Datamover Design Example

Datamover example design sets to demonstrate design practices and software solutions to achieve high performance real time application with HPS ARM processor.

Category:	Demo
Board:	Altera Cyclone V SoC Board
Tools Version:	14.0
State:	running
Miembros:	TienHockLoh

Table of Contents

> Deliverables
>
> - FPGA, Linux, and Nios II Binaries
> - VxWorks and HWLibs Binaries
> - Hardware Design
> - Linux Patch
> - Linux Yocto Recipes
> - VxWorks
> - HWLibs
>
> Reference

Overview

The Datamover example design uses a Cyclone V SoC development kit to demonstrate data communication between the FPGA logic and SDRAM controlled through the Hard Processor System (HPS) portion of the device. The design uses a Nios II processor along with an mSGDMA in the FPGA fabric to move data through the F2H bridge. The Nios II is also used to track the time for each DMA transfer as well as the loop time, which is defined as the total duration for a packet to enter the SoC, be processed, and then transferred into on-chip memory in the FPGA fabric. The jitter of each measurement, ingress DMA time, and egress DMA time are also calculated.

The software for this design is discussed with respect to implementation using Linux with core affinity, VxWorks with core affinity, and a baremetal solution using hardware libs. All three software implementations yield different results and statistics that can assist a user in deciding the best software solution to meet the real-time requirements of a design.

Prerequisites

The following hardware and software components are required to implement this design:

Hardware

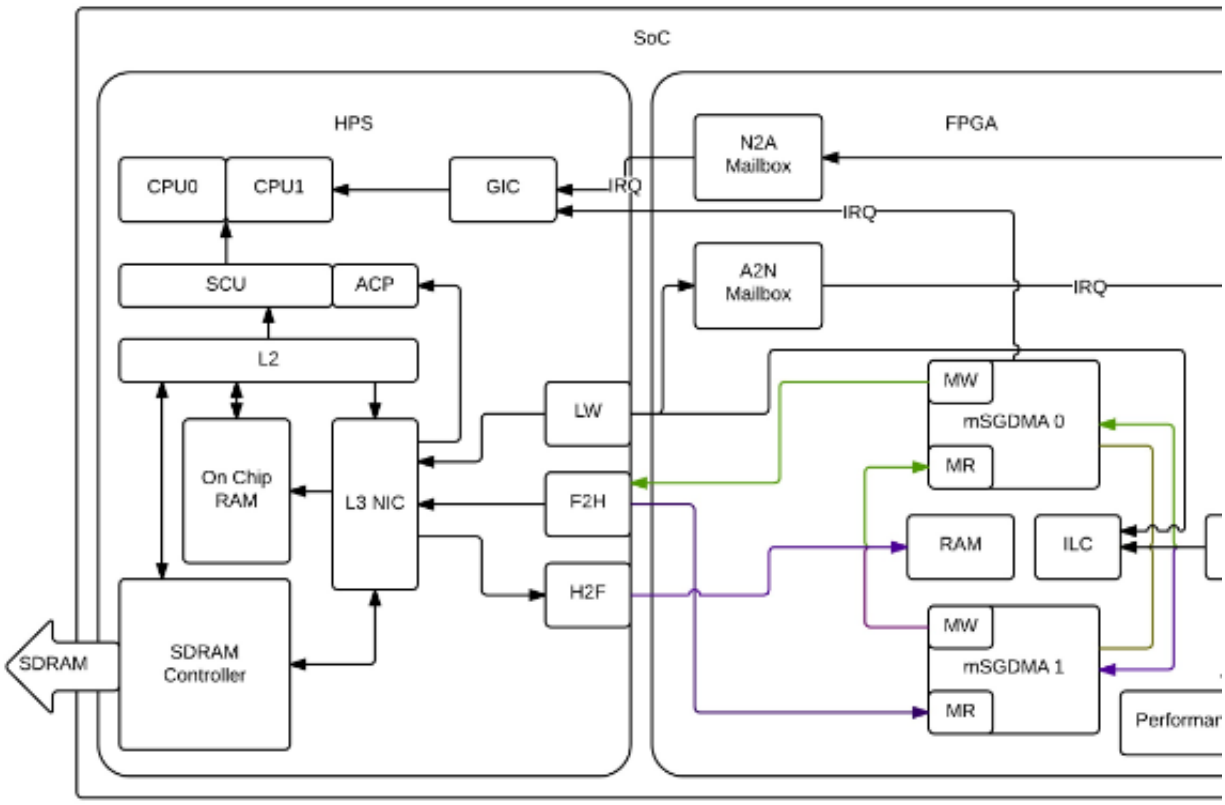
- Cyclone V SoC Development Kit
- 4GB Micro SDHC flash card

Software

- Quartus II Version 14.0
- Wind River Workbench version : 3.3.4
- ARM DS-5 Development Studio
- Win32DiskImager

Hardware Implementation

The following diagram shows the hardware implementation of the design in the FPGA and the required connections between the FPGA and the HPS.



The Datamover design system requires the following components:

- FPGA
  - Nios II
  - Two Modular Scatter Gather Direct Memory Access (mSGDMA)
  - Performance Counter
  - Two Mailboxes
  - Payload RAM
- HPS (ARM Cortex-A9)

Nios II
A Nios II processor is placed in the FPGA as a traffic controller to perform the necessary data transfer between FPGA and HPS. The Nios II processor also serves as a time tracker to monitor the time taken to perform each data transfer.

Modular Scatter Gather Direct Memory Access (mSGDMA)

The mSGDMA is a soft logic component that is used to perform the necessary data transfer between the FPGA and the HPS via the FPGA to HPS (H2F) bridge. Two mSGDMA components are placed in the FPGA to perform the data from the FPGA to the HPS and the HPS to the FPGA.

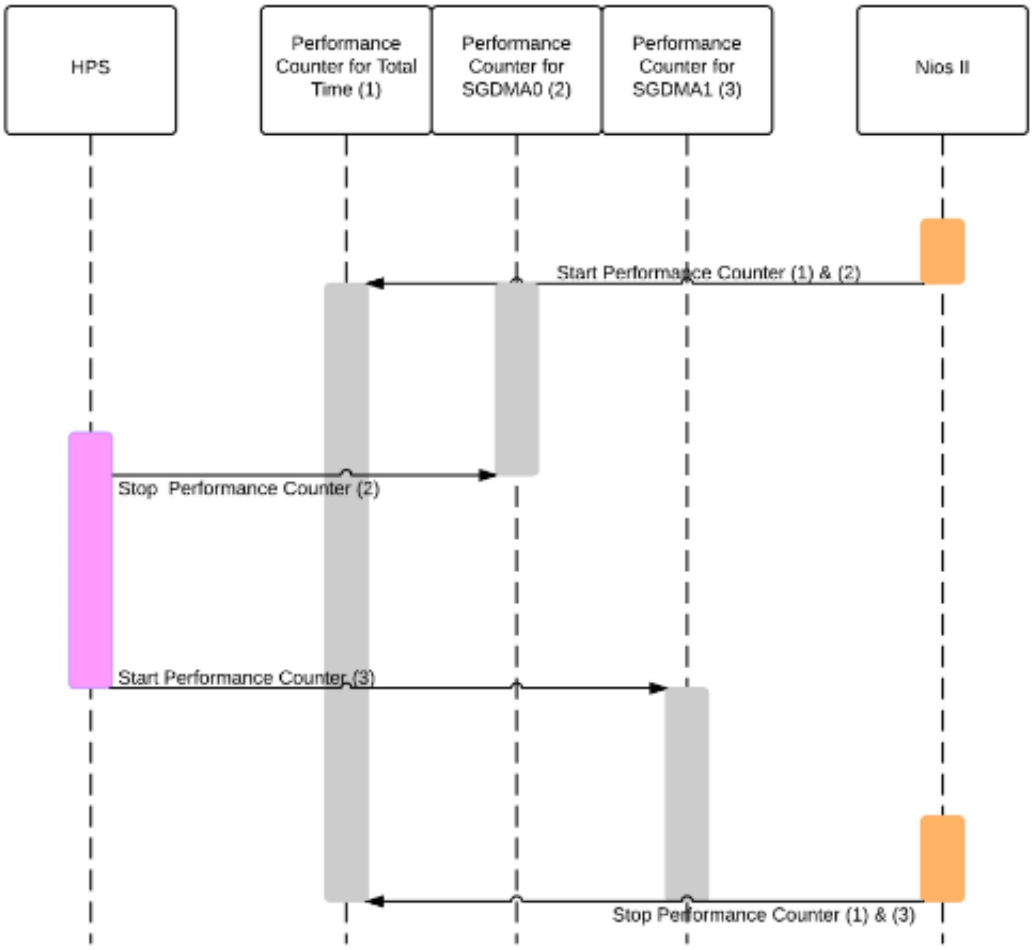
Performance Counter
The performance counter soft logic in the FPGA serves as a performance timer. It is configured by the Nios II to perform the performance monitoring of each data transfer. Each performance counter consists of one global timer and three performance timers to monitor the time taken for ingress, egress and total loop time of the data transfer.

Mailboxes
The mailboxes act as a command passing mechanism between the FPGA and the HPS and allow synchronization of the software between the Nios II and the HPS.

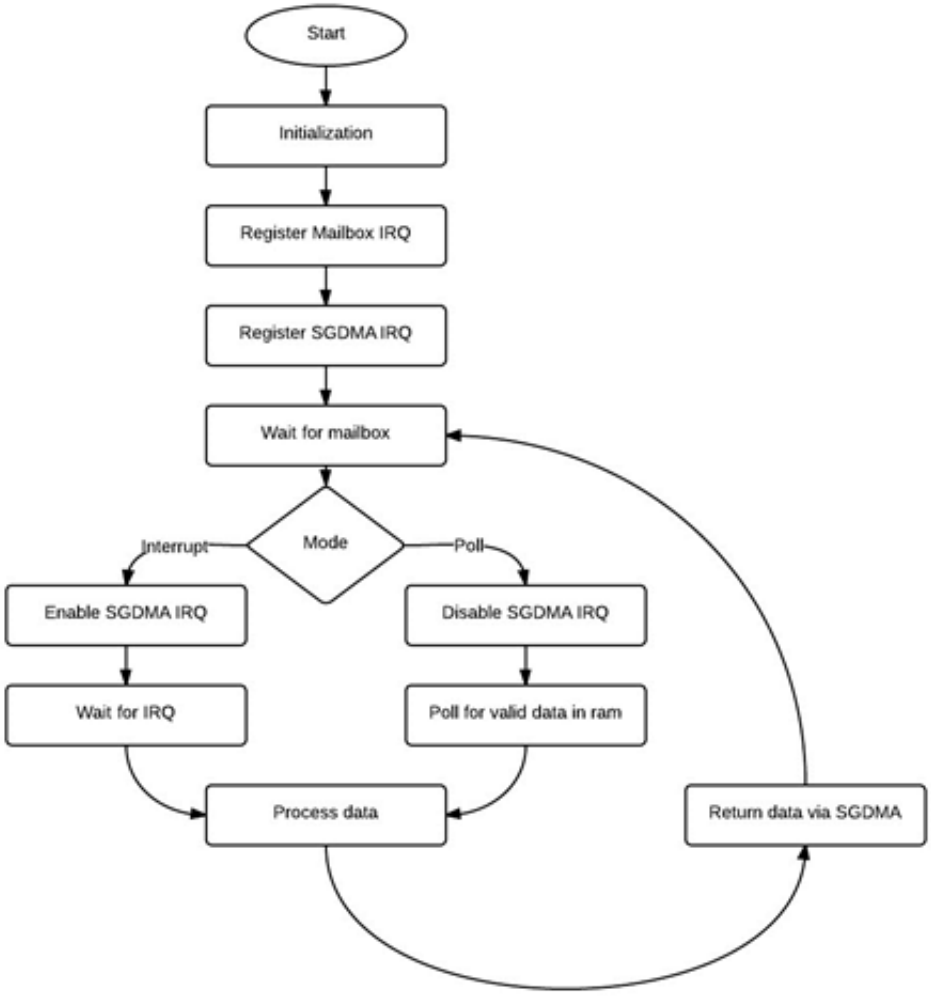
Payload RAM
The payload RAM is an on-chip RAM in the FPGA used as a temporary memory to store the data to be transferred between the FPGA and the HPS.

Software Flow

1. The Nios II processor configures mSGDMA0 to move data through the HPS SDRAM controller using the F2H AXI bridge.
2. Either mSGDMA0 or mSGDMA1 is configured to move data from the HPS on-chip RAM into the FPGA on-chip memory.
3. The Nios II processor is configured to track the time for each DMA transfer to the HPS and for returning the DMA data to the FPGA. Three time stamps are used to track the complete loop of the data transfer:
    - Total looptime (FPGA -> HPS -> FPGA)
    - DMA transfer time (FPGA -> HPS)
    - DMA transfer time (HPS -> FPGA)
4. Nios II starts the performance counters associated with the total looptime and the FPGA -> HPS DMA transfer time. Once the DMA completes the transfer, the HPS stops the counter associated with the FPGA -> HPS DMA transfer and starts the performance counter for the HPS -> FPGA DMA transfer before initiating the DMA transfer. Once the total loop is completed, Nios II stops the counter for the HPS -> FPGA DMA transfer (HPS -> FPGA) and for the total loop. The figure below illustrates these times using the vertical gray bars.



The software flow is shown in the following figure:



There are 3 different platforms employed for the same software flow:

- Linux SMP
- VxWorks SMP
- Baremetal running on HPS.

The performance results of the three platforms are benchmarked against each other and are shown in the NiosII terminal.

Getting Started

Obtain FPGA, Linux and NiosII Binaries

Download cv_datamover_ed.tar.gz from http://releases.rocketboards.org/release/2014.09/cv-datamover-ed and save it to the user home folder.

Unzip the example design archive:

```
$ cd ~
$ tar xzf cv_datamover_ed.tar.gz
```

The following files are used to run the Reference Design:

Folder	Folder	Folder	File	Description
cv_datamover_ed	bin	Linux	sdimage.bin	SD Card Image
		FPGA	Datamover_5csxfc6.flash	FPGA Configuration Flash
		Nios	Datamover_5csxfc6.flash	Nios Configuration Flash

Obtain VxWorks Binaries

Download the VxWorks binaries from https://www.altera.com/support/support-resources/download/rtos_tools.html and save to the user home folder. Unzip the example design archive:

```
$ cd ~
$ unzip Altera_datamover_ed.zip
```

The following files are used to run the Reference Design:

Folder	Folder	Folder	File	Description
Altera_datamover_ed	bin	vxworks	vxw_datamover_sdmmc.img	Vxwork image

Board Setup

This section presents the board settings required for running the Datamover design.

Jumpers:

Jumper	Setting
J5	open
J6	shorted
J7	shorted
J9	open
J13	shorted
J16	open
J26	right shorted

J27	right shorted
J28	right shorted
J29	right shorted
J30	left shorted
J31	open

Switches:

Switch	Setting
SW1	All OFF
SW2	All OFF
SW3	1:ON 2:OFF 3:ON 4:ON 5:OFF 6:ON
SW4	1:OFF 2:OFF 3:ON 4:ON

Configuring Board to Use EPCQ

By default the board is configured to use the onboard FPGA configuration device as EPCS. In order to configure the board to use the configuration device as EPCQ, follow the instructions posted at http://www.altera.com/support/kdb/solutions/rd11192013_118.html. Note that these steps are only required to be performed one time. Once the board has been configured to use EPCQ, this configuration is stored in flash memory.

For a step-by-step tutorial on how to configure board to use EPCQ, refer to click here.

Programming the Board

Open the Nios II Command shell and write the FPGA Flash Image into EPCQ:

```
$ cd ~/cv_datamover_ed
$ nios2-configure-sof bin/fpga/datamover_5csxfc6.sof –d 2
$ nios2-flash-programmer --base=0x10040000 --epcs bin/fpga/datamover_5cs
xfc6.flash
```

Write the Nios II flash image into EPCQ:

```
$ nios2-flash-programmer --base=0x10040000 --epcs bin/nios/Datamover_5cs
xfc6.flash
```

After the FPGA and Nios flash image have been written successfully into EPCQ, the FPGA flash and Nios II flash automatically boot from EPCQ after the device is powered up:

```
$ nios-terminal
```

```
nios2-terminal: connected to hardware target using JTAG UART on cable
nios2-terminal: "USB-BlasterII on pg-csng [USB-1]", device 2, instance 0
nios2-terminal: (Use the IDE stop button or Ctrl-C to terminate)

Datamover: Starting poll mode
Data Size       Ingress(jitter),us      Regress(jitter),us      Total(ji
```

Getting Start Guide – Linux

After following the board set up and programming steps above, prepare the SD Card image by executing the following commands:

```
$ cd ~/cv_datamover_ed
$ sudo dd if=bin/linux/sdimage.bin of=/dev/sdx
* Please replace ‘dev/sdx’ with the name of the SD card device on your h
ost computer.
* Please ensure that the name of SD card device on your host computer is
correct. You will not get the SD card image if you write into wrong driv
e
```

```
$ sudo sync
```

Insert the SD card into the CV SoC dev kit and power up the board. After the booting process is finished, login as root at the kernel terminal and follow the steps below to run the Datamover application:

```
$ modprobe datamover
```

The following is example output from Nios II terminal:

```
Datamover: Starting poll mode
Data Size       Ingress(jitter),us      Regress(jitter),us      Total(ji
32               0.145(4.962)            0.301(0.015)            0.310(4
64               0.173(0.165)            0.343(0.188)            0.380(0
128              0.213(0.090)            0.431(0.030)            0.507(0
256              0.386(0.075)            0.602(0.045)            0.851(0
512              0.799(0.068)            1.006(0.068)            1.669(0
1024             1.631(4.925)            1.816(0.075)            3.312(4
2048             2.797(0.075)            2.953(0.105)            5.616(0

Datamover: Starting interrupt mode
Data Size       Ingress(jitter),us      Regress(jitter),us      Total(ji
32               1.585(3.962)            0.301(0.023)            1.767(3
64               1.588(2.917)            0.343(0.030)            1.812(2
128              1.648(2.820)            0.431(0.180)            1.960(2
256              1.821(3.090)            0.601(0.173)            2.305(3
512              2.232(4.135)            1.006(0.188)            3.120(4
1024             3.056(5.135)            1.816(0.075)            4.755(5
2048             4.228(3.068)            2.951(0.241)            7.063(3

Datamover: Completed
```

The following is example output from the UART terminal when the Datamover is executed in the HPS:

```
Datamover CMD: Mode: poll, Data Size: 32
root@socfpga:~# Datamover CMD: Mode: poll, Data Size: 64
Datamover CMD: Mode: poll, Data Size: 128
Datamover CMD: Mode: poll, Data Size: 256
Datamover CMD: Mode: poll, Data Size: 512
Datamover CMD: Mode: poll, Data Size: 1024
Datamover CMD: Mode: poll, Data Size: 2048
Datamover CMD: Mode: interrupt, Data Size: 32
Datamover CMD: Mode: interrupt, Data Size: 64
Datamover CMD: Mode: interrupt, Data Size: 128
Datamover CMD: Mode: interrupt, Data Size: 256
Datamover CMD: Mode: interrupt, Data Size: 512
Datamover CMD: Mode: interrupt, Data Size: 1024
Datamover CMD: Mode: interrupt, Data Size: 2048
```

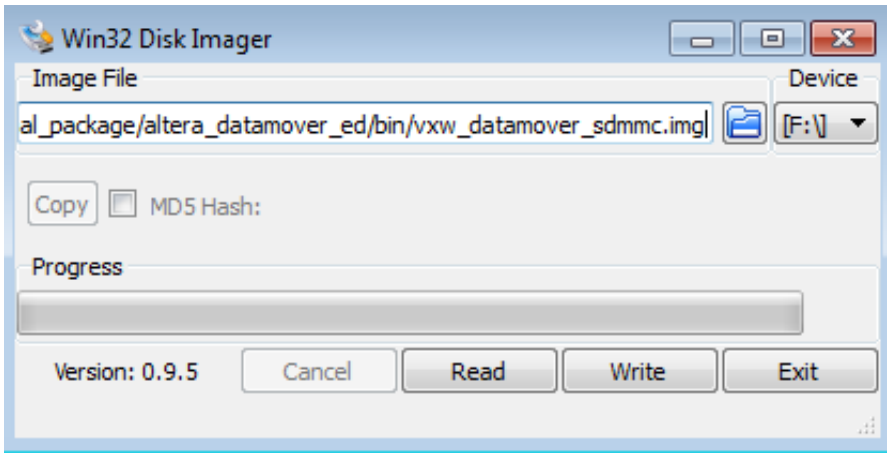
Continue testing the program by pressing the CPU_RST button on the board to obtain stable and consistent results.

Getting Start Guide –VxWorks

After following the board set up and programming steps above, copy the SD card image from /bin/vxworks/vxw_datamover_sdmmc.img file into your local directory.
Use Win32DiskImager to write the SD image to the flash card:

- For the Image File, browse to the vxw_datamover_sdmmc.img directory.

- For the Device, browse to the SD card driver.
- Click Write to write the image to the SD card.



Insert the SD card into the development kit after it has been successfully written. Power up the board.

Perform the configuration for the VxWorks environment by following the steps below to modify the boot parameters:

- Boot the board and hit a key to stop the boot count-down process.
- Use the "c" command to check and modify each of the boot parameters.
- Enter the correct value and then hit <enter> to move to the next boot parameter.
- Do not edit the existing values. You will need to enter the whole string.

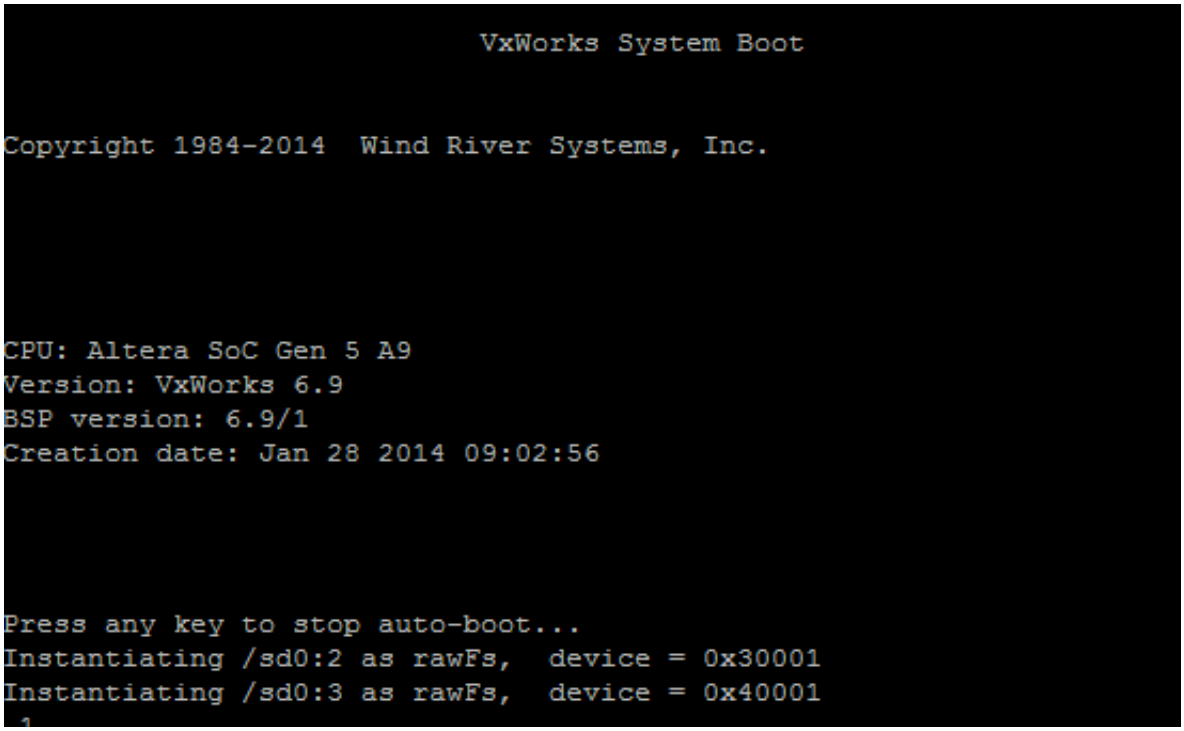
```
[VxWorks Boot]: c

boot device: fs
processor number: 0
host name: host
file name: /sd0:1/vxWorks
inet on ethernet (e) : 192.168.192.204:ffffff00
inet on backplane (b):
host inet (h): 192.168.192.1
gateway inet (g):
user (u): target
ftp password (pw): vxTarget
flags (f): 0x0
target name (tn): alt_soc_gen5
startup script (s):
other (o): emac1

[VxWorks Boot]:

To delete the field use '.'
To go back to the previous parameter use '-'
To go to the next parameters use <enter>
To backspace use <ctrl>H
```

After finishing the configuration, boot the VxWorks image by pressing the "@" key or power up the development kit and wait for the count down. After VxWorks is booted, you should see the VxWorks splash screen as below:



```
auto-booting...


boot device          : fs
unit number          : 0
processor number     : 0
host name            : host
file name            : /sd0:1/vxWorks
inet on ethernet (e) : 192.168.192.204:ffffff00
host inet (h)        : 192.168.192.1
user (u)             : target
ftp password (pw)    : vxTarget
flags (f)            : 0x0
target name (tn)     : alt_soc_gen5
other (o)            : emac1

Loading /sd0:1/vxWorks...1895020 + 342132
Starting at 0x100000...


Adding 7640 symbols for standalone.


                VxWorks

Copyright 1984-2014  Wind River Systems, Inc.

            CPU: Altera SoC Gen 5 A9x2 MPCore
    Runtime Name: VxWorks
 Runtime Version: 6.9 SMP
     BSP version: 6.9/1
         Created: Aug 14 2014 13:58:23
ED&R Policy Mode: Deployed
   WDB Comm Type: WDB_COMM_END
            WDB: Ready.


->
```

Type “datamover_start” and you should see the screen below:

```
-> datamover_start
interrupt: Datamover CMD: Mode: poll, Data Size: 32
0x427f298 (tShell0): Datamover: Sucessfully loaded
value = 0 = 0x0
```

Example output from the Nios II terminal:

```
Datamover: Starting poll mode
Data Size       Ingress(jitter),us      Regress(jitter),us      Total(j
32               0.577(208.835)          0.204(0.008)            0.827(
64               0.400(0.060)            0.243(0.015)            0.689(
128              0.445(0.060)            0.321(0.030)            0.811(
256              0.632(0.053)            0.476(0.060)            1.154(
512              1.047(0.008)            0.846(0.120)            1.939(
1024             1.873(0.008)            1.587(0.241)            3.508(
2048             3.044(0.008)            2.590(0.481)            5.682(

Datamover: Starting interrupt mode
Data Size       Ingress(jitter),us      Regress(jitter),us      Total(j
32               0.878(2.917)            0.204(0.008)            1.143(
64               0.908(2.271)            0.243(0.015)            1.212(
128              0.968(2.444)            0.321(0.030)            1.350(
256              1.135(2.083)            0.477(0.060)            1.673(
512              1.547(2.842)            0.848(0.120)            2.456(
1024             2.370(1.226)            1.588(0.241)            4.020(
2048             3.541(1.180)            2.591(0.481)            6.195(

Datamover: Completed
```

Example output from the UART terminal when VxWorks is executed in HPS

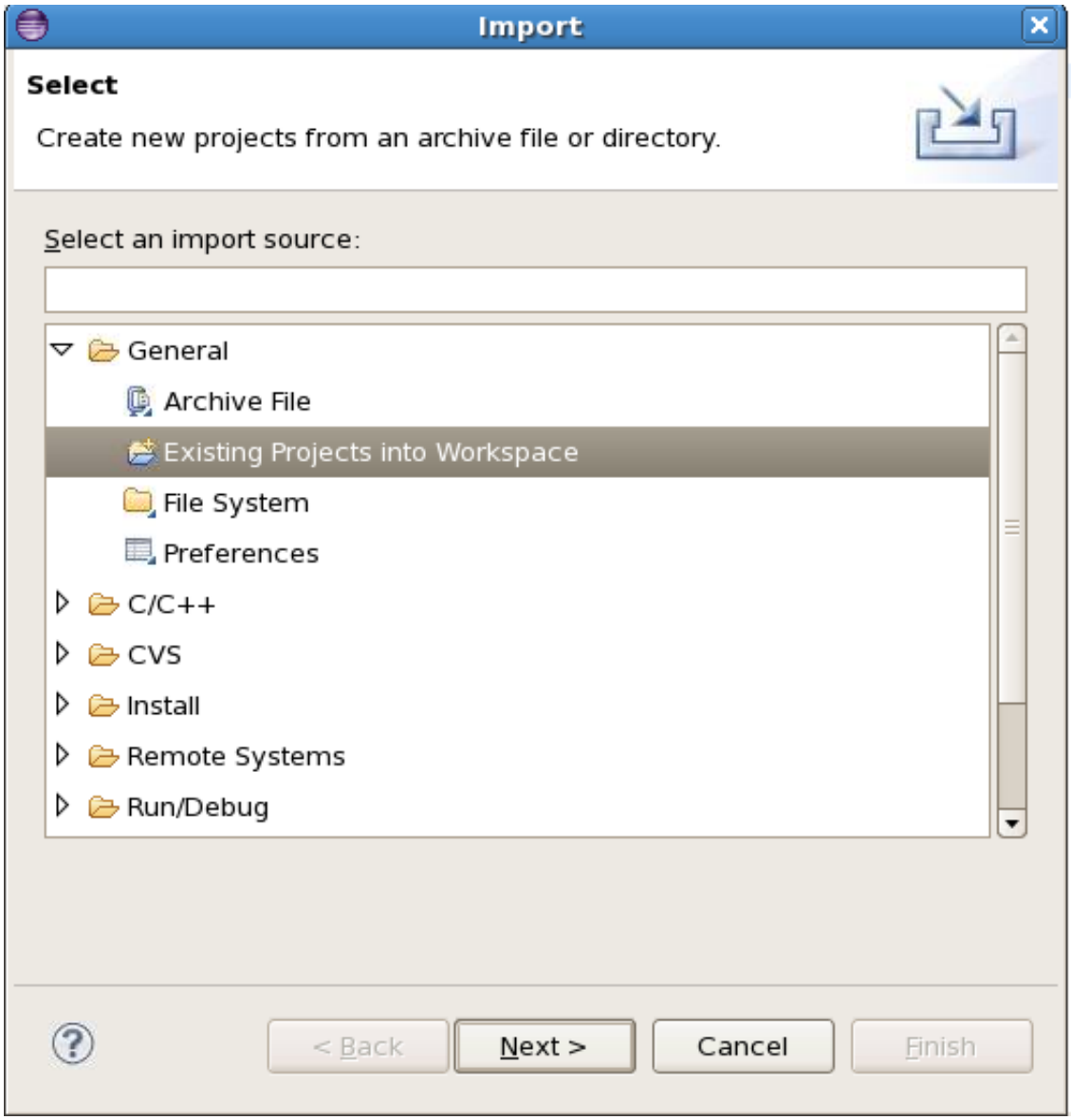
```
-> datamover_start
interrupt: Datamover CMD: Mode: poll, Data Size: 32
0x427f298 (tShell0): Datamover: Sucessfully loaded
value = 0 = 0x0
-> interrupt: Datamover CMD: Mode: poll, Data Size: 64
interrupt: Datamover CMD: Mode: poll, Data Size: 128
interrupt: Datamover CMD: Mode: poll, Data Size: 256
interrupt: Datamover CMD: Mode: poll, Data Size: 512
interrupt: Datamover CMD: Mode: poll, Data Size: 1024
interrupt: Datamover CMD: Mode: poll, Data Size: 2048
interrupt: Datamover CMD: Mode: interrupt, Data Size: 32
interrupt: Datamover CMD: Mode: interrupt, Data Size: 64
interrupt: Datamover CMD: Mode: interrupt, Data Size: 128
interrupt: Datamover CMD: Mode: interrupt, Data Size: 256
interrupt: Datamover CMD: Mode: interrupt, Data Size: 512
interrupt: Datamover CMD: Mode: interrupt, Data Size: 1024
interrupt: Datamover CMD: Mode: interrupt, Data Size: 2048
```

Continue to test the program by pressing the CPU_RST button on the board to obtain stable and consistent results.

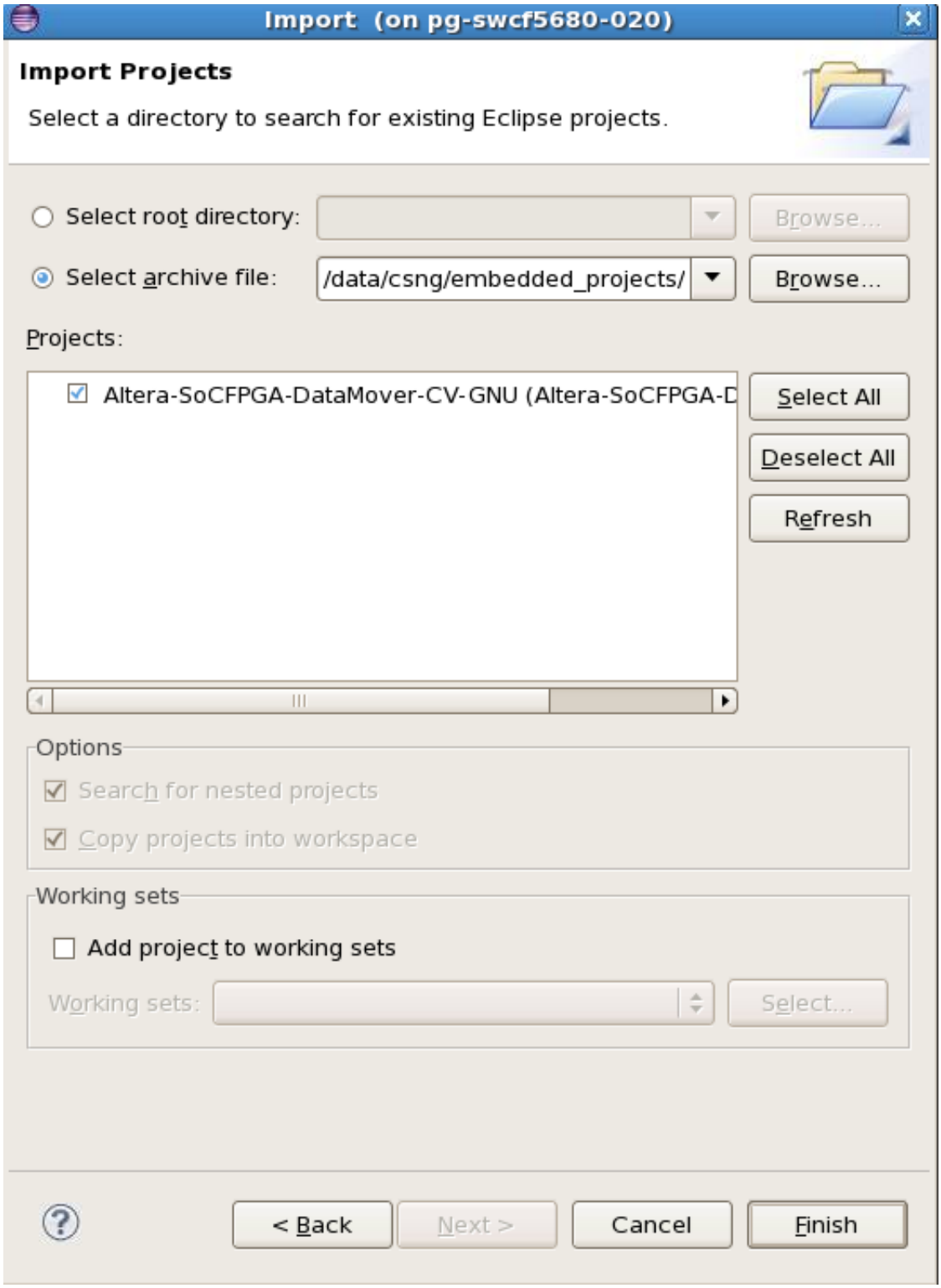
Type “datamover_busy” to exercise processor core 1. In this way, you can observe whether the performance of processor core 0 is affected when processor core 1 is busy.

After following the board set up and programming steps above, copy the Altera-SoCFPGA-DataMover-CV-GNU.tar.gz file into your local directory. Open ARM DS-5 Development Studio. Import the HWlibs archive file into ARM DS-5 Development Studio:

- File -> Import
- Select Existing Projects into Workspace
- Click Next.



In the Import Projects dialog box, browse to the HWlibs archive project directory. Select the archive file and browse to the Altera-SoCFPGA-DataMover-CV-GNU.tar.gz directory. Click Finish. You should able to see the imported project in the Projects Explorer.



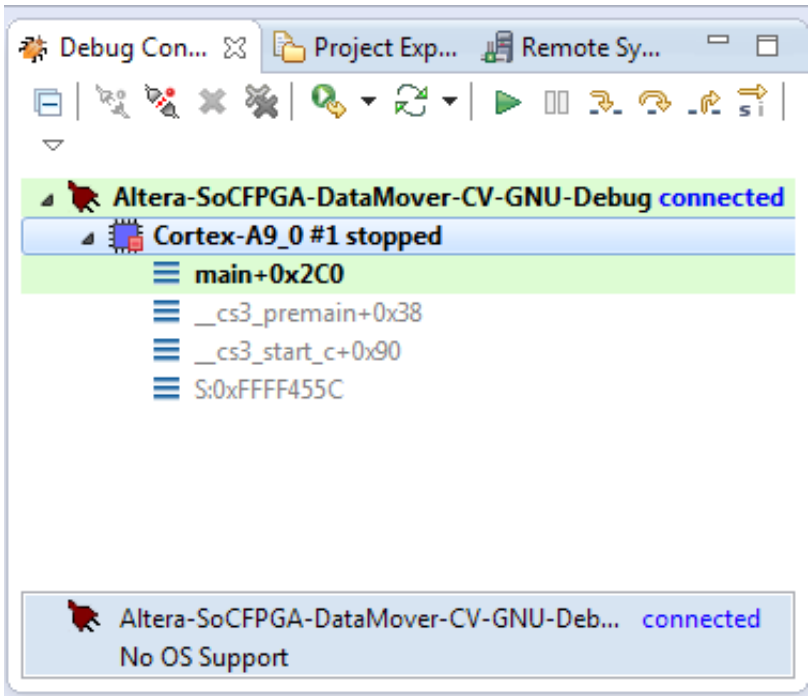
Go to Window -> Open Perspective -> Others

- Select DS-5 Debug and click Ok.

Right-click on the Altera-SoCFPGA-DataMover-CV-GNU project and select build project. After the project has finished building, right-click on the project and select Debug As -> Debug Configurations. Expand the DS-5 Debugger and click on the project name.

- In the connection tab, select Debug Cortex-A9_0
- Target Connection: USB-BlasterII
- In the Bare Metal Debug, ensure the connection is connecting correctly.

Click on Debug, after entering debug mode, you should see the debugger is connected to the device.



Click on the “Play” button and the HWLibs design is executed.

Example output from Nios II terminal :

```
Datamover: Starting poll mode
Data Size       Ingress(jitter),us      Regress(jitter),us      Total(ji
32               0.162(0.602)            0.280(0.023)            0.459(0
64               0.186(0.586)            0.322(0.023)            0.529(0
128              0.218(0.549)            0.410(0.030)            0.650(0
256              0.403(0.534)            0.580(0.045)            1.004(0
512              0.816(0.549)            0.985(0.068)            1.821(0
1024             1.643(0.541)            1.794(0.075)            3.457(0
2048             2.801(0.526)            2.931(0.120)            5.757(0

Datamover: Starting interrupt mode
Data Size       Ingress(jitter),us      Regress(jitter),us      Total(ji
32               3.863(0.541)            0.284(0.030)            4.064(0
64               3.858(0.579)            0.326(0.023)            4.101(0
128              3.916(0.534)            0.414(0.030)            4.247(0
256              4.090(0.541)            0.585(0.045)            4.592(0
512              4.474(0.556)            0.988(0.068)            5.380(0
1024             5.314(0.556)            1.798(0.098)            7.031(0
2048             6.496(0.556)            2.936(0.135)            9.352(0

Datamover: Completed
```

Continue to test the program by pressing the CPU_RST button on the board to obtain stable and consistent results.

Building the Linux example design

Software Development Flow Setting Up Yocto Environment

Follow the link to set up the Yocto environment before starting this section.

Apply Yocto Recipes and Patch

Use the commands below to acquire the poky source from Rocketboards Git:

```
$ git clone http://git.rocketboards.org/poky-socfpga.git
```

```
$ cd poky-socfpga $ git checkout -b datamover-ed ACDS14.0_REL_GSRD_PR
```

Download the patch from the release content section and apply the patch with this command:

```
$ git am cv_datamover_ed/sw/linux/altera_datamover_yocto.patch
```

Note:

Replace the file meta-alera/recipes-refdes/datamover-altera/datamover-altera_1.0.bb, replace "refdes-socfpga.git" with "linux-refdesigns.git"

Build U-Boot/Kernel/Rootfs:

```
$ source altera-init $ bitbake virtual/bootloader
```

```
$ bitbake virtual/kernel
```

```
$ bitbake altera-datamover-image
```

The output files are:

u-boot-socfpga_cyclone5.img	U-Boot image
zImage	Compressed kernel image
altera-datamover-image-socfpga_cyclone5.ext3	Root Filesystem as ext3 image
altera-datamover-image-socfpga_cyclone5.tar.gz	Root Filesystem in tar gzip archive format

Generate Device Tree Blob

Use the command below to generate the device tree blob:

```
$ sopc2dts --input datamover_5csxfc6.sopcinfo --output <dtb.file.name>.d
tb --type dtb --board soc_system_board_info.xml –board hps_common_board_
info.xml --bridge_removal all –clocks
```

Output file: <dtb.file.name>.dtb
For more details about device tree generation, refer to GSRD User Manual - Generating the Device Tree

Generate Preloader

For more details about preloader generation, refer to GSRD User Manual- Generating and Compiling the Preloader
Output file: preloader-mkpimage_cyclone5.bin.

Build SD Card Image

Replace all required components in the SD Card or build and replace the whole image:

```
$ wget http://releases.rocketboards.org/release/2014.06/gsrd/tools/make_sdimage.py
$ mkdir rootfs $ cd rootfs $ sudo tar xzf ../altera-datamover-image-socfpga_cyclone5.tar.gz
$ cd ..
$ sudo ./make_sdimage.py -f -P preloader-mkimage.bin,u-boot-socfpga_cyclone5.img,num=3,format=raw,size=10M,type=A2 -P rootfs/*,num=2,format=ext3,size=1500M -P zImage ,
socfpga.dtb,num=1,format=vfat,size=500M -s 2G -n sd_card_image_cyclone5.bin
```

For more details about SD card image generation, refer to GSRD User Manual - Creating and Updating SD Card.

Steps to import VxWorks source and Image into Wind River Workbench

To view and modify the VxWorks source code, follow the steps below to import the archived files.

VxWorks Software Prerequisites

Wind River Workbench version : 3.3.4

Create the VxWorks Source Build Project

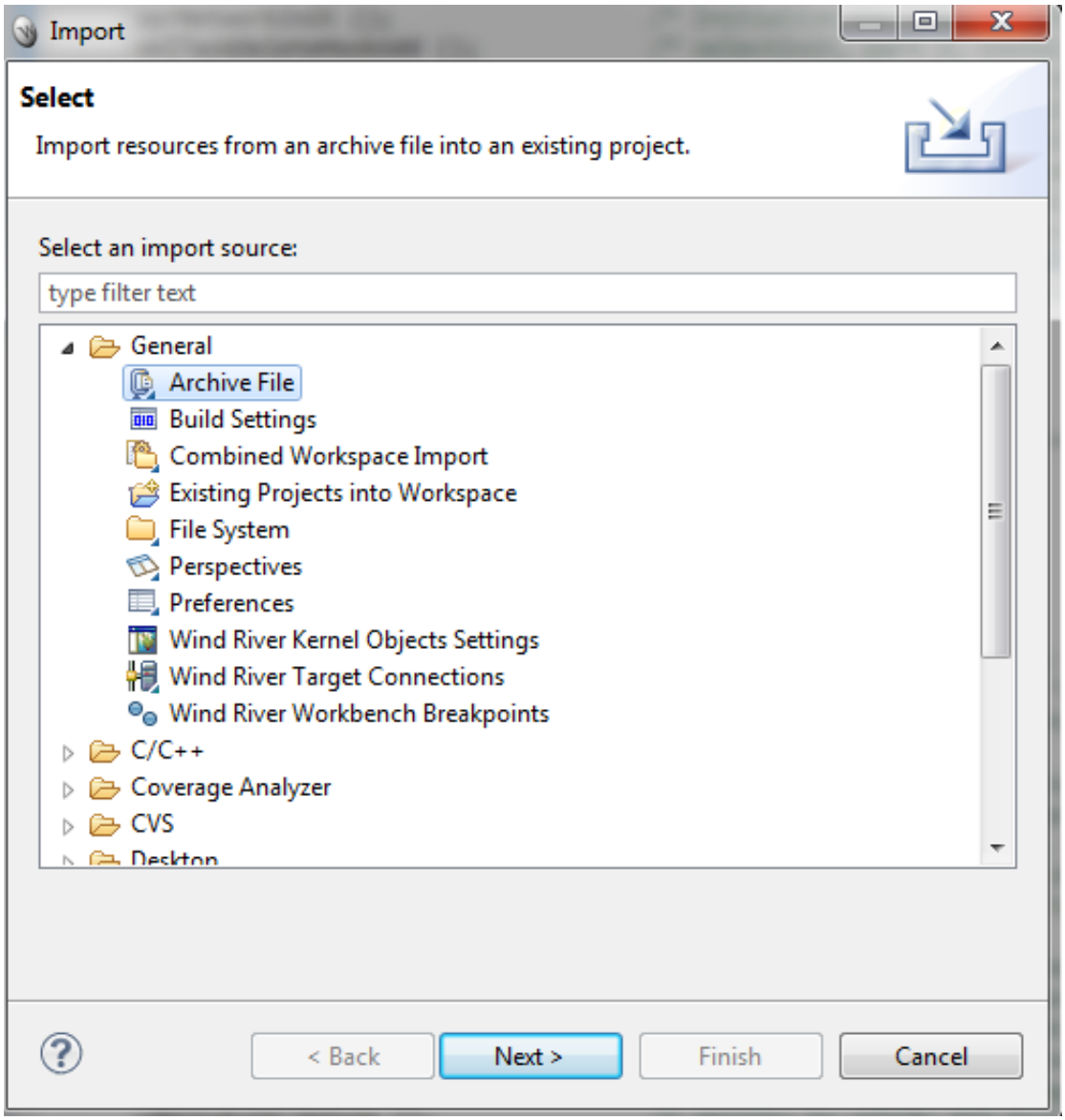
This section shows how to create a source build project (VSB) to incorporate the new BSP changes.

1. Open WindRiver Workbench.
2. Go to File -> New -> VxWorks Source Build (Kernel Library) ProjectEnter project name as VSB_SMP. Click Next.
3. Base the project on the board support package: alt_soc_gen5. Press Finish.
4. Edit VSB Options
5. Enable Symmetric Multiprocessing: SMP = y.
6. Enable processsort specific optimizations: ARMV7_CORE_CTX_A9_VSB = y.
7. Save VSB Project: File -> Save All.
8. Build Project: Project -> Build Project.

Create VxWorks Image Build Project

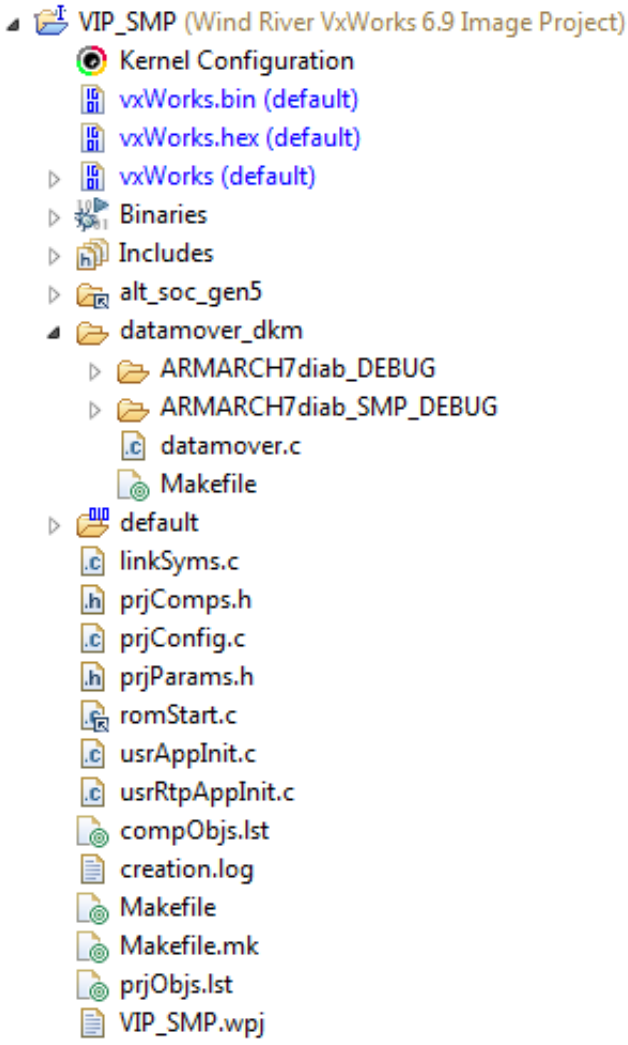
This section shows how to create an image buid project (VIP) to incorporate the new BSP changes.

1, Open WindRiver Workbench.
2. Go to File -> New ->VxWorks Image Project.
3. Enter project name as VIP_SMP. Click Next.
4. Select VSB_SMP tab. Select the toolchain to be diab. Check the Enable WDB Target Agent to enable easy debugging. Uncheck Add support to project for the BSP validation test suite. Click Next.
5. Click Next on the Options screen
6. On the Configuration Profile window, select the Profile PROFILE_STANDALONE_DEVELOPMENT. Click Finish.
7. Import the datamover source file into the VIP_SMP project.
8. Go to File -> Import -> General -> Archive File as shown in the diagram below:



9. Click “Next.”
10. At From archive file, browse to the datamover_dkm.zip directory.
11. At Into folder, browse to the VIP_SMP project.

12. Click “Finish.”
13. You should able to see that the datamover_dkm folder has imported into your project.



14. Select the VIP_SMP project and build it by going to Project -> Build All

Deliverables

FPGA, Linux, and Nios II Binaries

The reference design deliverables include binaries that can be used to run the reference design directly. The Linux binaries are delivered as an archive file accessible at http://releases.rocketboards.org/release/2014.09/cv-datamover-ed

Folder	Folder	Folder	File	Description
cv_datamover_ed	bin	Linux	altera-datamover-image-socfpga_cyclone5.tar.gz	Zipped Root filesystem
			sdimage.bin	SD Card Image

			altera-datamover-image-socfpga_cyclone5.dtb	Device Tree Binary
			Preloader-mkimage.bin	Preloader Image
			vmlinux	Linux Kernel Executable
			zImage	Linux Kernel Compressed Image
		FPGA	Datamover_5csxfc6.sof	FPGA Configuration SOF
			Datamover_5csxfc6.flash	FPGA Configuration Flash
		Nios II	Datamover_demo.elf	Nios II Application Executable
			Datamover_demo.flash	Nios II Application Flash Image

VxWorks and HWLibs Binaries

The VxWorks and HWLibs binaries are delivered as an archive file accessible at https://www.altera.com/support/support-resources/download/rtos_tools.html

The binaries archive contains the following items:

Folder	Folder	Folder	Item	Description
Altera_datamover_ed	bin	vxworks	vxw_datamover_sdmmc.img	VxWorks SD Card Image
			vxWorks	VxWorks Kernel

Hardware Design

The hardware design is delivered as an archive file accessible at http://releases.rocketboards.org/release/2014.09/cv-datamover-ed

Some of the relevant included files and folders are:

Folder	Folder	Item	Description
cv_datamover_ed	hw	ip/	Folder containing IP files
		datamover_5csxfc6.qpf	Quartus Project File

		datamover_5csxfc6.qsf	Quartus Setting File
		datamover_5csxfc6.qsys	Qsys File
		soc_system_timing.sdc	Timing file
		soc_system_board_info.xml	SoC system board XML file
		hps_common_board_info.xml	HPS XML file
		datamover_top.v	Top Level Verilog File

Nios II Application Source

The Nios II application source is delivered as an archive file accessible at http://releases.rocketboards.org/release/2014.09/cv-datamover-ed

Folder	Folder	Folder	Item	Description
cv_datamover_ed	sw	niosii_hal	datamover_demo.c	Nios II Application Source

Linux Patch

The Linux patch is delivered as an archive file accessible at http://releases.rocketboards.org/release/2014.09/cv-datamover-ed

Folder	Folder	Folder	Item	Description
cv_datamover_ed	sw	linux	altera_datamover_yocto.patch	datamover yocto patch

Linux Yocto Recipes

All the necessary files to build the Linux kernel, drivers, applications, and root filesystem are delivered as a set of Yocto recipes accessible through the git trees at http://rocketboards.org/gitweb

Folder	Folder	Tag
Yocto Recipes	Poky-socfpga.git	ACDS14.0_REL_GSRD_PR

VxWorks

The VxWorks source is delivered as an archive file accessible at https://www.altera.com/support/support-resources/download/rtos_tools.html

Folder	Folder	Folder	Item	Description

Altera_datamover_ed	sw	vxworks	datamover_dkm.zip	Datamover DKM Project

HWLibs

The Hwlibs source is delivered as an archive file accessible at https://www.altera.com/support/support-resources/download/rtos_tools.html

Folder	Folder	Folder	Item	Description
Altera_datamover_ed	sw	hwlibs	Altera-SoCFPGA-DataMover-CV-GNU.tar.gz	Datamover HwLibs Example Application

Reference

1. GSRD User Manual