Profiling and Performance Tuning

Introduction

This lab guides you through the process of profiling an application and analyzing the output. The application is then accelerated in hardware and profiled again to analyze the performance improvement.

Objectives

After completing this lab, you will be able to:

- Setup the board support package (BSP) for profiling an application
- Set the necessary compiler directive on an application to enable profiling
- Setup the profiling parameters
- Profile an application and analyze the output

Procedure

This lab is separated into steps that consist of general overview statements that provide information on the detailed instructions that follow. Follow these detailed instructions to progress through the lab.

Design Description

In this lab, you will design an embedded system consists of ARM Cortex-A9 processor SoC and two instances of the provided FIR filter IP. The following diagram represents the completed design (**Figure 1**).

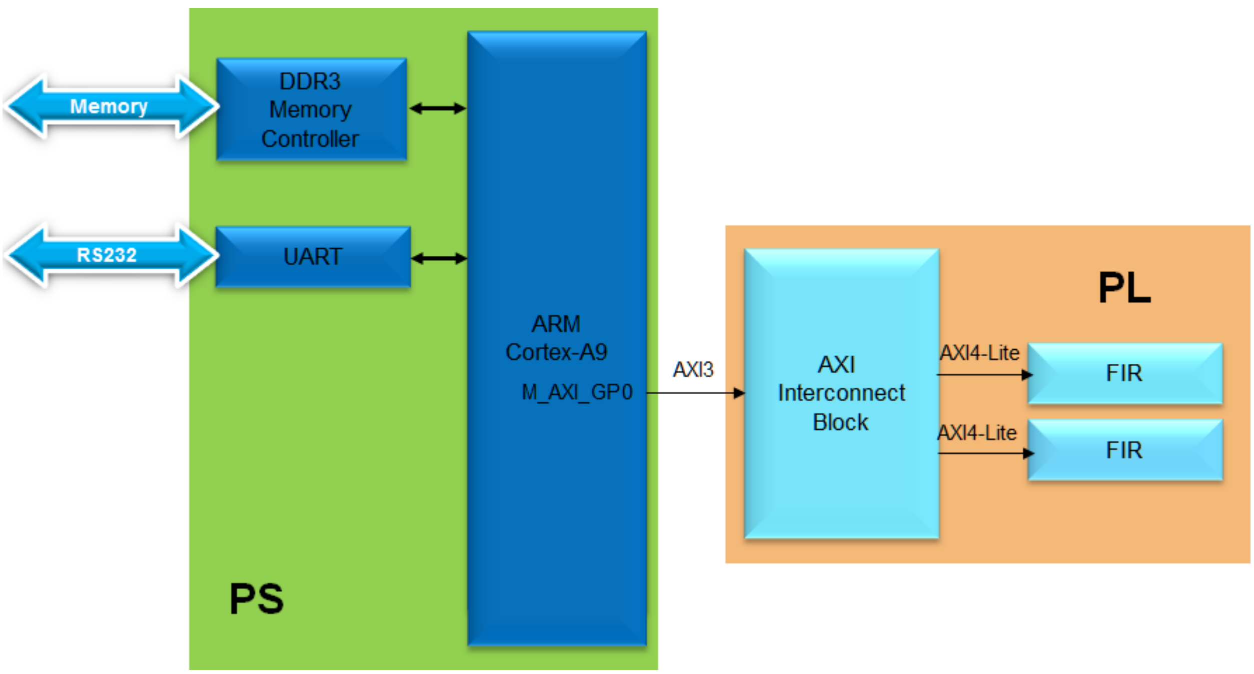
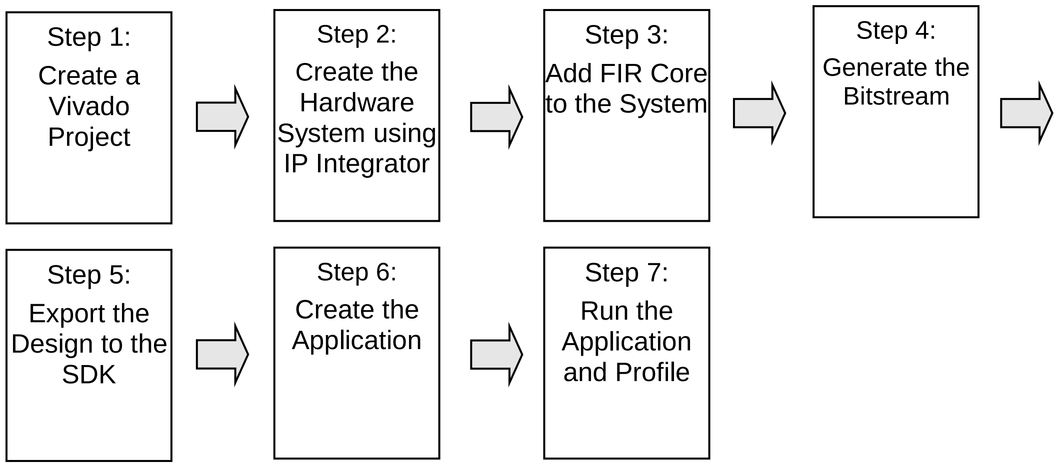


Figure 1. Completed Design

General Flow for this Lab



In the instructions below;
{***sources***} refers to: C:\Xilinx_trn\Zynq_adv\lab_sources
{***labs***} refers to : C:\Xilinx_trn\Zynq_adv

Create a Vivado Project Step 1

1-1. Launch Vivado and create an empty project, called lab6, targeting the ZedBoard and using the Verilog language.

1-1-1. Open Vivado and create a new project new project call *lab6* in the **C:\Xilinx_trn\Zynq_adv** directory.

1-1-2. Select the **RTL Project** option in the *Project Type* form, and click **Next**.

1-1-3. Select **Verilog** as the *Target Language* in the *Add Sources* form, and click **Next**.

1-1-4. Click **Next**.

1-1-5. In the *Default Part* form, click on *Boards* and select the Zedboard and click **Next**.

1-1-6. Click **Finish** to create an empty Vivado project.

1-2. Set the project settings to include provided fir_top IP

1-2-1. Click **Settings** in the *Flow Navigator* pane.

1-2-2. Expand **IP** in the left pane of the *Project Settings* form.

1-2-3. Click Repository and using “minus” button remove entries, if any.

1-2-4. Click on the “plus” button, browse to **C:\Xilinx_trn\Zynq_adv\lab_sources\lab6** and click **Select**.

1-2-5. Click **OK**.

The directory will be scanned and it will report one IP was detected.

1-2-6. Click **OK**.

Creating the Hardware System Using IP Integrator Step 2

2-1. Create a block design in the Vivado project using IP Integrator to generate the Zynq ARM Cortex-A9 processor based hardware system.

2-1-1. In the Flow Navigator, click **Create Block Design** under IP Integrator.

2-1-2. Name the block **system** and click **OK**.

2-1-3. Click on the + button.

2-1-4. Once the IP Catalog is open, type zy into the Search bar, and double click on **ZYNQ7 Processing System** entry to add it to the design.

2-1-5. Click *Run Block Automation*, and click **OK** to accept the default settings.

2-1-6. Double click on the Zynq block to open the *Customization* window for the Zynq processing system.

A block diagram of the Zynq should now be open, showing various configurable blocks of the Processing System.

2-2. Configure the I/O Peripherals block to only have UART 1 support. Deselect the TTC device.

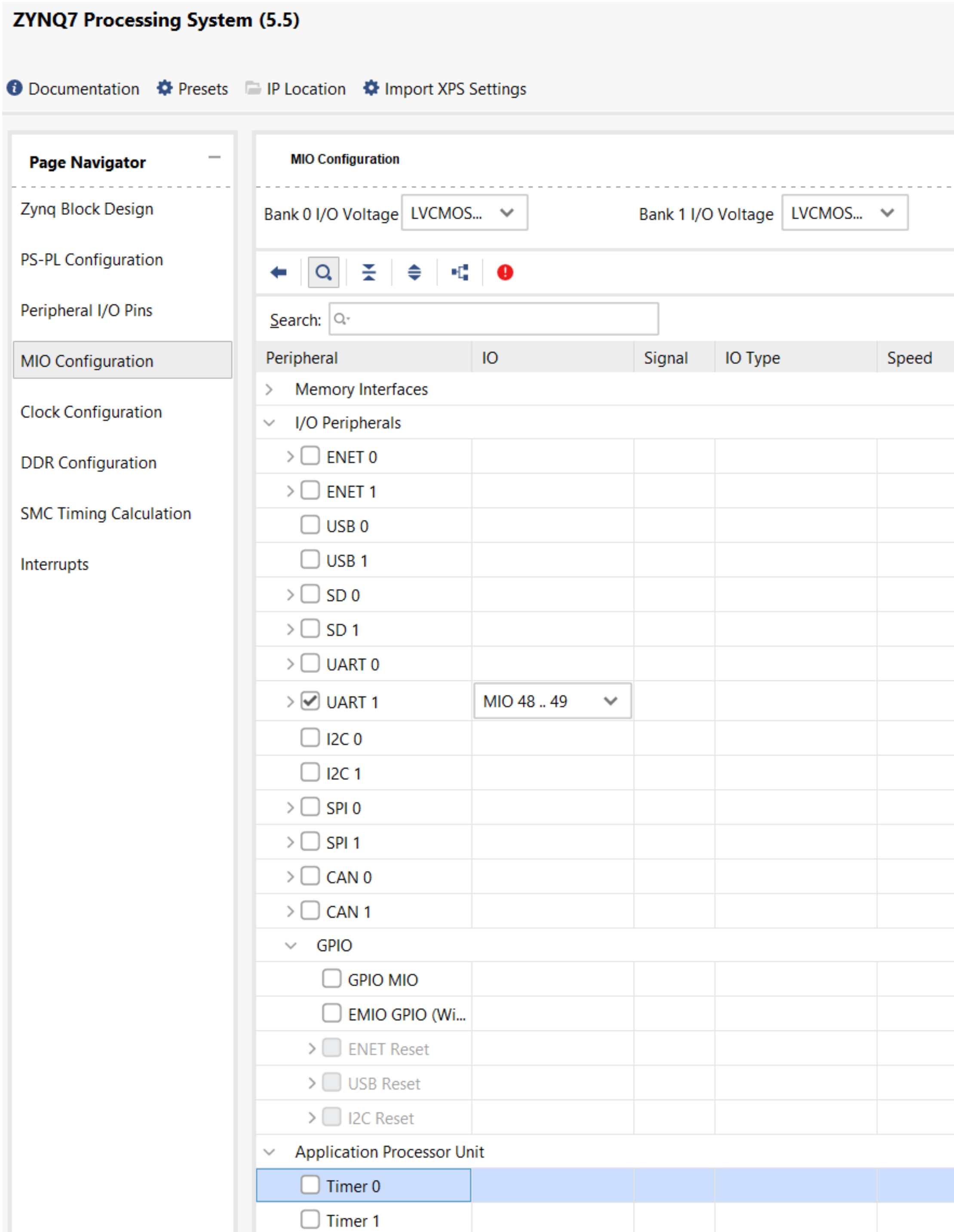
2-2-1. Click on the *MIO Configuration* panel to open its configuration form.

2-2-2. Expand the *I/O Peripherals* on the right.

2-2-3. Uncheck *ENET 0*, *USB 0*, *SD 0*, *GPIO (GPIO MIO)*, leaving *UART 1* selected.

2-2-4. Expand the *I/O Peripherals > GPIO* and uncheck *GPIO MIO (USB Reset*, *I2C Reset* will be unchecked automatically).

2-2-5. In the **MIO Configuration** panel, expand the **Application Processing Unit** and uncheck the **Timer 0**.



2-2-6. Click **OK**.

2-2-7. You should see the following Diagram.

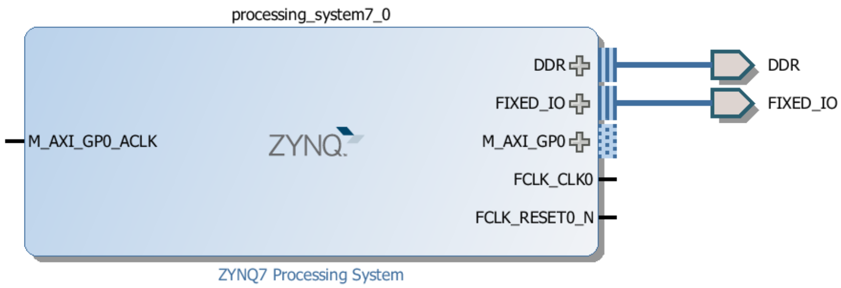


Figure 2. ZYNQ Processing System configured block

Add FIR Core to the System Step 3

3-1. Instantiate the provided FIR core twice naming the instances as fir_left and fir_right. Validate the design.

3-1-1. Click the + button and search for **fir** in the catalog.

3-1-2. Double-click on the **fir_top_v1_0** to add the IP instance to the system

3-1-3. Select the *fir_top_1* instance and change its name to **fir_left** in its property form.

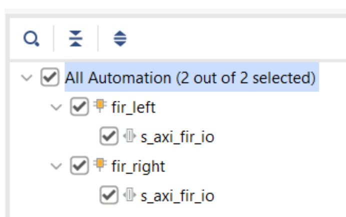
3-1-4. Click the + button and search for **fir** in the catalog.

3-1-5. Double-click on the **fir_top_v1_0** to add the IP instance to the system

3-1-6. Select the *fir_top_1* instance and change its name to **fir_right** in its property form.

3-1-7. Click on **Run Connection Automation**, and select **All Automation** to select *fir_left* and *fir_right*.

s_axi_fir_io for both *fir_left* and *fir_right* will be automatically connected to the Zynq *M_AXI_GP0* port



3-1-8. Click **OK** to connect the two blocks to the *M_AXI_GP0* interface.

3-1-9. Click on redraw button .

The design should look similar to shown below:

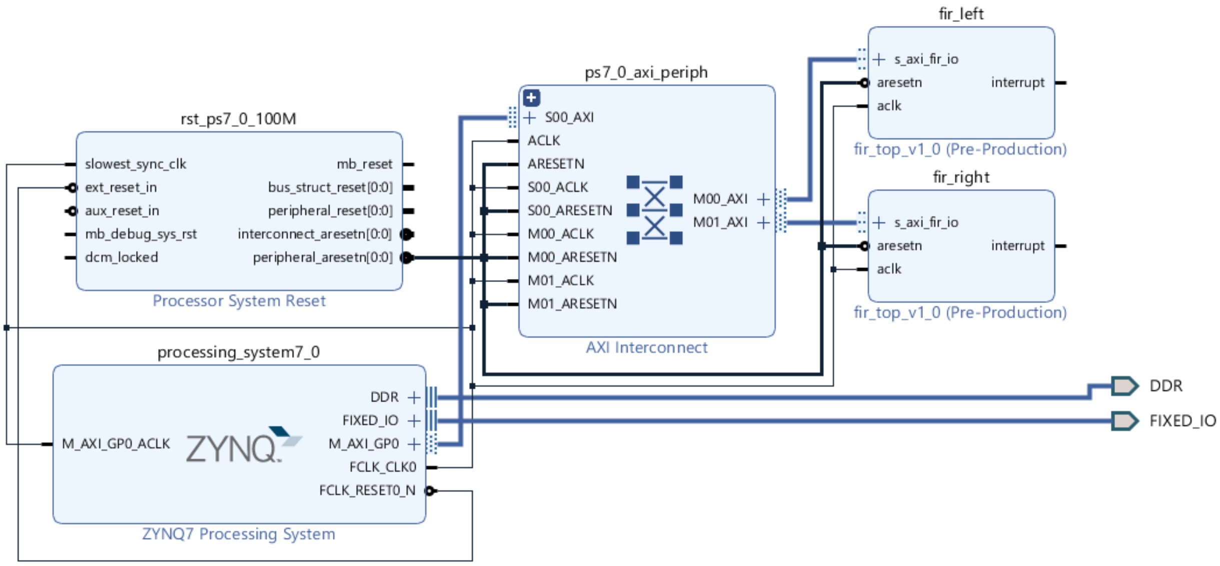


Figure 3. The completed design

It is not necessary to connect the *interrupt* signals of the *fir* blocks.

3-1-10. Click on the (Validate Design) button to make sure that there are no errors.

Ignore warnings.

Generate the Bitstream Step 4

4-1. Create the top-level HDL of the embedded system, and generate the bitstream.

4-1-1. In Vivado, select the *Sources* tab, expand the *Design Sources,* right-click the *system.bd* and select **Create HDL Wrapper** and click **OK**.

4-1-2. Click on the **Generate Bitstream** in the *Flow Navigator* pane to synthesize and implement the design, and generate the bitstream.

4-1-3. Click **Save** to save the design and **Yes** to run the necessary processes. Click **OK** to launch the runs.

4-1-4. When the bitstream generation process has completed click **Cancel.**

Export the Design to the SDK Step 5

5-1. Export the design to the SDK, create the software BSP using the standalone operating system and enable the profiling options.

5-1-1. Export the hardware configuration by clicking **File > Export > Export Hardware…**

5-1-2. Tick the box to *Include Bitstream*, and click **OK**

5-1-3. Launch SDK by clicking **File > Launch SDK** and click **OK**

5-1-4. In SDK, select **File** > **New** > **Board Support Package.**

5-1-5. Notice **Standalone_bsp_0** in the **Project name** field and click **Finish** with default settings.

A Board Support Package Settings window will appear.

5-1-6. Select the **Overview > standalone** entry in the left pane, click on the drop-down arrow of the *enable_sw_intrusive_profiling Value* field and select **true**.

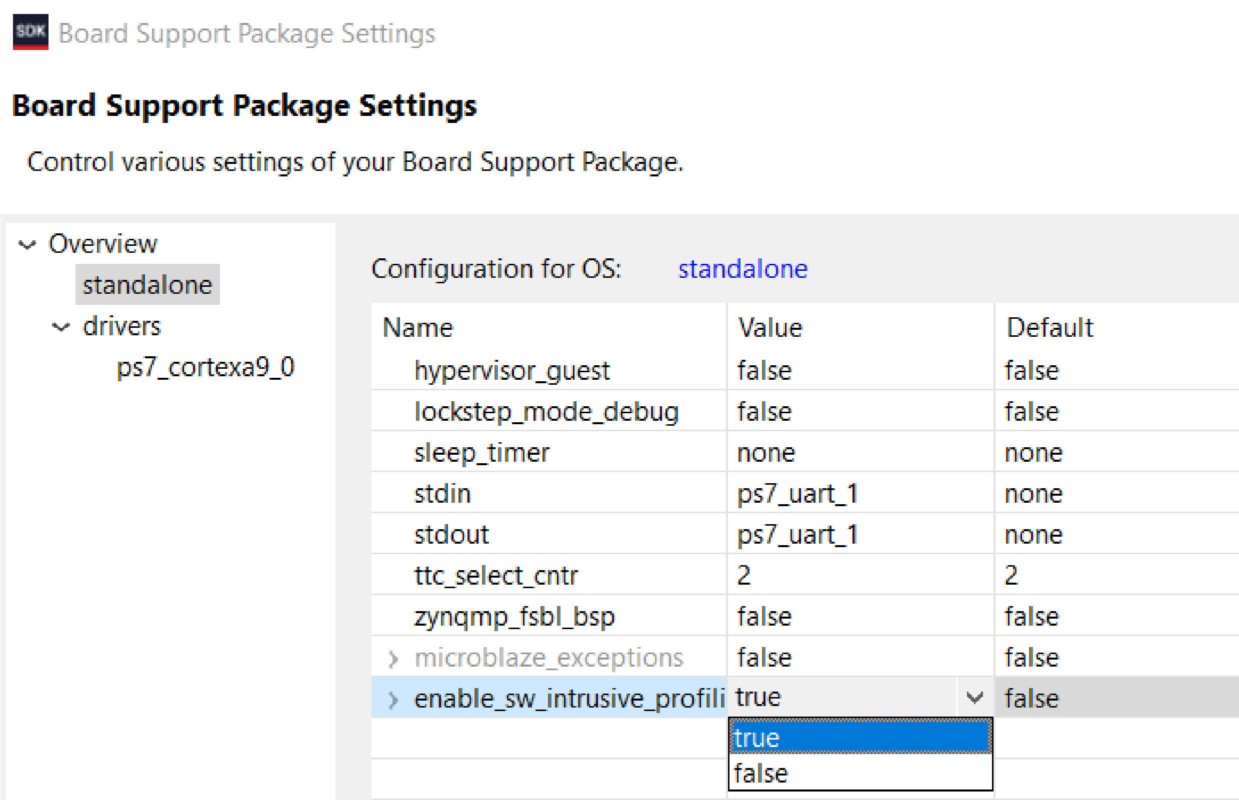


Figure 4. Enable profiling in the board support package

5-1-7. Select the **Overview > drivers > ps7_cortexa9_0** and add **–pg** in the *extra_compiler_flags Value* field.

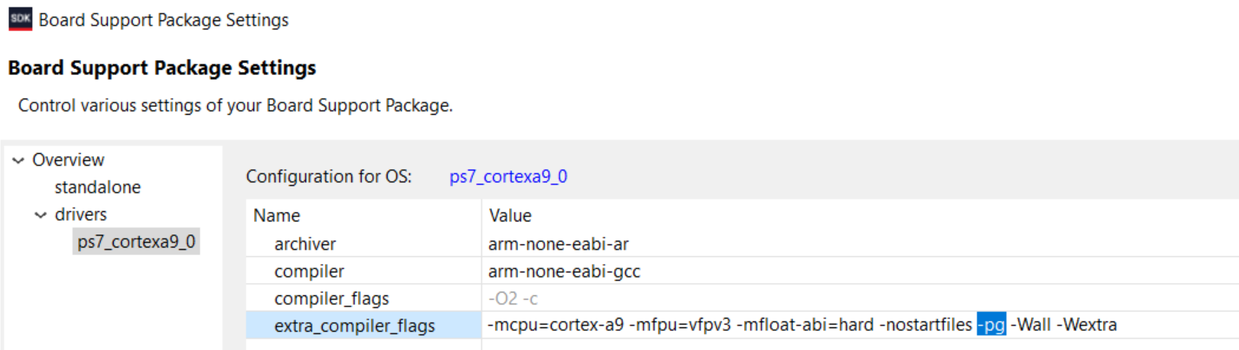


Figure 5. Adding profiling switch

5-1-8. Click **OK** to accept the settings and create the BSP.

Create the Application Step 6

6-1. Create the *lab6* application using the provided lab6.c, fir.c, fir.h, fir_coef.dat, and xfir_fir_io.h files.

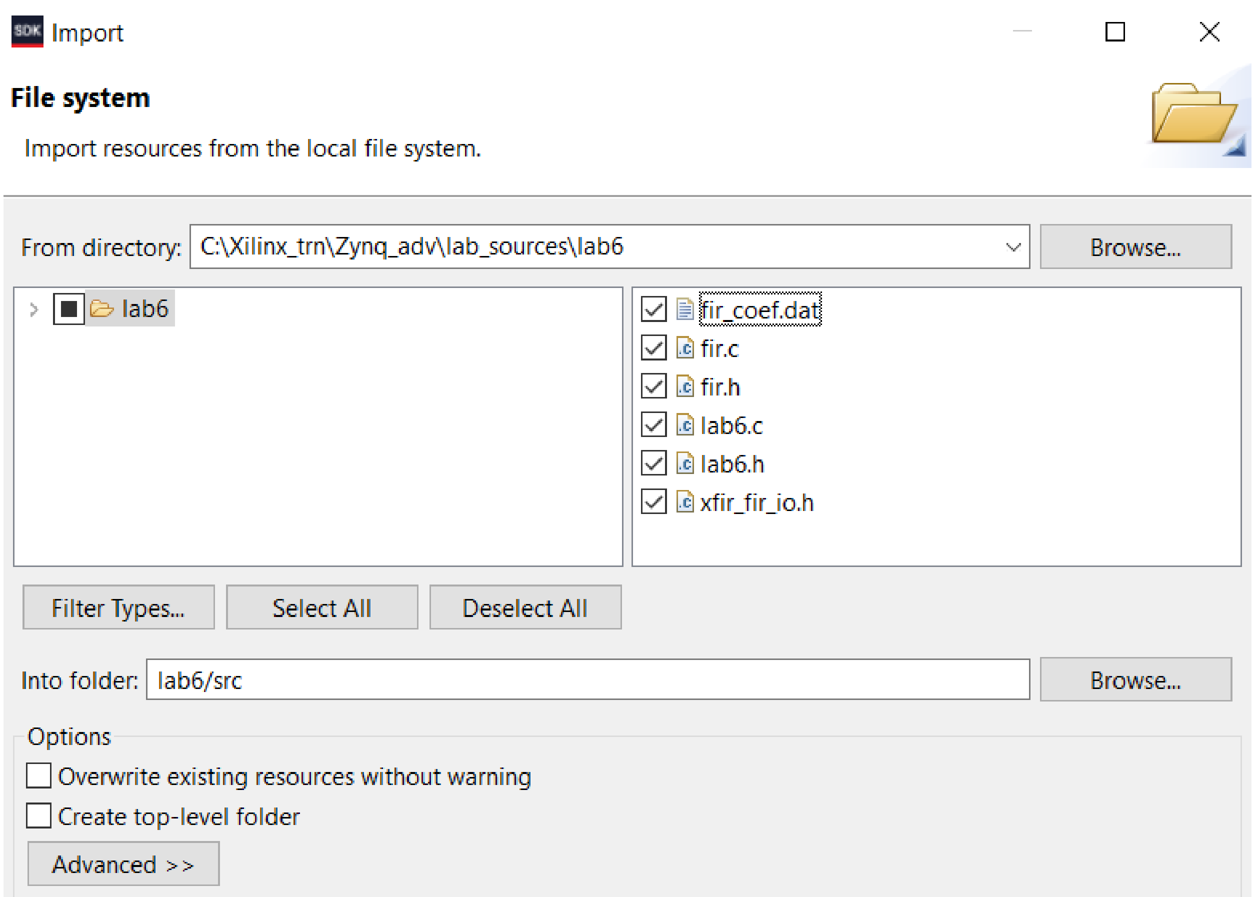
6-1-1. Select **File** > **New** > **Application Project.**

6-1-2. Enter **lab6** as the project name, select the **Use existing** *standalone_bsp_0* option, and click **Next**.

6-1-3. Select **Empty Application** in the *Available Templates* pane and click **Finish**.

6-1-4. In the *lab6* project, right click on the *src* directory and select **Import.**

6-1-5. Expand the *General* folder and double-click on **File system,** and browse to the **C:\Xilinx_trn\Zynq_adv\lab_sources\lab6** directory.



6-1-6. Select **fir_coef.dat, fir.c, fir.h, lab6.c, lab6.h, xfir_fir_io.h,** and click **Finish.**

The program should compile successfully and generate the lab6.elf file.

6-1-7. Open the *lab6.c* file and scroll to the main function at the bottom. Notice the following code:

```
#ifdef SW_PROFILE
    fir_software(&output,signal);
#else
    filter_hw_accel_input(&output,signal);
#endif
```

The function *fir_software*( ) function is a software implementation of the FIR function. The *filter_hw_accel_input*( ) function offloads the FIR function to the two FIR blocks that have been implemented in the PL.

Run the Application and Profile Step 7

7-1. Place the board into the JTAG boot up mode. Program the PL section and run the application using the user defined SW_PROFILE symbol.

7-1-1. Place the board in the JTAG boot up mode.

7-1-2. Power ON the board.

7-1-3. Select **Xilinx > Program FPGA** and click on **Program**.

7-1-4. Right click on the *lab6* folder, and select **C/C++ Build Settings**.

7-1-5. Under the **ARM v7 gcc compiler** group, select the **Symbols** sub-group**,** click on the button to open the value entry form, enter **SW_PROFILE.**

This will allow us to profile the software loop of the FIR application.

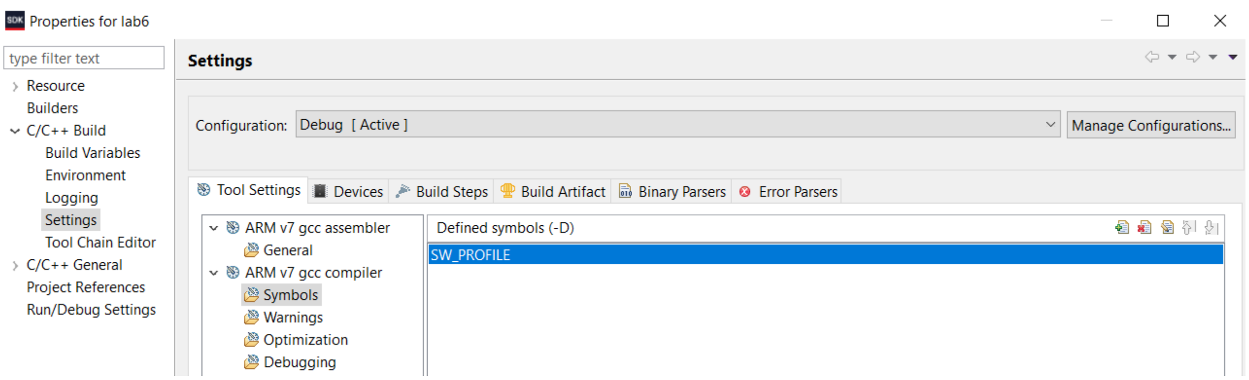


Figure 6. Add user-defined symbol

7-1-6. Click **Apply**.

7-1-7. Under the **ARM v7 gcc compiler** group, select the **Profiling** sub-group, then check the **Enable Profiling** box.

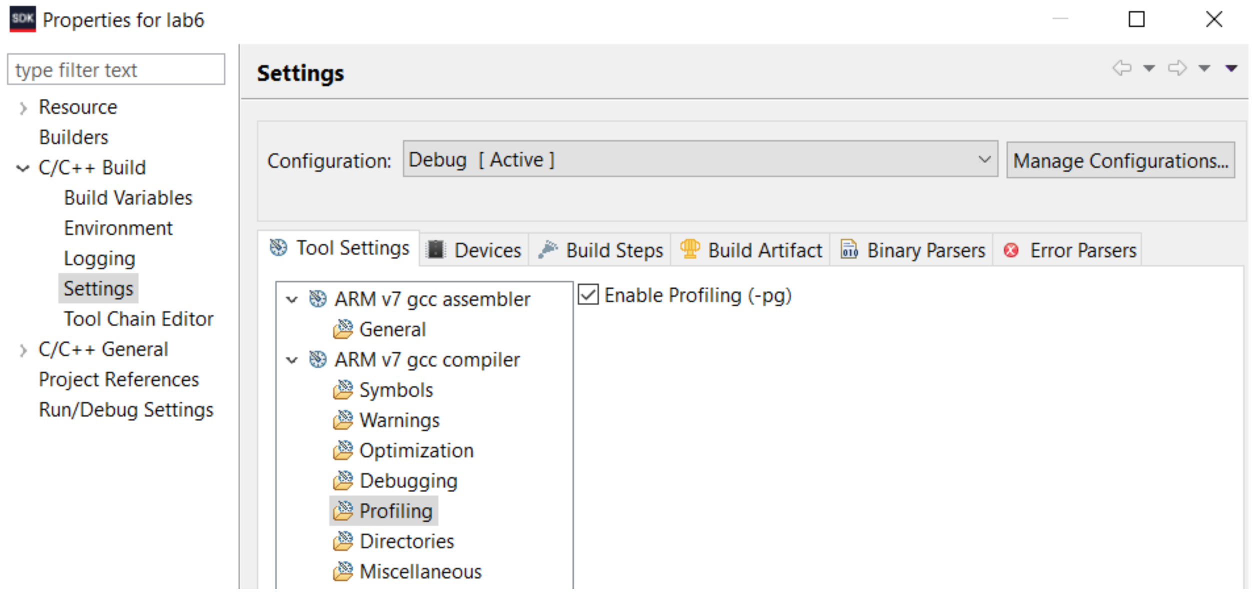


Figure 7. Compiler setting for enabling profiling

7-1-8. Click **OK**.

7-1-9. From the menu bar, Select **Run > Run Configurations…** and double click on *Xilinx C/C++ application (System Debugger)* to create a new configuration.

7-1-10. Click on the newly created configuration, and select the **Application** tab.

7-1-11. Click on the *Advance Options* **Edit…** button.

7-1-12. Click on the *Enable Profiling (gprof)* check box, enter **100000** (100 kHz) in the Sampling Frequency field, enter **0x10000000** in the scratch memory address field.

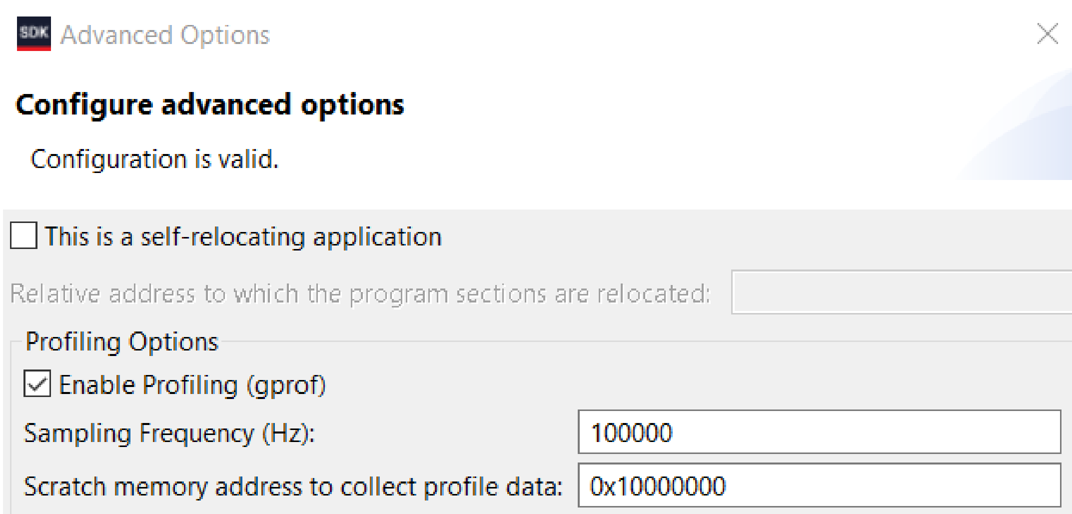


Figure 8. Profiling options

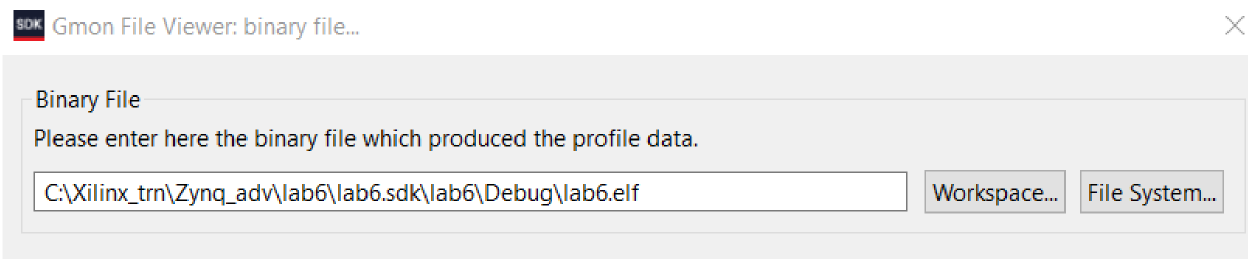
7-1-13. Click **OK**.

7-1-14. Click the **Run** button to download the application and execute it.

The program will run.

7-2. Analyze the results.

7-2-1. When execution is completed, the Gmon File Viewer dialog box will appear showing *lab6.elf* as the corresponding binary file.

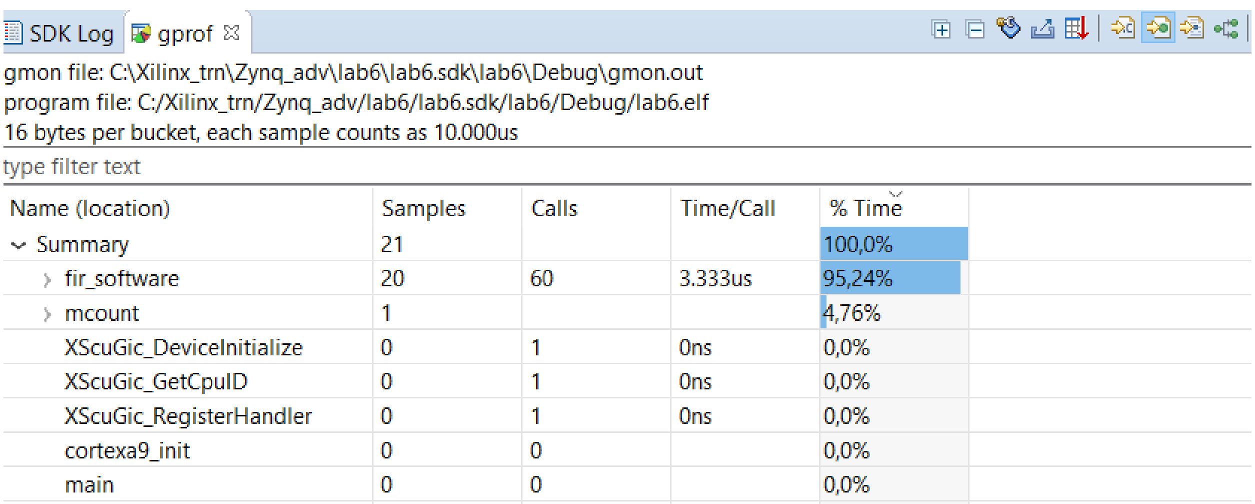


7-2-2. Click **OK**.

7-2-3. Click on the **Sort samples per function** button ().

7-2-4. Click in the **%Time** column to sort in the descending order.

Note that the fir_software routine is called 60 times, 20 samples were taken during the profiling, and on an average of 3.333 (ZedBoard) microseconds were spent per call.



SDK Log | gprof

gmon file: C:\Xilinx_trn\Zynq_adv\lab6\lab6.sdk\lab6\Debug\gmon.out
program file: C:/Xilinx_trn/Zynq_adv/lab6/lab6.sdk/lab6/Debug/lab6.elf
16 bytes per bucket, each sample counts as 10.000us

type filter text

Name (location)	Samples	Calls	Time/Call	% Time
Summary	21			100,0%
fir_software	20	60	3.333us	95,24%
mcount	1			4,76%
XScuGic_DeviceInitialize	0	1	0ns	0,0%
XScuGic_GetCpuID	0	1	0ns	0,0%
XScuGic_RegisterHandler	0	1	0ns	0,0%
cortexa9_init	0	0		0,0%
main	0	0		0,0%

Figure 9. Sorting results

7-2-5. Go back to the *Run Configuration*, and change the sampling frequency to **1000000** (1 MHz) and profile the application again.

7-2-6. When execution is completed, click **OK** and the gprof viewer will be updated.

7-2-7. Invoke **gprof**, select the **Sorts samples per function** output, and sort the **%Time** column.

Notice that the output has better resolution and reports more functions and more samples per function calls. Note that the number of calls to the fir_software function has not changed but the number of samples taken increased, and the average time spent per call is 5.000 microseconds in the figure below.

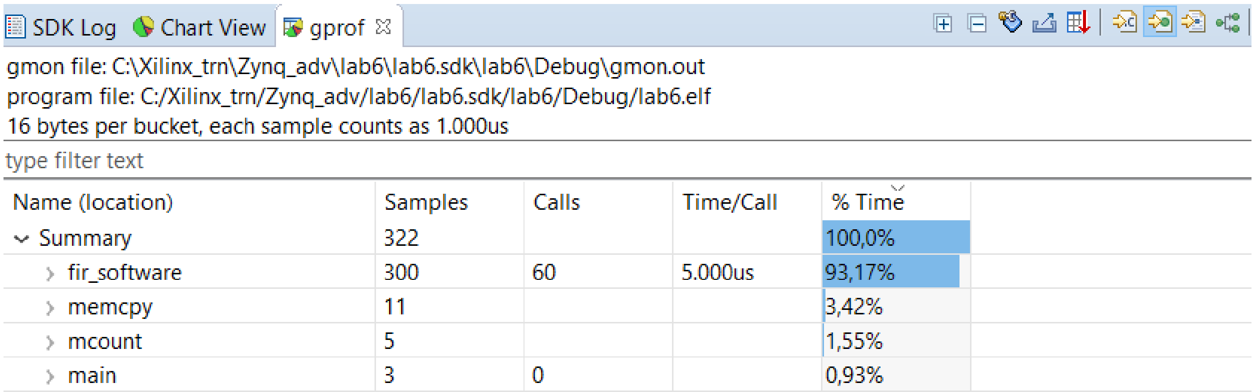


Figure 10. Profiled results with 1 MHz sampling frequency

At this stage, the designer of the system would decide if the FIR function should be ported to hardware.

7-3. Profile the application using the hardware FIR filter IP by removing the user defined SW_PROFILE symbol.

7-3-1. Select the *lab6* application, right-click, and select **C/C++ Build Settings**.

7-3-2. Under the **ARM v7 gcc compiler** group, select the **Symbols** sub-group**,** select **SW_PROFILE**, and delete it by clicking on the delete button.

This will allow us to profile the hardware IP of the FIR application.

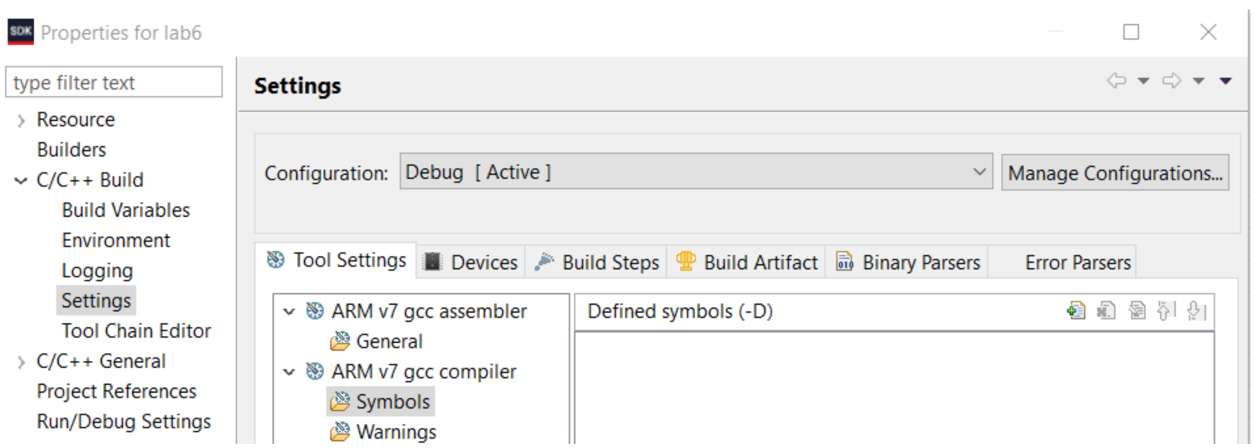


Figure 11. Deleting the user-defined symbol

7-3-3. Click **Apply**, and then click **OK**

7-3-4. Select **Run > Run Configurations** and click the **Run** button to profile the application again and click **OK** when profiling completes.

Notice that the output now shows filter_hw_accel_input function call instead of the fir_software function call. Note that the average time spent per call is much less as the filtering is done in the hardware instead of the software.

7-3-5. Close the SDK and Vivado programs by selecting **File > Exit** in each program.

7-3-6. Turn OFF the power on the board.

Conclusion

This lab led you through enabling the software BSP and the application settings for the profiling. You went through creating the hardware which included the hardware IP and was later profiled in the application. You analyzed the profiled application output.