard Rise activities and the standard Rise activities and the standard Rise activities activitities activities activities activities activities acti		I IIIIBBBBBB				- ✓ Connected
UCF Rile-adapystem.ud/ IMPACT Comman Rile-etc/download.cmd Implementation Options Rile-etc/download.cmd Implementation Options Rile-etc/download.cmd Implementation Options Rile-etc/download.cmd Power (* * * * * * * * * * * * * * * * * * *		÷				- Vinconnected
Index Command His exclosed and information of the States of the State						By Bus Standard
<pre>biging options file: etc/ats_numbers.pt biging options file: etc/ats_numbers.pt b</pre>						- ✓ AXI
Bit field of the state of the s						- V LMB
<pre>bit of these s mb_1 s mb_2 s mb_</pre>		· · · · · · · · · · · · · · · · · · ·				Xilinx Point To Point
s mb_1 s mb_2 s mb_2 s mb_3 s hot, if mb postpations and state databased and our project option postpations postpa			- mb_3_ax	🙀 axi_inter 1.03.a		- V XIL BRAM
<pre>s mb_2 s mb_2 s mb_3 b mb_4 tf mb protect option Decision 2005 b mb st tf mb y10 2005 b mb st tf m</pre>			- DBUS m	🐈 lmb v10 2.00.b		
<pre> # mb_3 # bot; // nb # mb_2 // nb #</pre>			- DBUS m	🖕 lmb v10 2.00.b		XIL_MBDEBUG3
<pre># mb_3 # hot, "rob b hot, "rob in Executable in the contained at isolated at isolated at isolated at isolated in the contained at isolated at isolated at isolated at isolated in the contained at isolated a</pre>			- DBUS m	1mb v10 2.00.b		✓ XIL_MBTRACE2
B hot, if, mb						✓ User Defined
Imp Cecuciable: min becauciable: mi						By Interface Type
In Executable: In Executable: Project Option Device: xtolv22dH1156-1 Invitit: Expended Project Option Device: xtolv22dH1156-1 Invitit: Expended Project Option Device: xtolv22dH1156-1 Invitit: Expended Project Option Project						 ✓ Slaves
Project Options boxit/f		· · · · · · · · · · · · · · · · · · ·				 Masters
Device: xc0xl2d0ff135-1 Netlist: TopLed Precise: xc0xl2d0ff135-1 Preci	Project Options					 Master Slaves
Netlik: TopLevel * mb.2 * microbiaze 8.20.a * clasing * mb.2 * microbiaze 8.20.a * microbiaze 8.20.a * mb.2 * microbiaze 8.20.a * microbiaze 8.20.a * mb.2 * microbiaze 8.20.a * mb.2 * microbiaze 8.20.a * mb.2 * mb.3 * microbiaze 8.20.a * mb.3 * COM * Microbiaze 8.20.a * mb.3 * Boand 1 * mb.3 * mb.3 * COM * Microbiaze 8.20.a * mb.3 * Decipion * Vermin bin	- Device: xc6vlx240tff1156-1					 Monitors
<pre>chalog chalog chal</pre>	- Netlist: TopLevel					 Targets
P catalog model Bescription P Version (P Type F CEX Is natal Bescription (P Version (P Type F CEX Is natal F CEX Is natal Bescription (P Version (P Type F CEX Is natal F CEX Is nata						 Initiators
Constant						
excription IP Version IP Type = E EDK Install = Analog = Communication Low Speed = Communication Low Speed = Communication Low Speed = Debug = Debug						
Description (P Version	•					
E Constall Analog Analog Analog Analog Constall Analog Analog Constall	escription IP Type					
a Analog b Bus Add Add Add Add Add Add Add Add Add Ad						
s Bus and Bindge c Clock, Rest and Interrupt c Communication High-Speed c Communication Low-Speed c C C C C C C C C C C C C C C C C C C C						
e Cock, Reset and Interrupt e Communication High-Speed e Communication High-Speed e DMA and Timer e DMA		• • • • • • • • • • • • • • • • • • •				
i Communication High-Speed c Communication Low-Speed c Communication Low-		······································				
s Communication Lows. Speed b OMA and Timer b Consent Purpose IO c Control P						
Dekig PORA Reconfiguration Control Contro Control Control Control Control Con		· · · · · · · · · · · · · · · · · · ·	DCTRL_B	🙀 Imb_bra 3.00.b		
Bebug B			PCTRL_B	imb_bra 3.00.b		
a FRA Reconfiguration b General Pupoles IO a General Pupoles IO b Interprocessor Communication b Interprocesor Communicat			+ CM mb 1	🕁 bram bl 1.00.a		
s General Purpose 10 s in Manduer Stave Matter Stave Parent Charles & Concepted Unionanceted Minister is in Manduer Stave Matter Stave Matter Stave Manduer (Stave Parent Development Stave Parent		•	+ AXI CTR	👉 axi bra 1.02.a		
i Dodules interprocessor Communication interprocessorecommunication interprocessor Communication interprocessor						
a: Di Kodules	General Purpose IO	Master 🗢 Slave Master/Slave 🕨 Target <init< td=""><td>iator 👱 Connected OUnconnected</td><td>M Monitor</td><td></td><td></td></init<>	iator 👱 Connected OUnconnected	M Monitor		
Interpretendo Communication Image: Complete downloading bit file to device. INFOILIMENT: 177 - 177 : Completed downloading bit file to device. INFOILIMENT: 177 : 177 : Completed downloading bit file to device. INFOILIMENT: 177 :	IO Modules	Production License (paid) Clicense (eva	al) 🛸 Local 🚢 Pre Production 🖁	Beta 🗷 Development		
nskole piktro:IMPACT:579 - '2': Completed downloading bit file to device. piktro:IMPACT:579 - '2': Completed downloading bit file to device. piktro:IMPACT - '2': checking done pikdone. '2': Programming successfully.	Interprocessor Communication					
DINGCIBARCT373 - '2': Completed downloading bit file to device. DINGCIBARCT38 - '2': Forgamming completed successfully. DINGCIBARCT38 - '2': chogaramed successfully. '2': Forgamma successfully.		Design Summary	× 🗢	System Assembly View	×	
pINFOIPMEXT575 - '2': Completed downloading bit file to device. pINFOIPMEXT58 - '2': Programming complete successfully. pINFOIPMEXT - '2': checking done pindone. '2': Programming successfully.						_
<pre>DiMPOCIMPACT188 - '2': Programming completed successfully. DIMPOCIMPACT - '2': Checking done pindone. '2': Programmid successfully.</pre>						
DINFOIMMAT - '2': Checking done pindone. '2': Programmed successfully.						
'2': Programmed successfully.		117.				
	arapaca cime =					
Done!	Pone!					

Figure 3.16: Associating the host interface processor with its ELF file

7. You should now see the processed image being sent from the FPGA

3.2 Example II: Sobel filter - SystemC Backend

In this example, we show a step-by-step example of simulating the Sobel filter shown in Figure 1.3 using the timed SystemC backend of Daedalus. We assume that you configured the SystemC installation path correctly during the

installation phase (see Chapter 2).

WARNING: If you have used the auto-run script to run the FPGA backend in Section 3.1, then please be careful that using the auto-run script to run the SystemC backend will delete the parallel directory generated by the FPGA backend. Therefore, you have to either copy it or rename it.

3.2.1 Using the auto-run script

To invoke the Daedalus flow on the Sobel example using the timed SystemC backend, execute the following command:

\$DAEDALUS_DIR/scripts/daedalus.sh -p \$DAEDALUS_DIR/examples/sobel -o systemc-timed

The command in Figure 3.2.1 will generate a directory called parallel under the <code>\$DAEDALUS_DIR/examples/sobel</code> directory. Under the parallel directory, you will find another directory called <code>sobel_systemc</code> which contains the SystemC files. The script will also run automatically the simulation and present the resulting images and waveforms.

3.2.2 Running the Timed SystemC Backend Manually

To use the timed SystemC backend, you need to perform the same steps for parallelization as you have done for the FPGA backend in Section 3.1. The only difference is in invoking ESPAM. To invoke ESPAM with the SystemC backend, execute the command shown in Figure 3.2.2.

```
$ESPAM_DIR/bin/espam --adg sobel.kpn --platform sobel.pla --mapping sobel.map \
--systemc-timed --libsystemc $ESPAM_DIR/src/espam/libSystemC
```

A directory named \$DAEDALUS_DIR/examples/sobel/parallel/sobel_systemc will be generated. Copy all the header and images files together with sources to this directory. Then, invoke make and make run under sobel_systemc.

Using Daedalus^{RT}

In this chapter, we provide an example on how to use the Daedalus^{RT} framework. Throughput this chapter, we assume that the variables shown in Table 4.1 store the full paths pointing to the corresponding tools in the Daedalus^{RT} framework.

Variable	Meaning
<pre>\$DAEDALUS_RT_DIR</pre>	The full-path to the directory where Daedalus ^{RT} is installed
\$PN_DIR	The full-path to the directory where PNgen is installed
\$PNTOOLS_DIR	The full-path to the directory where pntools is installed
\$DARTS_DIR	The full-path to the directory where darts is installed
\$ESPAM_DIR	The full-path to the directory where ESPAM is installed

Table 4.1: Variables storing the full paths to Daedalus^{RT} tools

The examples in this chapter refer to Daedalus^{RT} release release_20120907. The source code of the examples can be found under \$DAEDALUS_RT_DIR/examples/synthetic.

4.1 Example I: Multiple Applications - FPGA Backend

In this example, we show a step-by-step example of implementing two synthetic applications (called pipeline and split_join) on an FPGA-based MPSoC. The MPSoC platform uses Xilinx MicroBlaze processors running on Xilinx ML605 FPGA board.

4.1.1 Using the auto-run script

In the directory where Daedalus^{RT} is installed, a script called daedalus-rt.sh can be found at the following location: $DAEDALUS_RT_DIR/scripts/daedalus-rt.sh$. This script can be used to run all the different parts of the flow automatically. To run the Daedalus^{RT} flow on the synthetic directory, which can be found also under $DAEDALUS_RT_DIR/scripts$, execute the following command:

\$DAEDALUS_RT_DIR/scripts/daedalus-rt.sh -p \$DAEDALUS_RT_DIR/examples/synthetic

The user will be asked to enter two parameters per application: 1) the period scaling factor, which is an integer multiplied by the minimum period of each actor in the application, and 2) the maximum token size (in 32-bit words) in the application. For both applications, use period scaling factor = 50, and maximum token size = 1. The command in Figure 4.1.1 will generate a directory called output under the \$DAEDALUS_DIR/examples/synthetic directory. The output directory contains the FPGA project in a directory called xps_project. Now, proceed to Section 4.1.3 for instructions on how to build and run the generated project.

4.1.2 Running the Daedalus^{RT} Framework Manually

Daedalus^{RT} shares the same steps for parallelization and system synthesis with Daedalus. Therefore, we show here only the steps specific for Daedalus^{RT}. The user can always refer to the auto-run script to see the exact steps executed by the flow.

Running pntools

Daedalus^{RT} replaces the DSE performed by Sesame with CSDF model derivation and hard-real-time multiprocessor schedulability analysis on the resulting CSDF graphs. The first step is to derive the CSDF model required by the analysis. To do that, parallelize the applications using the instructions given in Section 3.1.2, Then, open a terminal and execute the commands shown in Figure 4.1.

\$PN_DIR/bin/pn2adg < pipeline-pn.yaml > pipeline-adg.yaml
\$PNTOOLS_DIR/adg2csdf < pipeline-adg.yaml > pipeline.gph
\$PNTOOLS_DIR/adg2csdf -3 < pipeline-adg.yaml > pipeline.xml

Figure 4.1: Deriving CSDF from PPN. Steps shown only for pipeline

The resulting pipeline.gph and pipeline.xml files contain the CSDF graph representation. The .gph file contains the graph in a format derived from the StreamIt format, while the .xml file contains the graph in the XML format support by SDF^3 [8]. You have to repeat the steps shown in Figure 4.1 for the split_join application.

Running darts

darts allows deriving the platform and mapping specifications for a given set of CSDF graphs. To generate the platform and mapping specifications, open a terminal and execute the commands shown in Figure 4.2.

\$DARTS_DIR/PlatformGenerator.py -o . -m RM-FF -a 50 1 pipeline.gph -a 50 1 split_join.gph

Figure 4.2: Deriving the platform and mapping specifications

PlatformGenerator.py takes the following input arguments:

- \bullet \circ . which sets the directory where the platform and mapping specifications will be generated to the current directory
- The scheduling and allocation method. In Figure 4.2, it is set to fixed-priority preemptive scheduling with rate-monotonic priority assignment and first-fit allocation
- A set of CSDF graphs. In Figure 4.2, two graphs are passed: pipeline and split_join. Each graph is preceded by a period scaling factor (e.g., 50) and the maximum token size (in 32-bit words) in the application. The period scaling factor is an integer that gets multiplied by the minimum period of each actor in the application. For both applications, we use period scaling factor = 50, and maximum token size = 1.

After invoking PlatformGenerator.py, two files will be generated in \$OUTPUT_DIR: 1) platform.pla which contains the platform specification, and 2) mapping.map which contains the mapping specification. These two files serve as the input to ESPAM. Now, you can invoke ESPAM using the command shown in Figure 4.3.

\$ESPAM_DIR/bin/espam --adg pipeline.kpn --adg split_join.kpn --platform platform.pla \
--mapping mapping.map --xps --libxps \$ESPAM_DIR/src/espam/libXPS/ --sdk \
--libsdk \$ESPAM_DIR/src/espam/libSDK --funcCodePath \$DAEDALUS_RT_DIR/examples/func_code

Figure 4.3: Running ESPAM to synthesize the platform with multiple applications. $\$ is used to break the command over multiple lines

Now, proceed to Section 4.1.3 for instructions on how to build and run the generated project.

4.1.3 Building and Running the Generated Project

To build and run the generated project, perform the following steps:

Follow the steps from Section 3.1.3 for building the hardware part of the generated project using Xilinx tools. For the software part, perform step 1 by setting the workspace to:
 \$DAEDALUS_RT_DIR/examples/synthetic/output/xps_project/SDK. After performing step 1 from Section 3.1.3, return here and perform the following:

(a) Download FreeRTOS Board Support Package (BSP) for MicroBlaze. This can be downloaded from the following URL:

http://daedalus.liacs.nl/daedalus-rt/release_20120907/FreeRTOS-7.1.0.zip

Unzip this file and you will get a folder named FreeRTOS_Xilinx_SDK_BSP, which contains two subdirectories called bsp and sw_apps. Move the FreeRTOS_Xilinx_SDK_BSP directory to a directory where it will not be deleted or modified. The location of FreeRTOS_Xilinx_SDK_BSP will be denoted \$FREERTOS_DIR

(b) Go back to SDK, and click on "Xilinx Tools" > "Repositories" as shown in Figure 4.4

File Edit Source Refactor Navigate Search Run Project XilinxTools Window Help ID+ ID	×
Poroject Explorer X C by system.xml X Repositories C C An outline is no An outline is no An outline is no C C C C C C C C C C C C C C C C C C	
E 🖏 🎽 sobel_hw_platform Program FPGA An outline is no	Ma 🗖 🗖
	available.
BSP bot: If BSP_3 Shot: If	
Q4	

Figure 4.4: Importing FreeRTOS BSP into Xilinx SDK

(c) Now, you will see a window like the one shown in Figure 4.5. Add a new Global Repository by clicking on "New" next to "Global Repositories". Now, browse to \$FREERTOS_DIR/FreeRTOS_Xilinx_SDK_BSP/ and click "OK". Then, click on "Apply" and "OK" in the window shown in Figure 4.5

Preferences		-			
type filter text 🛛 🕅	Add, remove or change the order of SDK's software repositories.	\$7.\$7.			
▶ General	Local Repositories (available to the current workspace)				
 ► C/C++ ► Help 		New			
Install/Update		Remove			
 Remote Systems Run/Debug 		Up			
 Run/Debug Team 		Down			
Terminal	Global Repositories (available across workspaces)	Down			
 Xilinx SDK Flash Programming 	/home/mohamed/tools/FreeRTOS_Xillinx_SDK_BSP				
Hardware Specificati	6	New			
Log Information Leve	k3*	Remove			
Repositories Target Manager		Up			
larget Manager		Down			
	SDK Installation Repositories				
	Rescan Repositories				
	Note: Local repository settings take precedence over global repository settings.				
	Hote. Local repository sectings take precedence over global repository sectings.				
	Restore Defaults Apply				
?	Cancel	ок			

Figure 4.5: Specifying the BSP location

- 2. Now, continue with steps 2–6 from Section 3.1.3. After Step 6, return here.
- 3. After importing the projects successfully, we need to modify the maximum number of priorities supported by FreeRTOS. The default value is 4, however, in this example, we map 8 tasks onto a single processor. Right-click on BSP_P_1 and then click on "Board Support Package Settings" as shown in Figure 4.6.

Project Explorer 😫	= 🗆 🕞 system.xml 😫	• •	🗄 Ou 🕱 💿 Ma
= 4		rdware Platform Specification	
BSP_host_if			An outline is not availa
BSP_P_1	Design Information		
\$host_if	New		
\$P.1			
xps project hw platform	Open in <u>N</u> ew Window	2012	
http://www.anglecci.nethornet.org			
	Delete Dele	be 0 oxedolifi 0 oxedolifi	
		10 0×0003ffff	
	Rena <u>m</u> e	F2 10 0xf200ffff	
	Import	00 0xf100ffff	
	Export	00 Usb100ffff	
	<u>B</u> uild Project	0 0xaffffff	
	Clean Project	00 0x8001ffff	
		0.0xbffffff	
		F5 D0 0x8ffffff	
	Clo <u>s</u> e Project		
		11120000x0 000000	
	Build Configurations	000000 0x000000f	
	Make Targets	000000 0x0003fff	
	Index	111100000 (x400010111	
	Show in Remote Systems view	000000 0x6000ffff	
	Run As	.000000 0xb100ffff	
	Debug As	000000 0xaffffff	
	Profile As	000000 0xbrififf	
	Team		
	Comp <u>a</u> re With		
	Restore from Local History		
		🛛 Properties 🖉 Terminal 1 🕴 🕹 😧 🐻 💷	🔠 📭 🖻 🖬 🖏 🖓 🖓
	Board Support Package Settings		
	Properties Alt+En	ter	
	Compiling tmrctr	-	
	Compiling cpu		
	Running execs_generate. Finished building libraries		

Figure 4.6: Opening the Board Support Package Settings window

 Now, click on freertos > kernel_behavior > max_priorities, then change it to 10 as shown in Figure 4.7.

Overview					
freertos	Configuration for OS: fr	eertos			
drivers	Name	Value	Default	Туре	Description
cpu	stdin	host_if_mb_RS232_U	none	peripheral	Specify the instance nan
	stdout	host_if_mb_RS232_U	none	peripheral	Specify the instance nan
	systmr_interval	1000	10	integer	Specify the time interva
	kernel_behavior	true	true	boolean	Parameters relating to t
	use_preemption	true	true	boolean	Set to true to use the pr
	idle_yield	true	true	boolean	Set to true if the Idle tas
	max_priorities	10	4	integer	The number of task prio
	minimal_stack_size	120	120	integer	The size of the stack allo
	total_heap_size	65536	65536	integer	Only used if heap_1.c or
	max_task_name_len	8	8	integer	The maximum number o
	kernel_features	true	true	boolean	Include or exclude kerne
	hook_functions	true	true	boolean	Include or exclude appli
	software timers	true	true	boolean	Options relating to the

Figure 4.7: Modifying the maximum number of priorities in FreeRTOS

- 5. Now, proceed with steps 7–11 from Section 3.1.3.
- 6. After building the software, perform steps 2, 3 and 5 from Section 3.1.4 to program the FPGA.
- 7. Now, open a UART terminal and you should observe periodic messages from the applications together with the output computed by them. This is illustrated in Figure 4.8. The period of the message from pipeline is around 500 ms, while the messages from split_join have a period equal to around 1000 ms.

😣 🗐 🗉 CuteCom			
Open device	Device:	/dev/ttyUSB0 🔻	Parity: None 🛟
Cl <u>o</u> se device	Baud rate:	115200	Handshake: Software Hardware
About	Data bits:	8	Open for: 🗹 Reading 🕞 Writing
Quit	Stop bits:	1	🖉 Apply settings when opening
Pipeline produced a toke Pipeline produced a toke Split-join produced toke Pipeline produced a toke Pipeline produced toke Pipeline produced a toke Split-join produced a toke Pipeline produced toke Pipeline produced toke Pipeline produced toke	n (12) ns (10, 10) n (14) n (16) ns (30, 30) n (18) ns (50, 50) ns (50, 50) n (22) ns (20) ns (70, 70) n (26) n (28) ns (30, 90) (20)	nome/mohamed/cutecom.log	
2			
1			
H a5a5a5a5			
12345678			
Input:			
Send file Plain 🛟			LF line end 🔅 Char delay: 1 ms 🗘

Figure 4.8: The output from pipeline and $\texttt{split_join}$ running on FPGA

Contributing to Daedalus/Daedalus^{RT}

All the individual parts of Daedalus and Daedalus^{RT} are available as open-source software. However, the development of these tools and the access to their code repositories is limited. We provide here a quick overview on how you can contribute to each tool.

5.1 PNgen

The PNgen compiler is maintained by Sven Verdoolaege at the following git repository:

http://repo.or.cz/w/isa.git

Therefore, anyone with Internet access can follow the latest developments in PNgen. For bug reports, please contact the maintainer.

5.1.1 Building PNgen from the git repository

To build PNgen from the git repository, you need to clone the repository and follow the instructions in the INSTALL file accompanying the cloned repository. Installing PNgen requires installing several packages and libraries. The instructions in the INSTALL file show you how to build each library from its sources. However, on modern Linux distributions, these libraries can be installed using the pre-built packages provided by the distribution vendor. For example, on 64-bit Ubuntu machines, you can install PNgen by performing the commands shown in Figure 5.1

```
mkdir pngen
cd pngen
sudo apt-get install pkg-config libtool bison flex libgmp3-dev libyaml-dev libntl-dev libxml2-dev
wget -q http://nl.archive.ubuntu.com/ubuntu/pool/universe/s/syck/libsyck0-dev_0.55+svn270-1_amd64.deb
sudo dpkg -i libsyck0-dev_0.55+svn270-1_amd64.deb
wget -q http://daedalus.liacs.nl/pngen/release_20120706/clang+llvm-2.9-x86_64-linux.tar.bz2
tar -xvjf clang+llvm-2.9-x86_64-linux.tar.bz2
CWD='pwd'
git clone git://repo.or.cz/isa.git
cd isa
./get_submodule.sh
./autogen.sh
./configure --prefix=$CWD/pngen/ --with-clang-prefix=$CWD/pngen/clang+llvm-2.9-x86_64-linux.tar
make
make check
make install
```

Figure 5.1: Steps to install PNgen on 64-bit Ubuntu without building the libraries required by PNgen

5.2 pntools

pntools is maintained by Sven van Haastregt and Jiali Teddy Zhai at the following git repository: csartem@ssh.liacs.nl:pntools.git

The access to this repository is restricted to selected users only. For bug reports or any other inquiries, please contact one of the maintainers.

5.2.1 Building pntools from the git repository

To build pntools from the git repository, you need to clone the repository and follow the instructions in the README.md file accompanying the cloned repository.

5.3 darts

darts is maintained by Mohamed Bamakhrama at the following git repository:

csartem@ssh.liacs.nl:darts.git

The access to this repository is restricted to selected users only. For bug reports or any other inquiries, please contact the maintainer.

darts is developed in Python, and therefore, does not require any steps for building it. The user can directly use the tools.

5.4 ESPAM

ESPAM is maintained by Todor Stefanov at the following git repository:

csartem@ssh.liacs.nl:espam.git

The access to this repository is restricted to selected users only. For bug reports or any other inquiries, please contact the maintainer.

5.4.1 Building ESPAM from the git repository

To build ESPAM from the git repository, you need to clone the repository and follow the instructions in the README.md file accompanying the cloned repository.

Bibliography

- [1] M. Thompson, H. Nikolov, T. Stefanov, A. D. Pimentel, C. Erbas, S. Polstra, and E. F. Deprettere, "A framework for rapid system-level exploration, synthesis, and programming of multimedia MP-SoCs," in *Proceedings of the 5th IEEE/ACM international conference on Hardware/software codesign and system synthesis*, CODES+ISSS '07, (New York, NY, USA), pp. 9–14, ACM, 2007.
- [2] H. Nikolov, M. Thompson, T. Stefanov, A. Pimentel, S. Polstra, R. Bose, C. Zissulescu, and E. Deprettere, "Daedalus: toward composable multimedia MP-SoC design," in *Proceedings of the 45th annual Design Automation Conference*, DAC '08, (New York, NY, USA), pp. 574–579, ACM, 2008.
- [3] M. Bamakhrama, J. Zhai, H. Nikolov, and T. Stefanov, "A methodology for automated design of hard-real-time embedded streaming systems," in *Proceedings of the 15th Design, Automation, and Test in Europe Conference* and Exhibition, DATE 2012, pp. 941–946, March 2012.
- [4] G. Bilsen, M. Engels, R. Lauwereins, and J. Peperstraete, "Cyclo-static data flow," *IEEE Trans. Signal Process.*, vol. 44, pp. 397–408, 1996.
- [5] S. Verdoolaege, H. Nikolov, and T. Stefanov, "pn: a tool for improved derivation of process networks," *EURASIP Journal on Embedded Systems*, vol. 2007, January 2007.
- [6] A. Pimentel, C. Erbas, and S. Polstra, "A systematic approach to exploring embedded system architectures at multiple abstraction levels," *IEEE Transactions on Computers*, vol. 55, pp. 99–112, February 2006.
- [7] H. Nikolov, T. Stefanov, and E. Deprettere, "Systematic and Automated Multiprocessor System Design, Programming, and Implementation," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 27, pp. 542–555, March 2008.
- [8] S. Stuijk, M. Geilen, and T. Basten, "SDF³: SDF For Free," in *Proceedings of the 6th International Conference on Application of Concurrency to System Design*, ACSD 2006, (Los Alamitos, CA, USA), pp. 276–278, IEEE Computer Society Press, June 2006.