

# **ARM DS-5 Tools and Avnet ZED Series**

**#7**

**DS-5 Streamline and Avnet ZedBoard or MicroZed**



Sept 2013  
Version 02

## Table of Contents

ARM DS-5 Tools and Avnet ZED Series .....	3
Required Installations .....	4
Technical Support .....	5
Using Streamline to Analyze Software Performance.....	6
Install Xaos on ZedBoard.....	8
Configure Streamline for Xaos .....	13
Examine the Streamline Profile Report .....	19
Appendix I – Install the DS-5 Linux Examples .....	26
Appendix II – Modifications to the Speedway Linux Kernel .....	28
Appendix III – Install and Run X-server on the Host .....	35
Appendix IV – Streamline Capture Options .....	38
Appendix V – Building Gator .....	39
Revision History .....	42
Resources .....	42

## ARM DS-5 Tools and Avnet ZED Series

This tutorial is one in a series of step by step instruction manuals. Together they document the procedures necessary to utilize the ARM Development Studio 5 (DS-5™) Software Suite and the DSTREAM Debugging tools with the Avnet Zynq Evaluation and Development (ZED) boards. These tutorials can be used on their own, or in combination with Avnet online videos and OnRamp Technical Session™.

The ARM software and hardware tools provide a powerful debugging suite for processor-based systems built around the dual Cortex-A9 cores present in the Xilinx Zynq SoC, at the heart of the Avnet ZED boards. A Linux software developer can simultaneously debug applications and kernel module code, with separate control over each thread. You can step through Linux boot code, first stage bare metal boot code, and bare metal applications. When used in concert with the Xilinx Vivado tools for FPGA fabric development, the ARM debugger and Internal Logic Analyzer (ILA) IP can be cross-triggered to stop on software and hardware breakpoints, or when a hardware event occurs. For difficult-to-isolate intermittent faults, DS-5 provides access to the Cortex-A9 on-chip Trace facility. Once your embedded system is running correctly, DS-5 uses Streamline, a graphical system profiler, to identify performance bottlenecks in your design to ensure top-shelf operation.

This tutorial series begins with the most basic tool configuration and board connection. It takes you all the way through to the most complex aspects of hardware/software co-debugging to root out design errors that are otherwise apparent only in very complex use cases, or worse, after a product is released. Together the ARM DS-5 tools, Xilinx Vivado and Avnet ZED boards provide an unparalleled combination to compress design timelines, cut project costs and optimize your product for the marketplace.

## Required Installations

### Software

The recommended software for this tutorial series is:

- ARM Development Studio 5 (Exact version used is 5.14, build 1702)
- Xilinx ISE WebPACK 14.5 (Free license and download from Xilinx website)
- Cypress CY7C64225 USB-to-UART Bridge Driver (for ZedBoard serial output)
- Silicon Labs CP2104 USB-to-UART Bridge Driver (for MicroZed serial output)
- Tera Term (Exact version used is V4.75)
- Xilinx Software Development Kit, version 14.5
- For hardware/software co-debugging, Xilinx Vivado 2013.2

### Hardware

The targeted hardware consists of the following:

- PC workstation with at least 5 GB RAM, 30GB free hard disk space, Windows 7 64-bit operating system, and a wired GB Ethernet connection
- Available SD card slot on PC or external USB-based SD card reader
- One of:
  - Avnet ZedBoard Kit (**AES-Z7EV-7Z020-G**)
    - USB cable (Type A to Micro-USB Type B)
    - 4GB SD card
    - 12v Power supply
  - Avnet MicroZed Kit (**AES-Z7MB-7Z010-G**)
    - USB cable (Type A to Micro-USB Type B)
    - 4GB SD card
- Avnet ZedBoard Debug Adapter Kit (**AES-ZBDB-ADPT-G**)
  - 14-pin Xilinx PC4 ribbon cable
- ARM DSTREAM unit and Keil pod with wide cable connector
  - 20-pin JTAG ribbon cable
  - USB cable (Type A to Printer)
  - 5v Power supply
- CAT-5 Ethernet cable

## Technical Support

For technical support with any of the instructions, please contact your local Avnet/Silica FAE or visit the support forums:

<http://www.zedboard.org/forum>

<http://www.microzed.org/forum>

Additional technical support resources are listed below.

*ZedBoard Kit/MicroZed Kit support page with Documentation and Reference Designs*

<http://www.zedboard.org/content/support>

<http://www.microzed.org/content/support>

For Xilinx technical support, you may contact your local Avnet/Silica FAE or Xilinx Online Technical Support at [www.support.xilinx.com](http://www.support.xilinx.com) . On this site you will also find the following resources for assistance:

- Software, IP, and Documentation Updates
- Access to Technical Support Web Tools
- Searchable Answer Database with Over 4,000 Solutions
- User Forums
- Training - Select instructor-led classes and recorded e-learning options

Contact your Avnet/Silica FAE or Avnet Support for any additional questions regarding the reference designs, kit hardware, or if you are interested in designing any of the kit devices into your next design.

<http://www.em.avnet.com/techsupport>

For ARM technical support, you may contact your local Avnet/Silica FAE or ARM Online Technical Support at [www.arm.com/support](http://www.arm.com/support) .

## Using Streamline to Analyze Software Performance

In this tutorial we will boot the ZED target using the Linux kernel from tutorial #6 as the base, and use only an Ethernet connection between our host and the ZED target (no DSTREAM). The kernel was rebuilt with some additional configuration changes specific to Streamline. In addition, the kernel has been modified to include an updated version of the dynamic loader (ld.so) that is required to execute the application examples included with the DS-5 software suite.

Linux provides a platform to run a fractal generation program called Xaos (pronounced “Kaos”) that is complex enough to yield some interesting statistics for analysis by the Streamline tool.

Xaos is one of the Linux examples included with DS-5. If you have not imported the examples into your workspace, do this now. If you require detailed instructions, consult “Appendix I – Install the DS-5 Linux Examples”.

Set the ZED target Boot Mode to SD boot as shown below:

### For ZedBoard:

To begin the procedure, set the ZedBoard Boot Mode to SD boot using jumpers JP11 to JP7 set to the following:

	<b>JP11</b>	<b>JP10</b>	<b>JP9</b>	<b>JP8</b>	<b>JP7</b>
Position	SIG-GND	3V3-SIG	3V3-SIG	SIG-GND	SIG-GND

### For MicroZed:

To begin the procedure, set the MicroZed Boot Mode to JTAG only using jumpers JP3 to JP1 set to the following:

	<b>JP3</b>	<b>JP2</b>	<b>JP1</b>
Position	2-3	2-3	1-2

Connect an Ethernet cable and micro-USB cable between the ZedBoard and your host PC, as per previous tutorials.

This tutorial assumes you already have your SD Card loaded with files from tutorial #6. You will need only to update the Linux kernel image on the card. This file is part of the download package that included this tutorial. Browse to the location on your host where the download package was decompressed, and look for the following folder:

**<Download package folder>\Support07**

For the purposes of this tutorial we will assume the file exists in folder:

**C:\OnRamps\Support07**

To prepare to boot the target, copy the **ulmage** and **uramdisk.image.gz** files from the support folder to the root directory of your SD card, replacing the existing file.

**C:\OnRamps\Support07\ulmage**

Insert the card into the SD card slot on the underside of the target.

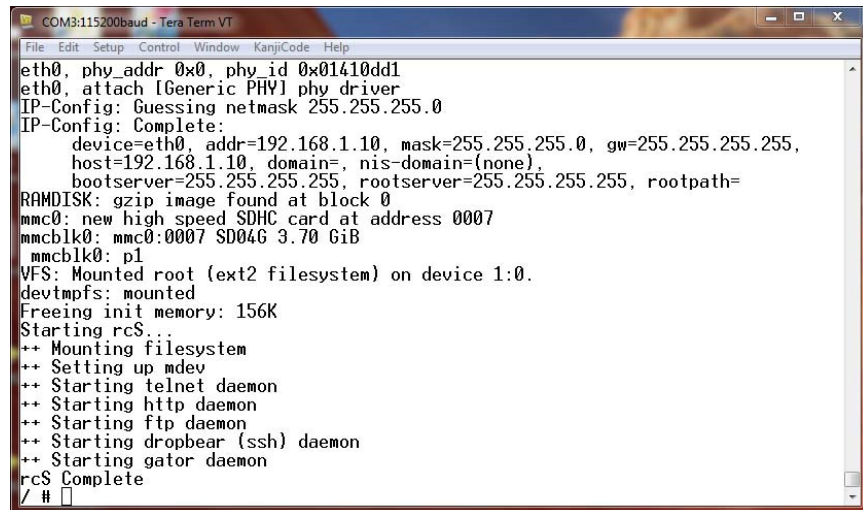
The only changes made to the files from tutorial #6 is that the Linux kernel (ulmage) has been rebuilt with configuration changes required for Streamline operation. If you would like to do this on your own instead of using the supplied files, please see Appendix II. The ramdisk has been modified to include the gator daemon and kernel module, and to start the daemon when Linux boots. Compilation of Gator is covered in Appendix V.

Install an X server on your Windows host to accept graphical output from Xaos. For detailed instructions, consult “Appendix III – Install and Run X-server on the Host”.


You will need to install the DS-5 examples in your workspace before you can complete this tutorial. For detailed instructions, refer to “Appendix I – Install the DS-5 Linux Examples”.

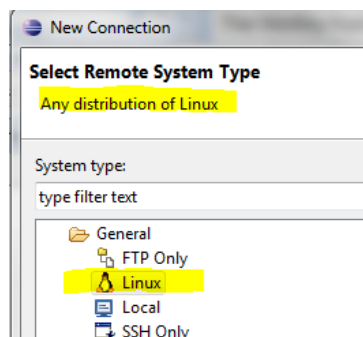
## Install Xaos on ZedBoard

1. Open the DS-5 IDE on your host PC. For the purposes of this tutorial series, we will use the same workspace called **ZedBoard DS-5 WS**.
2. Power the ZED target, and watch for the blue configuration LED to illuminate. As soon as this happens, open Tera Term Pro and allow your system to boot to the Linux prompt.



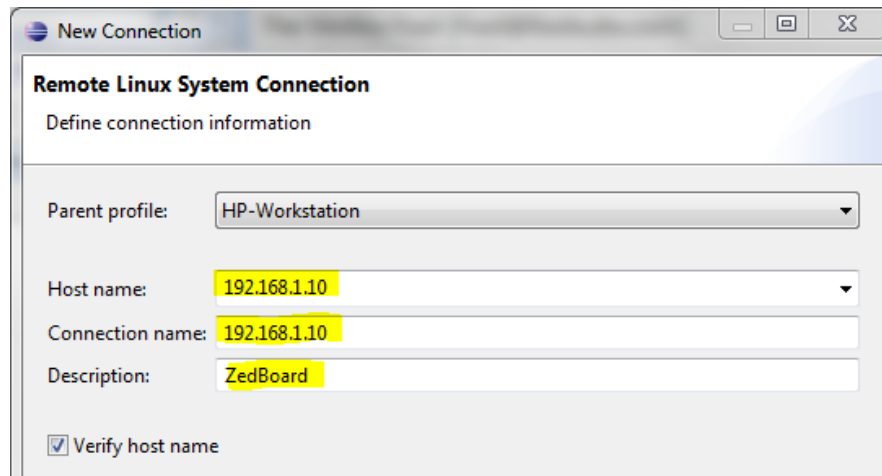
```
COM3:115200baud - Tera Term VT
File Edit Setup Control Window KanjiCode Help
eth0, phy_addr 0x0, phy_id 0x01410dd1
eth0, attach [Generic PHY] phy driver
IP-Config: Guessing netmask 255.255.255.0
IP-Config: Complete:
    device=eth0, addr=192.168.1.10, mask=255.255.255.0, gw=255.255.255.255,
    host=192.168.1.10, domain=, nis-domain=(none),
    bootserver=255.255.255.255, rootserver=255.255.255.255, rootpath=
RAMDISK: gzip image found at block 0
mmc0: new high speed SDHC card at address 0007
mmcblk0: mmc0:0007 SD04G 3.70 GiB
mmcblk0: p1
VFS: Mounted root (ext2 filesystem) on device 1:0.
devtmpfs: mounted
freeing init memory: 156K
Starting rcS...
++ Mounting filesystem
++ Setting up mdev
++ Starting telnet daemon
++ Starting http daemon
++ Starting ftp daemon
++ Starting dropbear (ssh) daemon
++ Starting gator daemon
rcS Complete
/ #
```

3. Switch to the DS-5 Debug Perspective by clicking the icon (  ) at the upper right of the DS-5 window.
4. Select the **Remote Systems** tab. If you have the ZED target entry for 192.168.1.10 that we created in tutorial #5, skip to step 7. If not, right-click on **Local** and select **New | Connection**. In the New Connection window, select **Linux** and click the **Next** button.

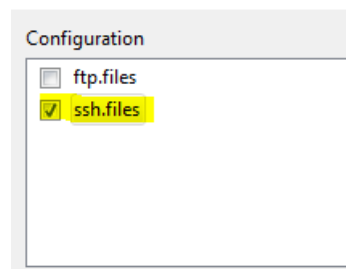




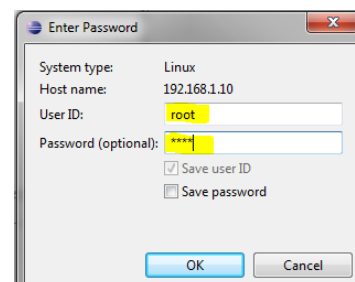
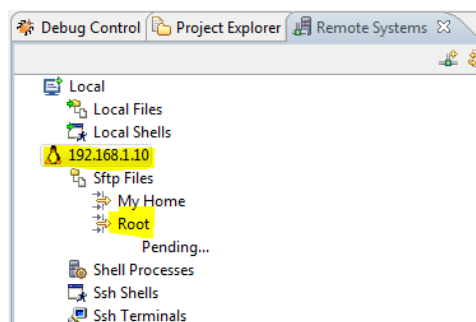
5. Enter the IP address of the ZedBoard for the Host Name (192.168.1.10), which will also be copied into the Connection Name. For Description, enter ZedBoard and click the **Next** button. The Parent profile is dependent upon your host computer, and should be left unchanged.



6. In the Configuration panel, select **ssh.files**. We will be using secure shell protocol to communicate with Linux on the ZedBoard, rather than FTP protocol. All other settings in the configuration can be left at their default values, so click the **Finish** button.

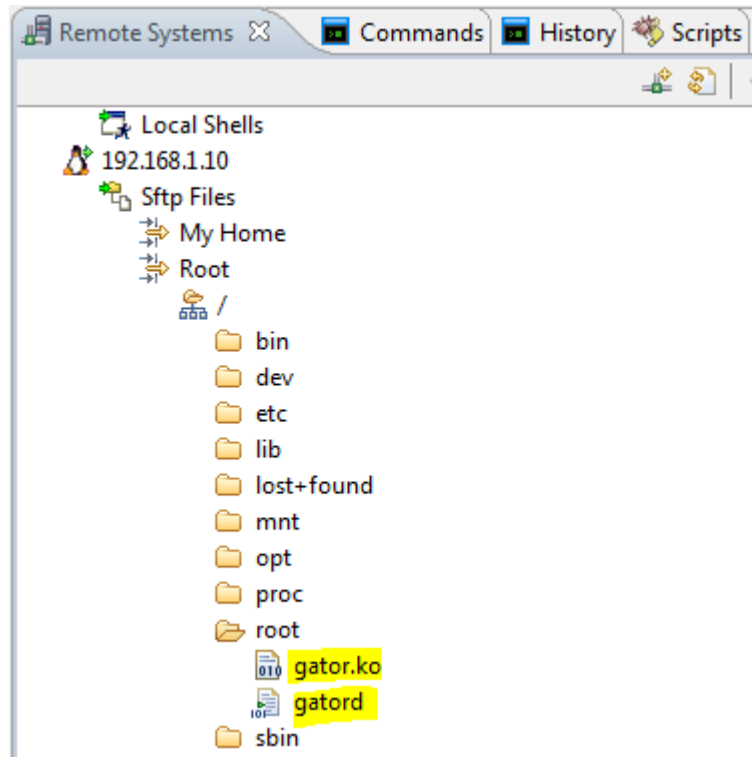


7. You should now see a new Linux connection in the Remote Systems tab, which you can expand as shown. If you are prompted for a **UserID** and **password**, enter **root** for both fields and click the **OK** button.

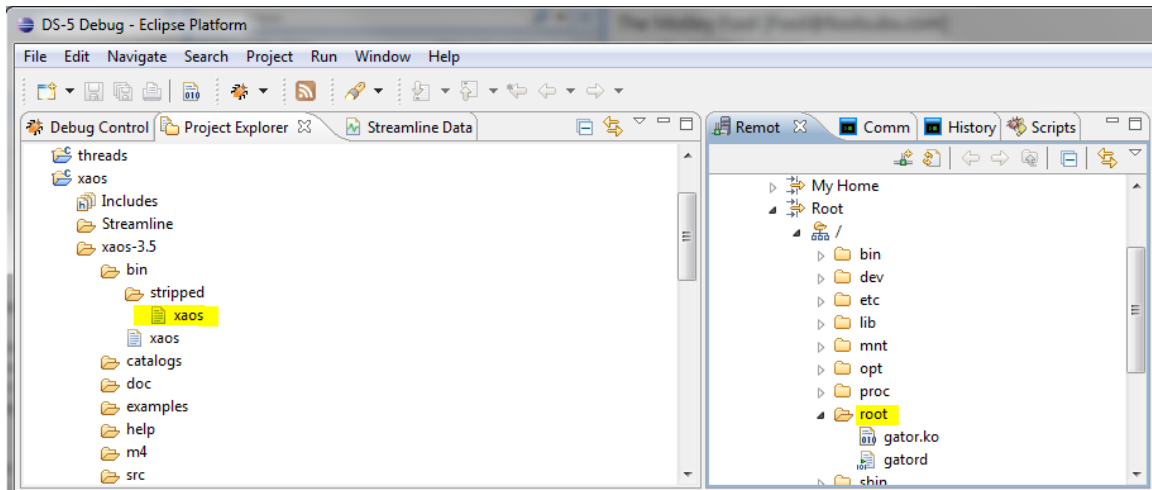


8. Once your credentials have been accepted, you can expand the **Root** entry completely. Further expand the Root folder, to show its current contents. The root file system has been built with the gator daemon and the gator kernel module. If you would like to build gator on your own, see the instructions in Appendix V.

Gator is a service that Streamline uses to collect statistics on programs while they are running. This also happens to be the directory where we want to copy over the Xaos application we will be analyzing.

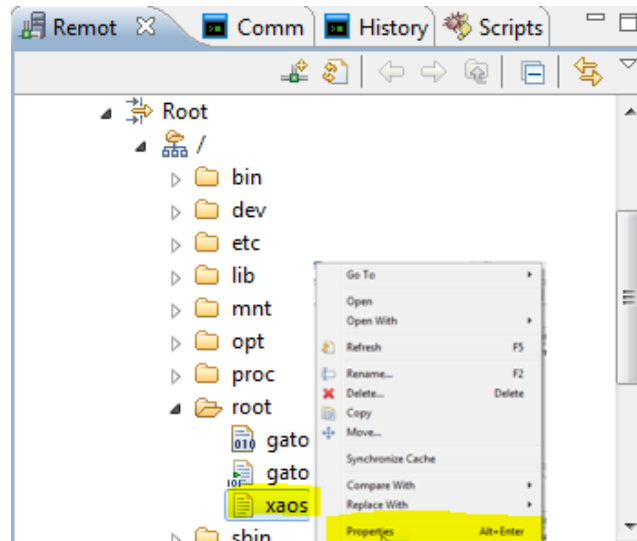


9. One of the nice features of Eclipse is to relocate tabs wherever we choose. This is useful right now since we want to move files from the Project Explorer tab to the Remote System tab (representing from the host to the ZED target). Click on the Remote System tab (the actual tab, not just in the pane) and hold the left mouse button down. Drag the tab into the next pane to the right, where the Commands tab is located. Release the mouse button and the Remote Systems tab will move to its new location. We can now select the Project Explorer tab and expand the xaos project, as shown, with both tabs in view.

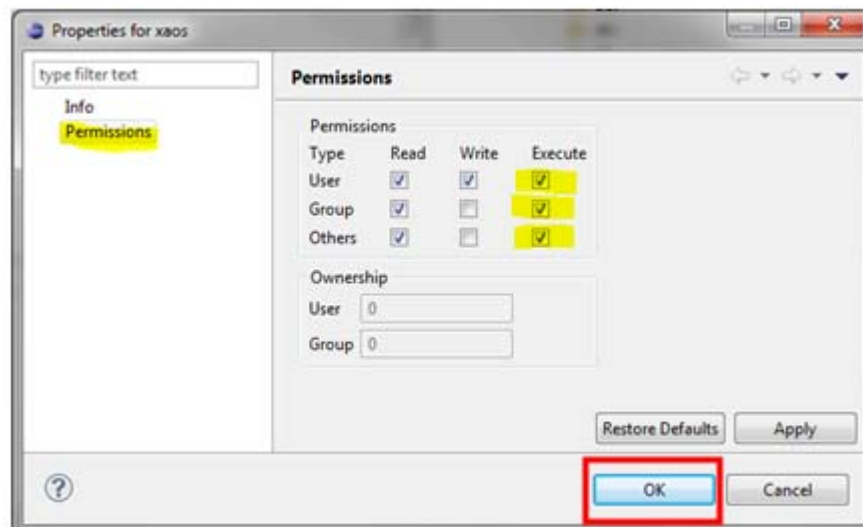


If you do not see the Project Explorer tab, in the main menu select **Window | Show View | Project Explorer** to open the panel.

10. We want to copy the **xaos** stripped executable file (no debug information, because we don't need it and the stripped version is smaller) over to the **ZED** target. To do this, simply drag and drop the **xaos** entry from the **stripped** folder in the Project Explorer tab over to the **root** folder in the Remote Systems tab. Now move the focus off of **xaos** in the Remote Systems tab by selecting the **gator** file, then select **xaos** once again (this is a bug in the Eclipse framework that prevents the proper menu from showing on a right-click). Right-click on the **xaos** entry and select **Properties** from the drop-down menu.



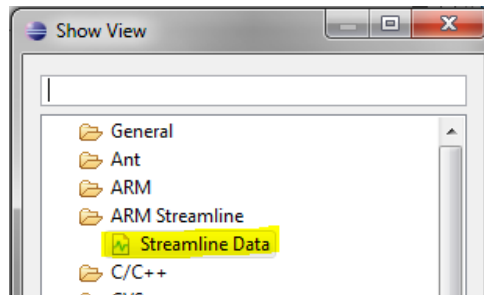
11. We need to make the **xaos** program executable on the target, so in the Properties for **xaos** window, select **Permissions** and click on all three **Execute** checkboxes. Click the **Apply** button, followed by the **OK** button.



**Xaos** is now ready to run on the **ZED** target as a Linux application.

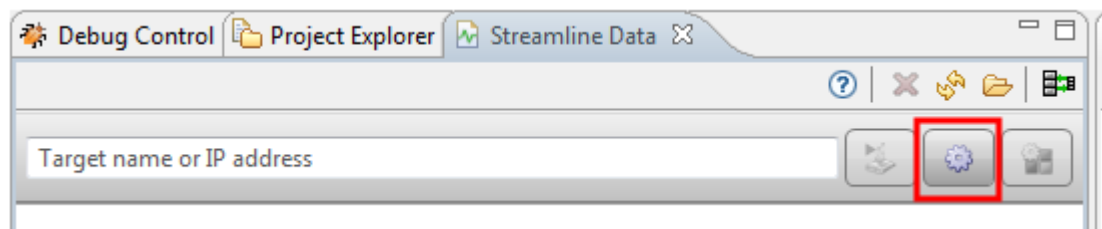
## Configure Streamline for Xaos

1. If you do not have tab in the Project Explorer pane for **Streamline Data**, open one by selecting from the main menu: **Window | Show View | Other**.

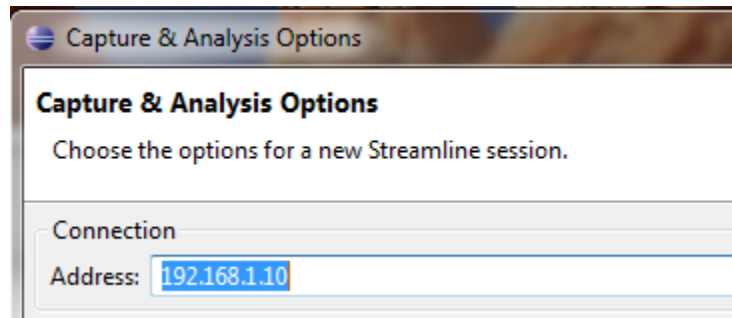


Expand the **ARM Streamline** folder and select **Streamline Data**. Click the **OK** button.

2. In the Streamline Data Tab, click the **Capture Options** button.



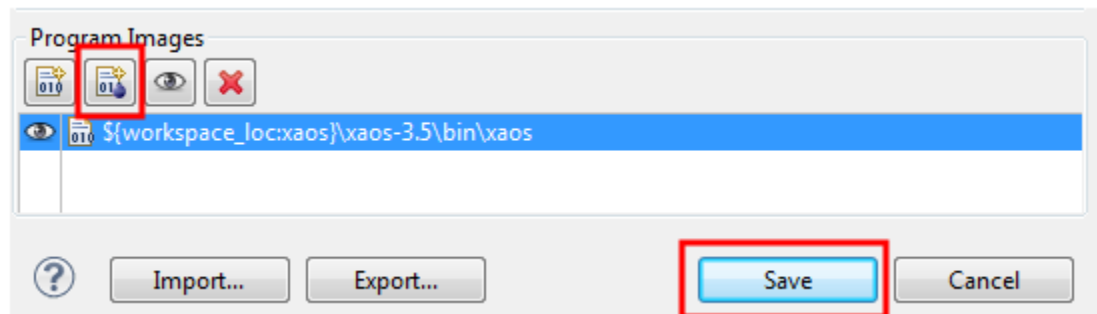
3. In the Capture and Analysis Options window, enter the address of the ZED target.




4. In the Program Images section, click the **Add ELF image from workspace** button and choose the file:

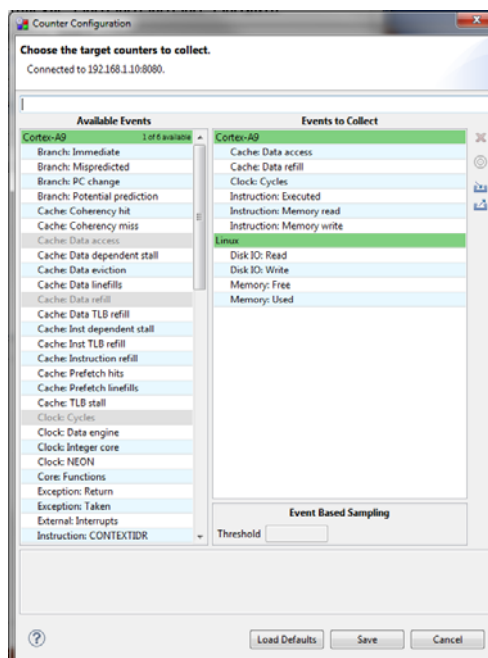
**xaos/xaos-3.5/bin/xaos**

This binary contains debug information that will be loaded for the Streamline analysis.



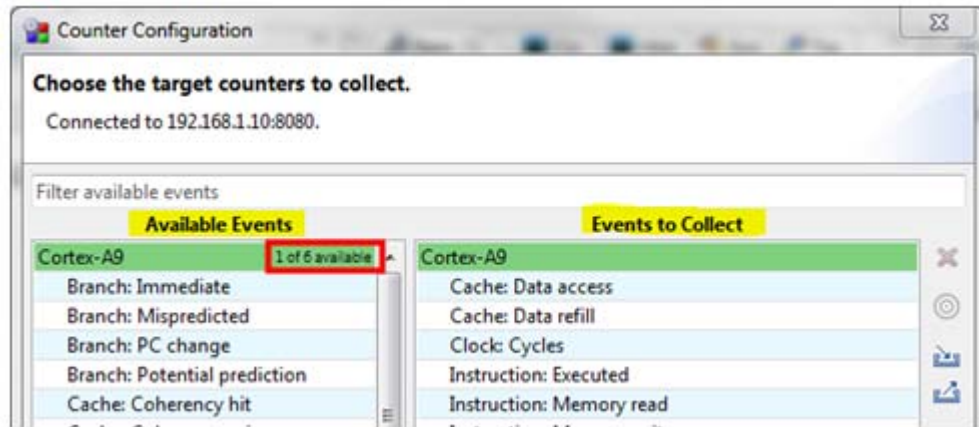
We will accept all the rest of the defaults on this screen, so click the **Save** button. However, if you want more information on other options, see “Appendix IV – Streamline Capture Options”.

5. There is a variety of information that Streamline can collect. We can choose a set of counters to pick only the information we are interested in. Click the **Counter Configuration** (  ) button to examine and change the counters.



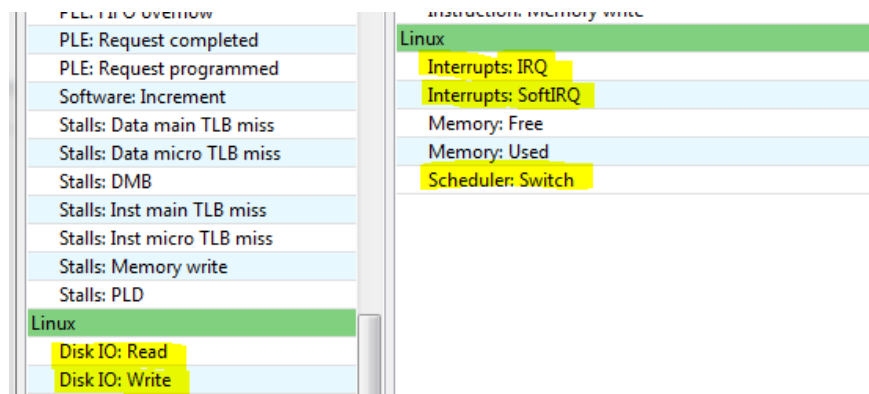
Note that gator must be running on the ZED target to access this window.

6. In the Counter Configuration window you can see the default set of counters DS-5 will use under the **Events to Collect** column. With the ARM Cortex-A9 dual cores in the Zynq chip of the ZED target, you can select up to 6 counters – the number remaining is shown under the **Available Events**<sup>1</sup> column.



You can scroll down the list of Available Events to see the Linux Events you can select. To choose an item, simply select and drag it to the Events to Collect column. If you would like to remove an item from the Events to Collect column, select it and click on the delete button (✖). You can get a short description of each counter by hovering over your selection.

7. Under the Linux Events to Collect, remove the two Disk IO counters, as xaos is memory intensive application and has few disk accesses. Add in the two Interrupt counters, and the Context switch as shown.



Click the **Save** button to retain the configuration on the target.


<sup>1</sup> The Clock: Cycles counter can be added for “free”, as it doesn’t incur a count against the total.

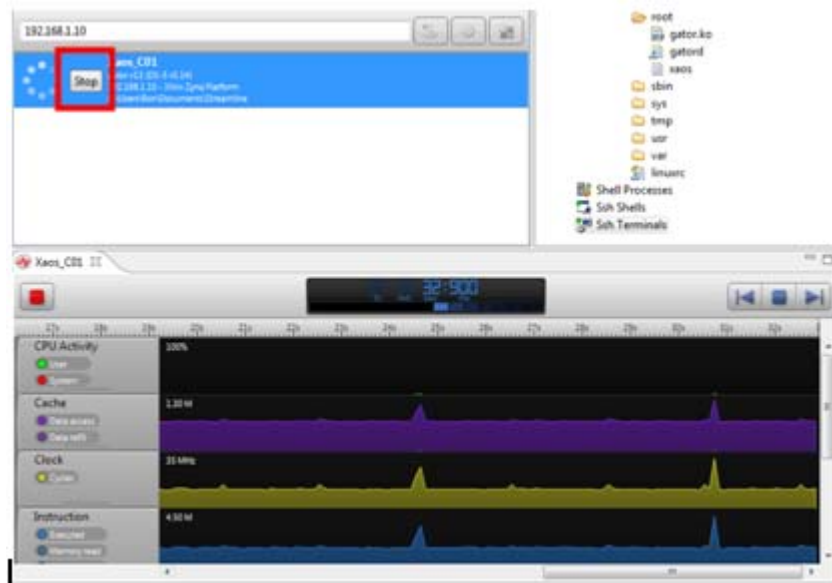
8. The Xaos application we intent to profile will send its output to an X-server window on your host, which must be running before Xaos can be initiated. If you have not yet installed and started the X-server, see the instructions in Appendix III before proceeding.

Assign the host as the remote system to receive the Xaos graphical output. You may use the Teraterm Pro window, or a terminal window launched from the Remote Systems tab. We will use the latter to enter the following command lines for xaos:

```
cd ~/root
export DISPLAY=192.168.1.50:0.0
```

If your host has a different IP address than 192.168.1.50, substitute your host address.

9. To begin collecting data, click the Start Capture button (  ). You will be prompted for a location on the host to save the data collected; accept the default location but change the name to **Xaos\_C01.apc**. Click the **Save** button.



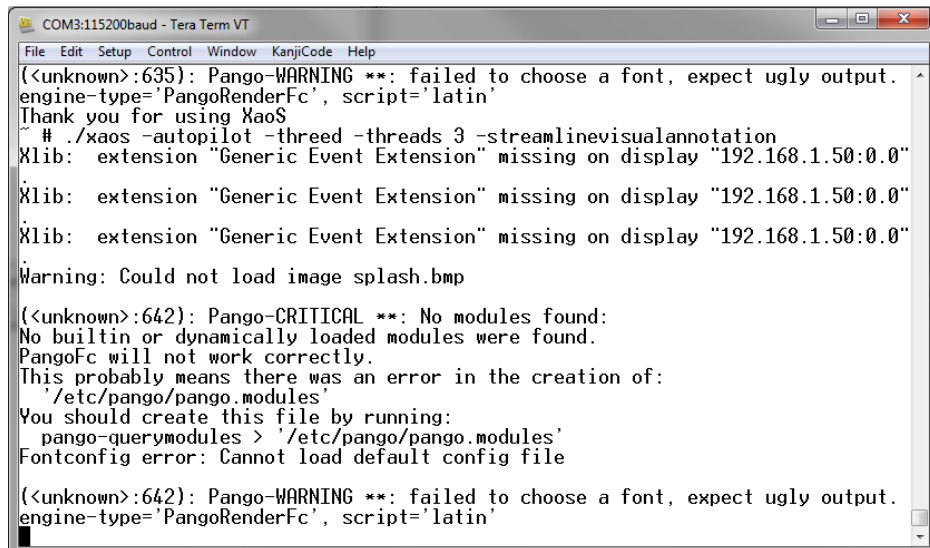
At this point you will see the capture file appear and statistics will start scrolling across the real time capture window, as shown above. You can terminate the capture with the **Stop** button in the upper panel, next to the file name, but don't do that right now.



10. Start the Xaos application using the same window where you assigned the remote display address.

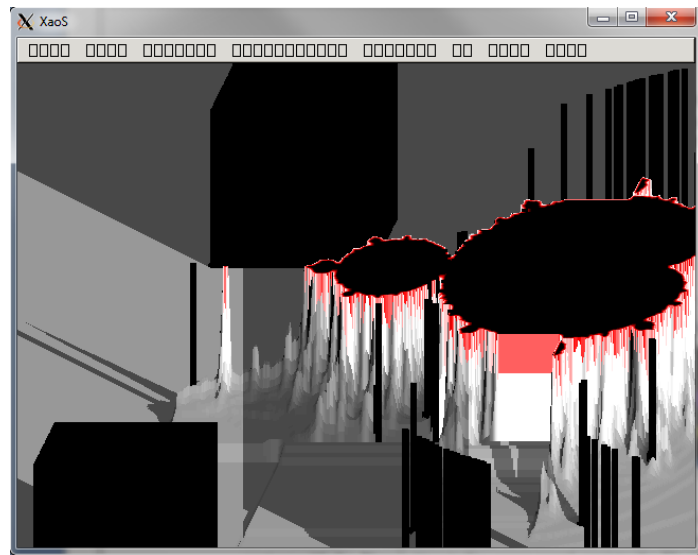
**`./xaos -autopilot -threed -threads 3 -streamlinevisualannotation`**

You will see a number of warnings at the startup due to missing fonts and library extensions, but you can safely ignore them as they do not affect the operation of Streamline gathering statistics.



```
COM3:115200baud - Tera Term VT
File Edit Setup Control Window KanjiCode Help
(<unknown>:635): Pango-WARNING **: failed to choose a font, expect ugly output.
engine-type='PangoRenderFc', script='latin'
Thank you for using XaoS
~ # ./xaos -autopilot -threed -threads 3 -streamlinevisualannotation
Xlib: extension "Generic Event Extension" missing on display "192.168.1.50:0.0"
Xlib: extension "Generic Event Extension" missing on display "192.168.1.50:0.0"
Xlib: extension "Generic Event Extension" missing on display "192.168.1.50:0.0"
Warning: Could not load image splash.bmp
(<unknown>:642): Pango-CRITICAL **: No modules found:
No builtin or dynamically loaded modules were found.
PangoFc will not work correctly.
This probably means there was an error in the creation of:
'/etc/pango/pango.modules'
You should create this file by running:
pango-querymodules > '/etc/pango/pango.modules'
Fontconfig error: Cannot load default config file
(<unknown>:642): Pango-WARNING **: failed to choose a font, expect ugly output.
engine-type='PangoRenderFc', script='latin'
```

11. The xaos command starts the application unattended mode, with three threads and displays in pseudo-3D. The X Server application we started earlier on our host receives the data and should look similar to the image below.



While **xaos** is running, you will see a significant change in the real-time data scrolling by on the capture screen. After a minute or so, click the **Stop** button in the Streamline Data tab to terminate the capture, and allow Streamline to



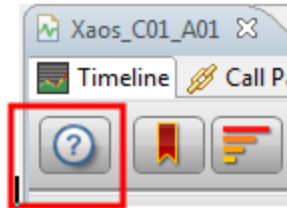
process the data file. Depending on how long you let the capture run, Streamline will take a few seconds to a few minutes to create a profile, which will appear in the window with the capture file and is saved in the same folder.

Once the file is processed, the real-time view will be replaced with actual data from the analysis file.

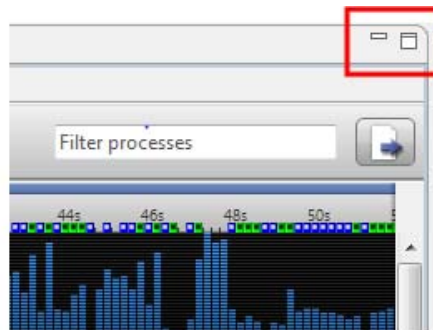
Xaos is still running on the ZED target. You can free up some host cycles by stopping it with **Ctrl-C** in your terminal window, which will terminate the data stream to the X-Server and close the display window. (Or, you can close the X-Window to terminate Xaos as well.)

## Examine the Streamline Profile Report

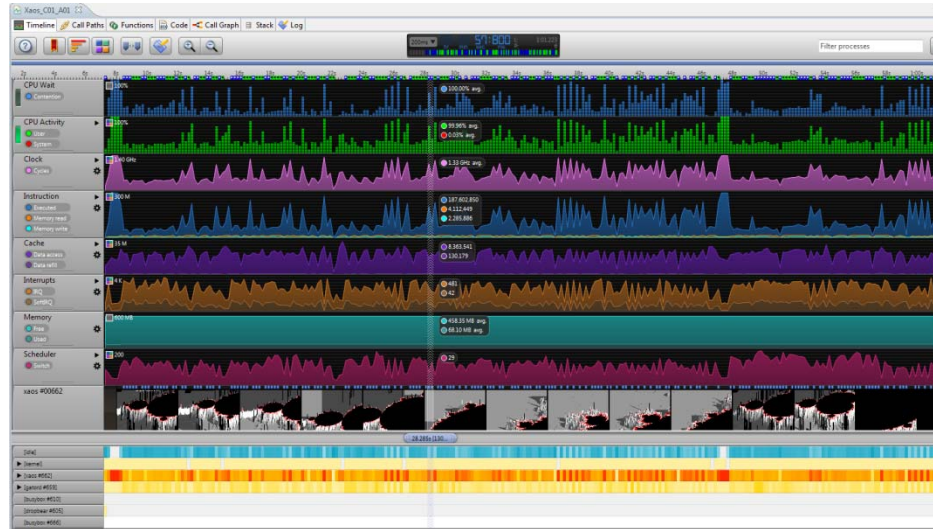
There is so much information available in a Streamline Profile report, and so many ways to analyze it, that we can only scratch the surface in a short tutorial. These instructions will get you started with the main features, and as you use Streamline with your own applications over time you will become familiar with the particular views that hold the most interest for you. More detailed information on all Streamline features can be available by clicking the Help button at the upper left of the Streamline Profile window.



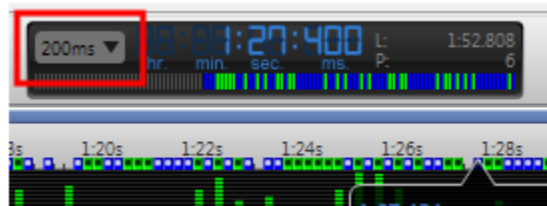
1. To make manipulation of the profile report a little easier, the first thing you can do is to expand the display window to the full screen. Click on the maximize button at the top right of the report window (once expanded, the maximize button is replaced with a restore button to return the display to its original size and location within DS-5).



2. The upper section of the report shows graphs of each counter we selected for capture during the xaos run. The lower section shows the processes that were running. Both sections can be manipulated to access different views of the statistics captured during the run.



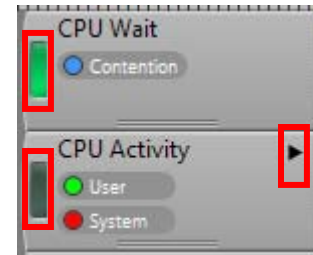
If you ran your capture session for about 1 minute, the **Timeline** tab should look similar to this one. If the data does not fill the screen, it may be due to the time resolution value in the timeline bar at the upper center. Click on the down arrow and select a value of 200 ms.



In the upper section the bar graphs illustrate the combined actions of all running processes as they affect the counters. In the lower panel, we see color coding to indicate where the most intensive resource usage occurs, with red showing the critical areas of highest use. As you look along the xaos process line, you can see many red areas for analysis.

3. Before we begin looking at what the data can tell us, let's get familiar with some of the controls.

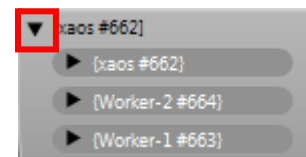
- a. The vertical bars on the left of legend are Process Focus buttons. Click on the **CPU Wait** bar to highlight areas where the CPU is in contention, when more processes are scheduled than there are CPUs to run. Click the **CPU Activity** bar to return to the original display. Click on the **arrow** to the right of CPU Activity to expand the view for each core.



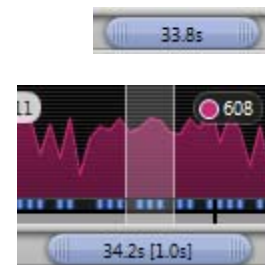
- b. Some of the lines have a configuration icon. Click on this to toggle a view to add additional counters to the line, each with its own color code.



- c. In the process section at the bottom, click on the horizontal arrows to drill down into the running processes. Each process also has a horizontal arrow to display channel information, where Streamline can display annotation details.



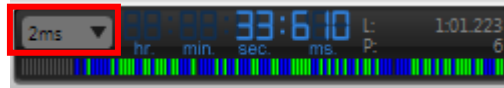
- d. The thumb control moves the current timeline position left or right as you select and move it. The current position within the time displays in the control. If you select either end of the control and hold the mouse key down, you can drag the selection line to a wider field. The control updates to show the width of the selected area.



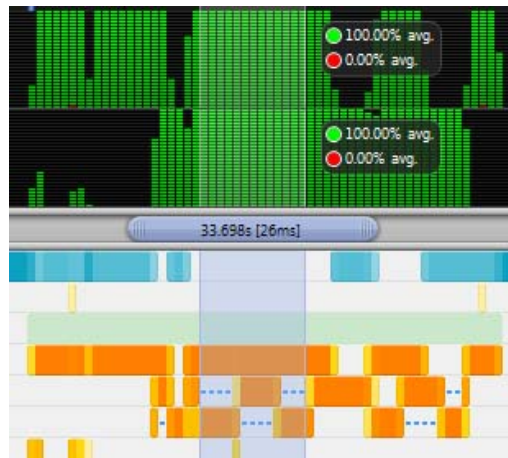
- e. Just above the timeline graphs is the Timeline Overview control. The large digits report the position of the mouse within the timeline. The small L on the right shows the timeline length, and the P the number of processes included. On the left side we have a magnification selector, which shows a dropdown menu for selecting a portion of the timeline for finer analysis.



4. Now let's examine the data we have. First, use the magnification selector in the Timeline Overview to view the timeline in much more detail. Select the dropdown and pick 2 ms from the list.

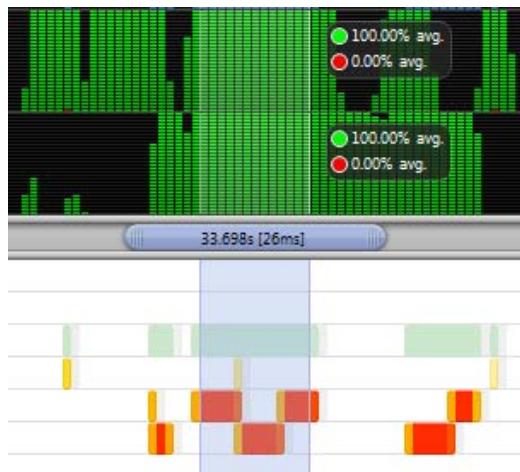



5. In the process field, expand the xaos entry to show each of the three running threads. Use the thumbwheel to move to an area where it looks like the xaos processes are using most of the CPU, and expand the thumb selection area to encompass the area of interest.

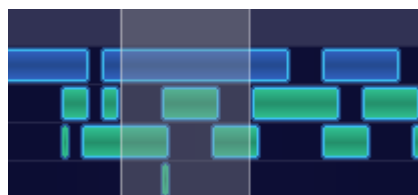


In the view above the CPU Activity has been expanded to show both CPUs, and the graph has been moved to the bottom of the display so that it sits right next to the process view. You can rearrange the graph lines by clicking in the gray control area on the left, and simply drag the line up or down to a new position.

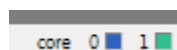
The dashed lines in blue indicate areas of contention. If you toggle the CPU activity and CPU wait views as shown previously in the controls description, you will get a better idea of how this can be used to highlight process stalls, and possible areas for code redesign.




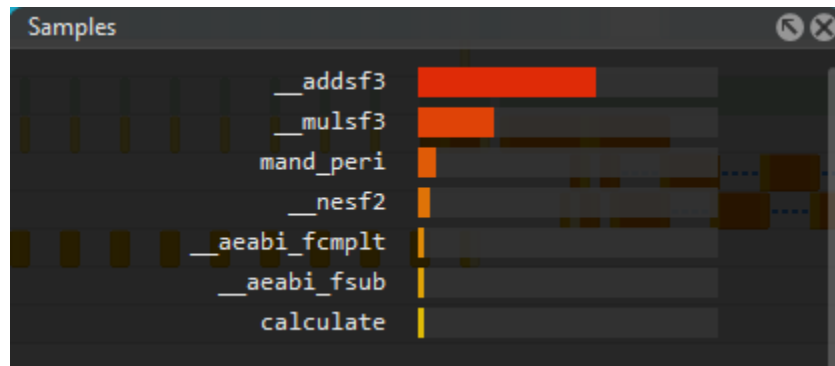
6. We can take the analysis one step further by using the X-ray View button(  ) to break out which processes are running on the individual cores.



Now the process lines are color coded to indicate which of the two cores each is running on.



7. Now that we have identified a section of the xaos code that looks promising for optimization, let's dive in further. Click on the Heads Up Display control (  ) in the toolbar above the graphs. When we originally configured Streamline, we added a symbols file, and the HUD makes use of it to show us which functions within xaos are consuming the CPU.

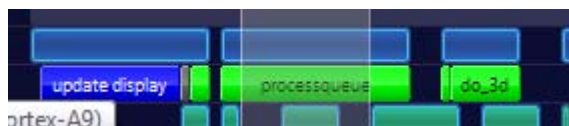


Here we can see some of the arithmetic calculations for addition and multiplication are heavy cycle consumers, which is not surprising considering the number of calculations that are required in a 3D fractal generation program.

These functions would likely be good candidates for parallelization using the Neon processor, or even a custom IP core within the PL on the Zynq chip.

If you slide the thumbwheel and the associated selection area back and forth over the data, you will see the values in the HUD change to reflect the current execution.

8. Streamline has an Annotate feature that allows an application to add time-stamped text, images and bookmarks at key points in the code. Click the arrow to expand one of the xaos threads to show annotation in the timeline view.



Click on the **Log** tab for a tabular representation of the annotation points.



9. If the program was compiled with frame pointer information, additional annotation can be added and accessed by Streamline. To compile a program using gcc with frame pointer information, the build argument is:

**-fno-omit-frame-pointer**

Select the **Call Paths** tab. Xaos was built using this argument, so Streamline can extract call stack information and display it under the Call Paths tab.

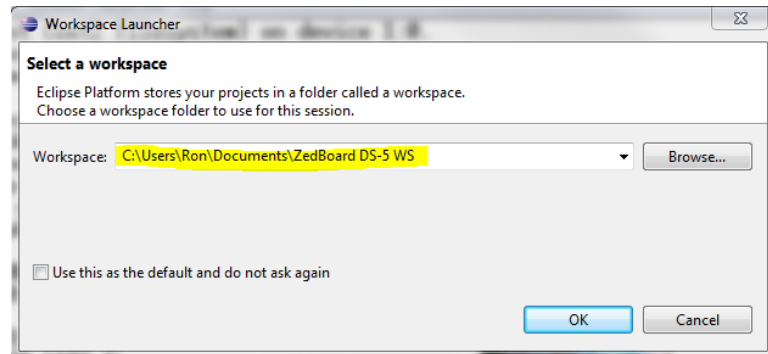
10. Additional detailed information is available in the **Functions**, **Call Graph** and **Stack** tabs. Call graphs is especially useful for tracing the flow of execution. The other tabs are useful to programmers to gather more detailed information about the operation of the program. Feel free to explore the information in these tabs.

This concludes the introduction to the Streamline profiling and analysis tool. You can use the restore button to return the display to its normal location within the DS-5 IDE.

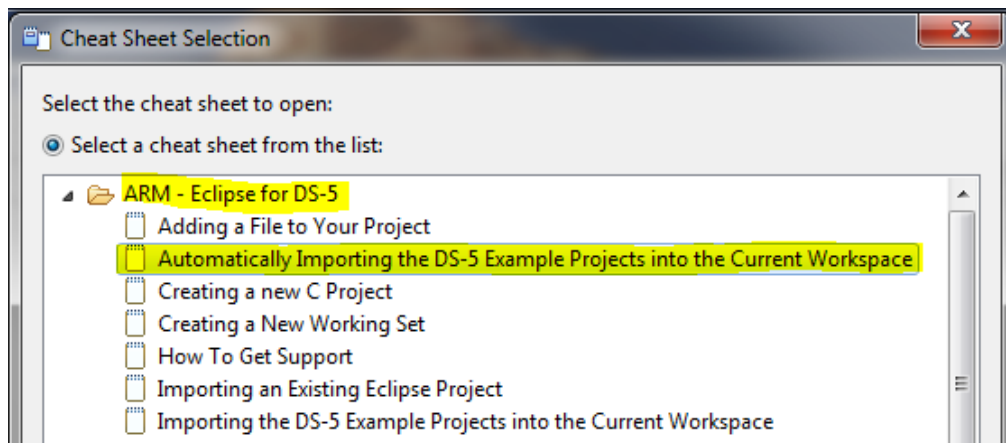
Additional Streamline information and exercises and labs are available from the ARM website.

## Appendix I – Install the DS-5 Linux Examples

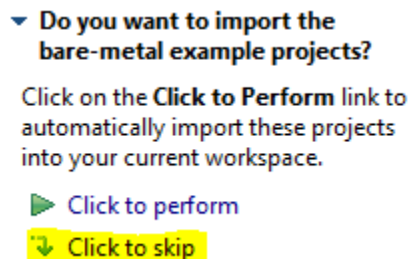
1. Launch the Eclipse DS-5 application on your host and select the same workspace you have been using throughout this tutorial series. For the purposes on this tutorial, we will assume it is called **ZedBoard DS-5 WS**.



2. Select the C/C++ Perspective. From the main menu, select **Help | Cheat Sheets**. Expand the **ARM – Eclipse for DS-5** entry and select **Importing the Automatically Importing the DS-5 Example Projects into the Current Workspace**. Click the **OK** button.



3. In the Cheat Sheets tab, **Click to skip** import of the bare-metal example projects.



4. **Click to perform** (or click to redo) to import the Linux application example projects.

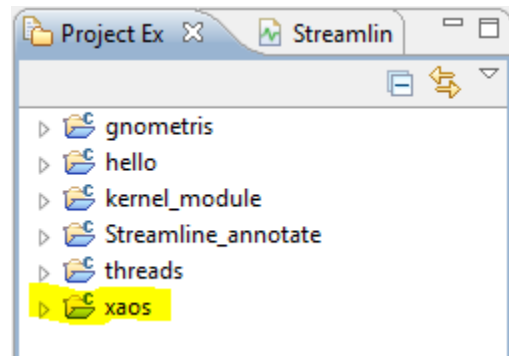
▼ Do you want to import the Linux application example projects?

Click on the **Click to Perform** link to automatically import these projects into your current workspace.

▶ Click to perform

↩ Click to skip

When the import is complete, there will be a number of new projects visible in the Project Explorer tab, one of which is the **xaos** program we need for the Streamline example.



If we were planning to build the Linux applications, you can see in the next Cheat Sheet item that there is an optional package to download and install for support libraries needed for compilation. The Xaos project is pre-built, so we don't need to do that now, but if you want to build Xaos or other of the Linux examples yourself, you will need to obtain and install the distribution package.

▶ Do you want to import the optional package containing the example Linux distribution project and the compatible headers and libraries?

You may close the **Cheat Sheets** tab at this point.

## Appendix II – Modifications to the Speedway Linux Kernel

1. Follow the normal steps you take to build the Linux kernel for your ZED target, but prior to the final make command a use menuconfig to update the build configuration. For complete details on how to build the Linux kernel, consult the Avnet Speedway for Implementing Linux on the Zynq 7000 SOC. Briefly, the procedure to update the kernel for Streamline use consists of the following steps:

- a. Clone the Linux source tree from the Xilinx repository.

**git clone git://git.xilinx.com/linux-xlnx.git <local dir>**

- b. Clean the build directories.

**make distclean**

- c. Configure the default kernel build parameters for the ZED target.

**make ARCH=arm xilinx\_zynq\_defconfig**

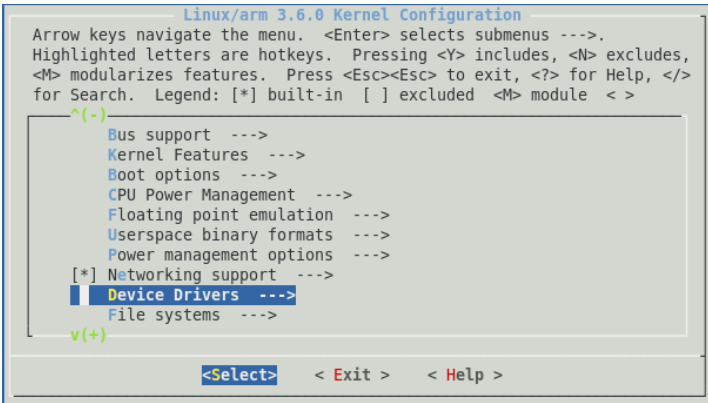
- d. Customize the build parameters. This is the extra step you must add to change the configuration for use with Streamline. If you are using the supplied ramdisk image, you will also need to update the configuration to use a larger ramdisk.

**make ARCH=arm menuconfig**

If you are familiar with the operation of menuconfig, or the .config file it creates, you can refer to this summary for all the required changes and skip to step e. If you need step by step details, continue on the following page.

- Update configuration for large ramdisk
  - **Device Drivers →Block Devices**
    - **Default number of RAM disks (change from 16 to 8)**
    - **Default RAM disk size (change from 16384 to 32768)**
- Add Streamline-specific configurations
  - **General setup →Kernel Performance Events and Counters**
    - **[\*] Kernel performance events and counters**
    - **[\*] Profiling Support**
  - **CPU Power Management**
    - **[\*] CPU Frequency scaling**
- Generate Kernel Debug Information and symbol file
  - **Kernel hacking**
    - **[\*] Tracers**
      - **[\*] Trace process context switches and events**

#### i. Select **Device Drivers**



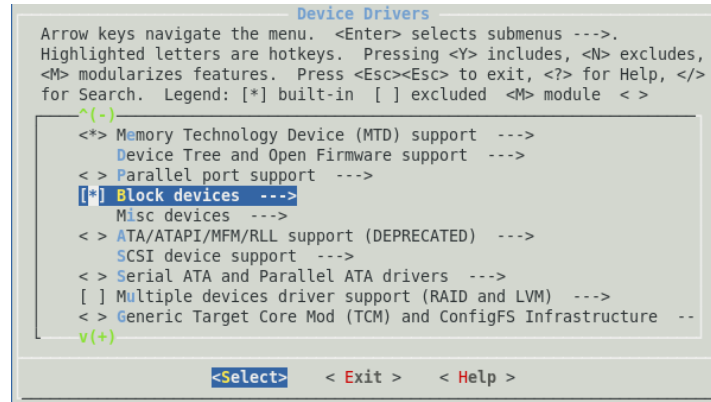
```
Linux/arm 3.6.0 Kernel Configuration
Arrow keys navigate the menu. <Enter> selects submenus --->.
Highlighted letters are hotkeys. Pressing <Y> includes, <N> excludes,
<M> modularizes features. Press <Esc><Esc> to exit, <?> for Help, </>
for Search. Legend: [*] built-in [ ] excluded <M> module <>

^(-)
Bus support --->
Kernel Features --->
Boot options --->
CPU Power Management --->
Floating point emulation --->
Userspace binary formats --->
Power management options --->
[*] Networking support --->
Device Drivers --->
File systems --->

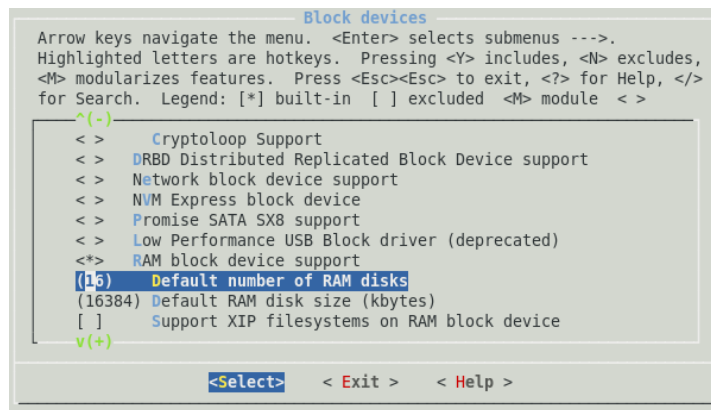
v(+)

<Select> <Exit> <Help>
```

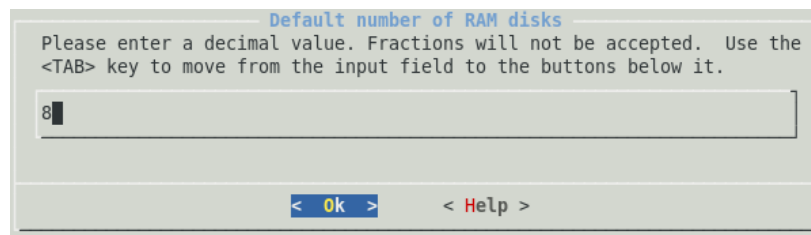
ii. Select **Block Devices**



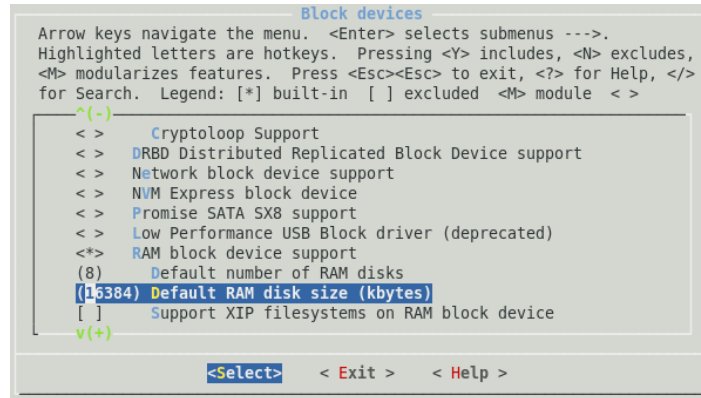
iii. Select **Default number of RAM disks**



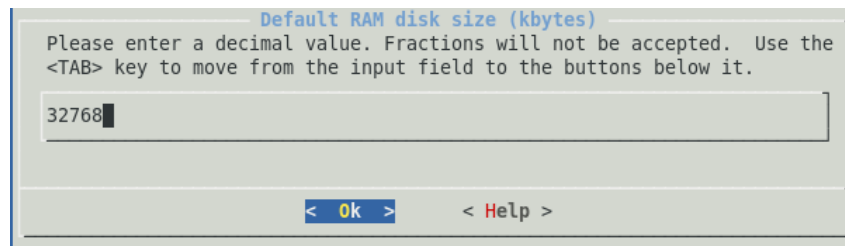
iv. Enter **8** for the number of RAM disks and click **OK**



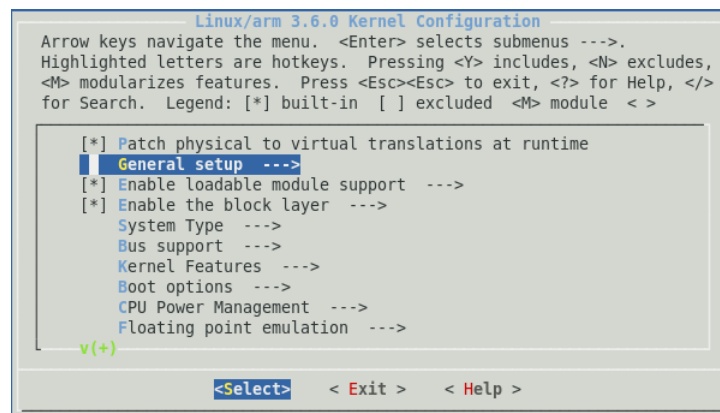
v. Select **Default RAM disk size (Kbytes)**



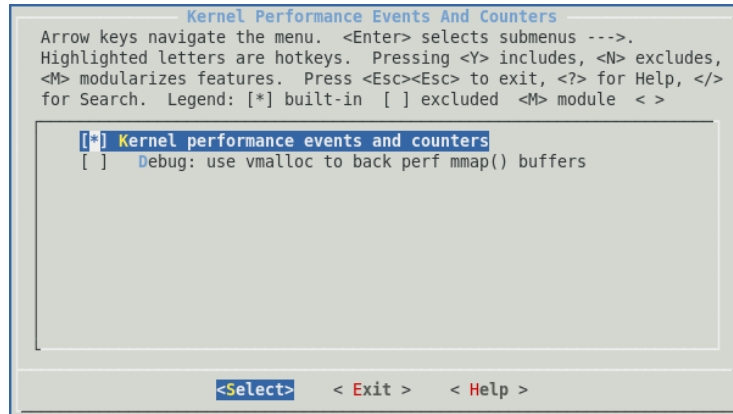
vi. Enter **32768** for the RAM disk size and click **OK**



vii. Click **Exit** twice to return to the main menu. Scroll to the top of and select **General setup**

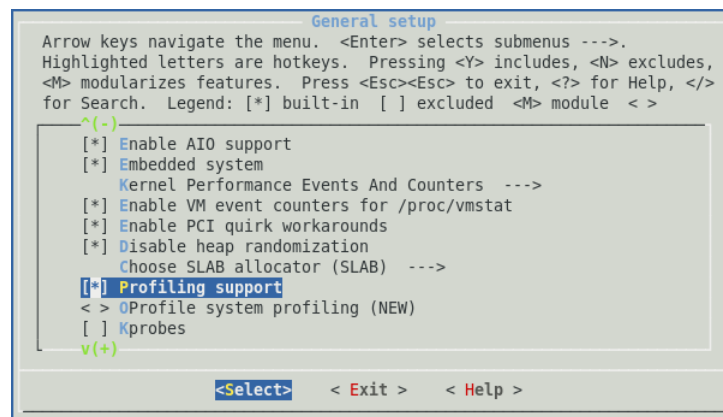


viii. Select **[\*]** Kernel performance events and counters



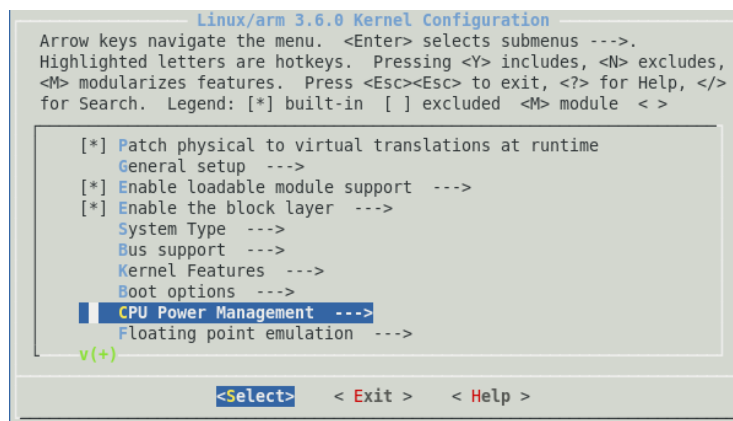
(enables CONFIG\_PERF\_EVENTS)

ix. Select **[\*]** Profiling Support



(enables CONFIG\_PROFILING)

x. Return to top level menu and select **CPU Power Management**





xi. Select **[\*]** CPU Frequency scaling

```

CPU Frequency scaling
Arrow keys navigate the menu. <Enter> selects submenus --->.
Highlighted letters are hotkeys. Pressing <Y> includes, <N> excludes,
<M> modularizes features. Press <Esc><Esc> to exit, <?> for Help, </>
for Search. Legend: [*] built-in [ ] excluded <M> module < >

[*] CPU Frequency scaling
<*> CPU frequency translation statistics
[*] CPU frequency translation statistics details
Default CPUFreq governor (userspace) --->
<*> 'performance' governor
<*> 'powersave' governor
-* 'userspace' governor for userspace frequency scaling
<*> 'ondemand' cpufreq policy governor
<*> 'conservative' cpufreq governor
ARM CPU frequency scaling drivers --->

<Select> < Exit > < Help >

```

(enables CONFIG\_CPU\_FREQ)

xii. Return to the main menu and scroll down to select **Kernel Hacking**

```

Linux/arm 3.6.0 Kernel Configuration
Arrow keys navigate the menu. <Enter> selects submenus --->.
Highlighted letters are hotkeys. Pressing <Y> includes, <N> excludes,
<M> modularizes features. Press <Esc><Esc> to exit, <?> for Help, </>
for Search. Legend: [*] built-in [ ] excluded <M> module < >

^(-)
Boot options --->
CPU Power Management --->
Floating point emulation --->
Userspace binary formats --->
Power management options --->
[*] Networking support --->
Device Drivers --->
File systems --->
Kernel hacking --->
v(+) Security options --->

<Select> < Exit > < Help >

```

### xiii. Select Tracers

```
Kernel hacking
Arrow keys navigate the menu. <Enter> selects submenus ---.
Highlighted letters are hotkeys. Pressing <Y> includes, <N> excludes,
<M> modularizes features. Press <Esc><Esc> to exit, <?> for Help, </>
for Search. Legend: [*] built-in [ ] excluded <M> module < >
^(-)
[ ] Force extended block device numbers and spread them
[ ] Force weak per-cpu definitions
[ ] Debug access to per_cpu maps
< > Linux Kernel Dump Test Tool Module
< > Notifier error injection
[ ] Fault-injection framework
[ ] Debug page memory allocations
[*] Tracers ---
[*] Enable dynamic printk() support
[ ] Enable debugging of DMA-API usage
v(+)

<Select> < Exit > < Help >
```

### xiv. Select Trace process context switches and events

```
Tracers
Arrow keys navigate the menu. <Enter> selects submenus ---.
Highlighted letters are hotkeys. Pressing <Y> includes, <N> excludes,
<M> modularizes features. Press <Esc><Esc> to exit, <?> for Help, </>
for Search. Legend: [*] built-in [ ] excluded <M> module < >
--- Tracers
[ ] Kernel Function Tracer (NEW)
[ ] Interrupts-off Latency Tracer (NEW)
[ ] Preemption-off Latency Tracer (NEW)
[ ] Scheduling Latency Tracer (NEW)
[*] Trace process context switches and events
Branch Profiling (No branch profiling) ---
[ ] Trace max stack (NEW)
[ ] Support for tracing block IO actions (NEW)
< > Ring buffer benchmark stress tester (NEW)

<Select> < Exit > < Help >
```

(enables CONFIG\_ENABLE\_DEFAULT\_TRACERS)

e. Exit and save the configuration.

f. Build the Linux kernel for the ZED target.

e.g.: **make ARCH=arm UIMAGE\_LOADADDR=0x8000 ulmage**

g. The **ulmage** is created in the **arch/arm/boot** directory.

## Appendix III – Install and Run X-server on the Host

The Xaos program runs on Linux, and sends its graphical output to an X Window. Windows hosts do not typically have servers for X Window programs installed by default, so you will need to download and install an X Server application. There is a free X Server available on SourceForge which will serve our purposes nicely.

1. Browse to the URL:

<http://sourceforge.net/projects/xming/>

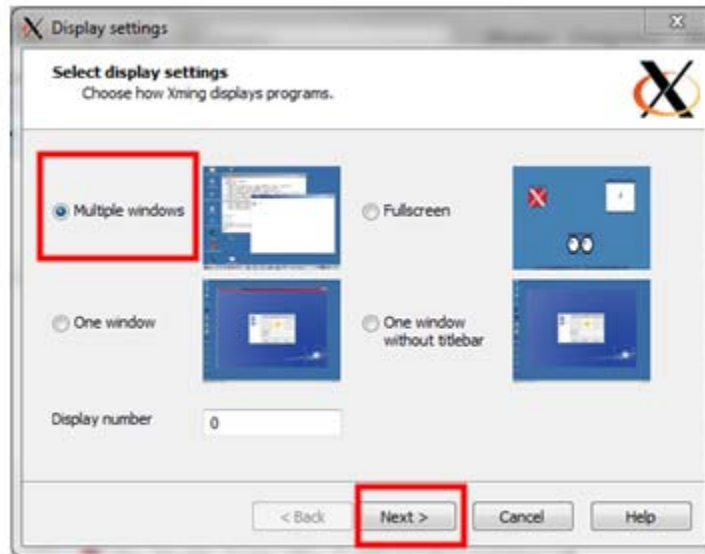
2. Click the Download button to acquire the installation package. At the time of writing for Windows 7 this was called **Xming-6-9-0-31-setup.exe**.



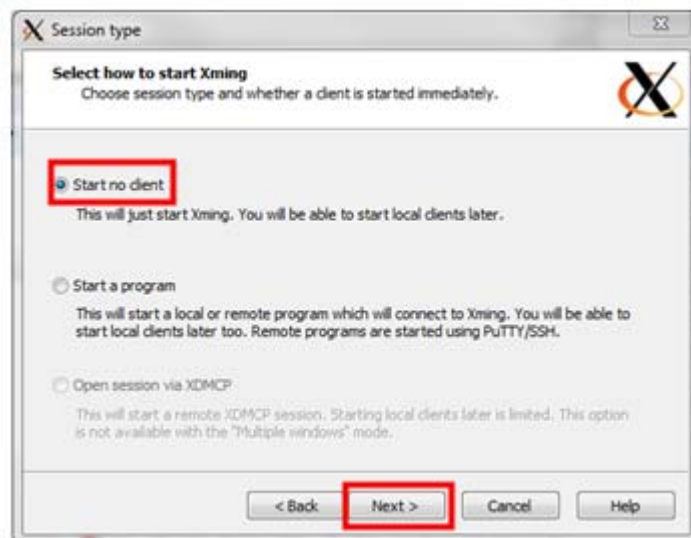
3. Save the self-extracting executable on your host system and run it after the download has completed. Click the **Next** button on the Welcome screen.



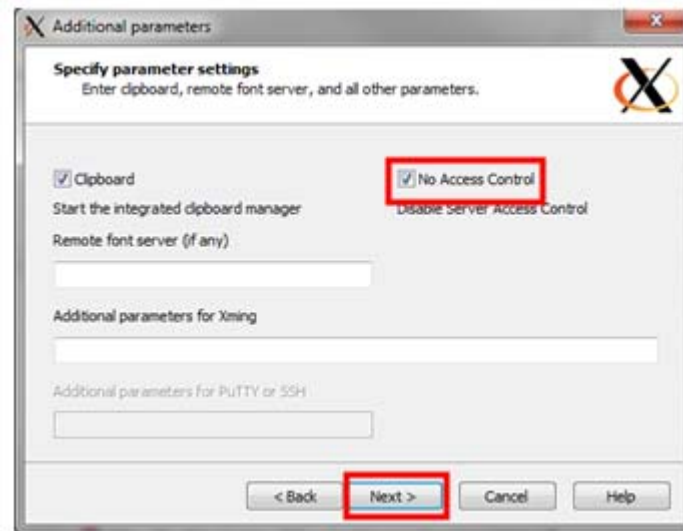
4. You may accept all the default settings on each installation screen, so click each **Next** button to advance. On the last screen, click the **Install** button.
5. When installation is complete, you can launch the application from **Start | Xming | XLaunch**.
6. In the Display settings window, select **Multiple** windows and click the Next button.



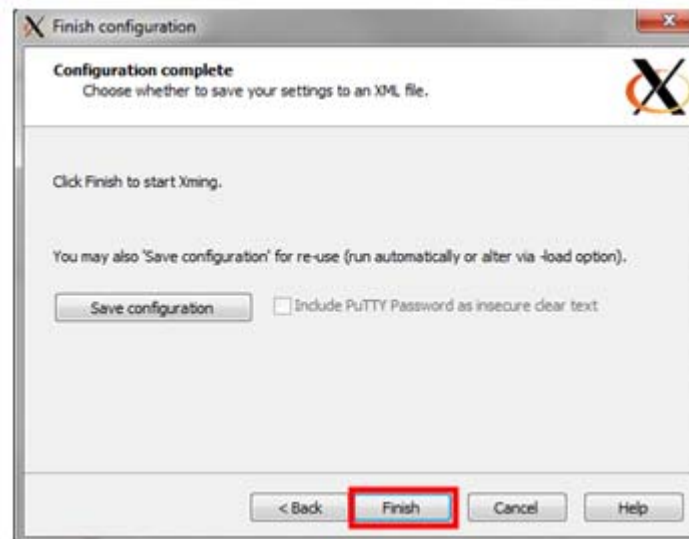
7. Select **Start no client** and click the **Next** button.



8. Select **No Access Control** and click the **Next** button.



9. You may optionally save the configuration for re-use, and click the Finish button to launch the application.



If you save the configuration, you will be prompted for a location to save a file called **config.xlaunch**. Once you have this file, you can double-click on it to start the Xming server with the same parameters next time.

When the application launches, there is no current window. The only indication you have that your X Server is running is the small X icon in the Windows task



When Xaos runs on your ZedBoard, it will communicate with your host and send graphical output to an X Server window.

## Appendix IV – Streamline Capture Options

Here is some additional information on capture options available in Streamline.

- You can select **Sampling Rate** as **Normal** (1000 samples per second), **Low** (100 samples per second) or **None** (no sampling). We'll use **Normal** which is the default.
- You can set **Buffer Mode** to **Streaming**, **Large**, **Medium** or **Small**. In **Streaming** mode the captured data is regularly streamed from the target over the network to the host PC. It may skew the performance of any network critical applications. If **Buffer Mode** is **Large** (16MB), **Medium** (4MB) or **Small** (1MB), the data will be captured to a buffer allocated on the target. The profile session will be stopped when the buffer is filled up. Then data will then be passed to the host in one go. By default buffer mode is set to **Streaming**.
- You can use the **Duration** field to a maximum length of time to capture data in seconds or *minutes:seconds*. By default Streamline will capture data until you click the **Stop** button which we'll see later.
- If **Call Stack Unwinding** is checked and the program has been compiled with frame pointers in ARM state then Streamline will be able to capture information about call paths, call graphs and stack usage. Xaos is built this way, so leave **Call Stack Unwinding** checked.

⇒ If you are using an ARM Energy Probe, in the **Energy Capture** section you should choose **ARM Energy Probe**. Once the **ARM Energy Probe** is selected, the **Tool Path** will display the path for the **caiman** application if the DS-5 was installed in the default directory. (If DS-5 is installed in the non-default directory then, manually add the path to the **caiman** application. The **caiman** application can be found in the `... \DS-5\bin` directory.)

During a capture, Streamline will collect power information from the target using the energy probe and then display the power charts in the report. There are three channels available on energy probe and each channel gives us the information about the power consumption, voltage and current. Check the channels you have connected and if you wish to analyze the **Voltage** and/or **Current** then those need be checked as shown in the above figure. Power analysis is discussed in more detail later. See [ARM Energy Probe setup](#) in the appendix on page 99 for setup information.

- The **Analysis** section lets you specify whether Streamline should **Process Debug Information**. This will allow you to use Streamline to view source code. The **High Resolution Timeline** option enables you to zoom in more levels in the timeline (for example context switching). These checkboxes only control the defaults for the first report; they do not change the captured data.

## Appendix V – Building Gator

The root file system supplied in the support folder for this tutorial already has the gator elements installed. However, if you wish to build gator from the source on your own, follow these instructions.

1. Locate the gator source folder on your host system, which was installed in your DS-5 tools folder. If you used the default installation path, this will be at:

**C:\Program Files (x86)\DS-5\arm\gator**

2. Copy the **gator** folder to a directory of your choice on your Linux build system. If you have been following the directory structure used for the this tutorial series, the recommended location is training home folder on the Virtual Machine:

**/home/training/**

This will place the source folders at the following locations on your Linux build system:

Gator daemon:                **~/gator/daemon-src/**  
Gator kernel module:        **~/gator/driver-src/**

3. To create the gator daemon, on your Linux build system make the current directory the daemon source folder.

**cd ~/gator/daemon-src/**

4. Unpack the gator daemon source code:

**tar -zxvf gator-daemon.tar.gz**

5. Change to the unpacked source sub-folder:

**cd gator-daemon**

6. Issue the make command and gatord will be created using the CodeSourcery cross toolchain, which is the tool suite used to build the kernel from the Avnet Linux Speedway. (If you built the daemon previously, clean the directory first)

**(make clean)**  
**make**

7. To create the gator kernel module, change to the source directory for the driver module:

```
cd ~/gator/driver-src/
```

8. Unpack the gator driver source code:

```
tar -zxvf gator-driver.tar.gz
```

9. Change to the unpacked source sub-folder:

```
cd gator-driver
```

10. Build the kernel module using the make command in the following format:

```
make -C kernel_build_dir M=`pwd` ARCH=arm CROSS_COMPILE=<...> modules
```

Note that it is important that the “back-apostrophe” is used around pwd, and not the regular single apostrophe.

11. If your Linux system is in directory ~/linux-xlnx/, the kernel module (gator.ko) can be created with this command. . (If you built the driver previously, clean the directory first)

```
(make clean)
```

```
make -C ~/linux-xlnx/ M=`pwd` ARCH=arm modules
```

12. Copy the compiled files to your SD card.

```
~/gator/daemon-src/gator-daemon/gatord  
~/gator/driver-src/gator-driver/gator.ko
```

If you would like to pack gator into your RAM disk image, follow these additional steps:

1. Change to your home directory.

```
cd ~
```

2. Uncompress the existing RAM disk image.

```
gunzip ramdisk32M.image.gz (creates ramdisk32M.image)
```



3. Mount the disk image at a new mount point called ramdisk.

```
sudo mount -o loop ramdisk32M.image ramdisk
```

4. Copy the gator files to the RAM disk /root/ folder.

```
sudo cp ~/gator/driver-src/gator-driver/gator.ko ramdisk/root/
```

```
sudo cp ~/gator/daemon-src/gator-daemon/gatord ramdisk/root/
```

5. Use gedit to modify the startup script to automatically run the gator daemon when the kernel is booted.

```
sudo gedit ramdisk/etc/init.d/rcS
```

6. Add the following lines into the rcS file immediately before **echo "rcS Complete"**.

```
echo "++ Starting gator daemon"  
/root/gatord &
```

**Save** the file and exit gedit.

7. Unmount the RAM disk image from the mount point

```
sudo umount ramdisk/
```

8. Compress the RAM disk image.

```
gzip -v9 ramdisk32M.image
```

9. Run the mkImage tool to convert the file format to a compatible form for use with the ulmage kernel.

```
mkimage -A ARM -n 'RAM disk 32MB' -T ramdisk -C gzip -d ramdisk32M.image.gz  
uramdisk.image.gz2
```

Copy **uramdisk.image.gz** to the SD card.

---

<sup>2</sup> If you need to undo the results of the mkimage command, you can recover the original input file by removing the first 64 bytes: **dd if=<image> of=<recovered file> bs=64 skip=1**

## Revision History

Date	Version	Revision
7 Jun 13	00	Initial Draft
24 Jun 13	01	Added xconfig.launch info
20 Sep 13	02	Release

## Resources

<http://www.zedboard.org>

<http://www.xilinx.com/zyng>

<http://www.arm.com/products/tools/software-tools/ds-5/index.php>