Profiling and Performance Tuning

This demo 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.

In this demo, we will:

* 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

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

A diagram of a computer

Description automatically generated

Figure 1. Completed Design

1. Create a Vivado Project Step
   1. Launch Vivado and create an empty project, called demoProfile, targeting the Zybo or ZedBoard Zynq Evaluation and Development Kit and using the VHDL language.
      1. Open Vivado and create a new project new project call *demoProfile*.
      2. Select the **RTL Project** option in the *Project Type* form, and click **Next**.
      3. Select **VHDL** as the *Target Language* in the *Add Sources* form, and click **Next**.
      4. Click **Next** two times.
      5. In the *Default Part* form, click on *Boards* and select either the Zybo or Zedboard and click **Next**.
      6. Click **Finish** to create an empty Vivado project.
   2. Set the project settings to include provided fir\_top IP
      1. Click **Settings** in the *Flow Navigator* pane.
      2. Expand **IP** in the left pane of the *Project Settings* form.
      3. Click Repository and using “minus” button remove entries, if any.
      4. Click on the “plus” button, browse to the directory where you placed the downloaded IP that accompanies this demo and click **Select**.
      5. Click **OK**. The directory will be scanned and it will report one IP was detected.
      6. Click **OK** twice.
2. Creating the Hardware System Using IP Integrator Step 2
   1. Create a block design in the Vivado project using IP Integrator to generate the Zynq ARM Cortex-A9 processor-based hardware system.
      1. In the Flow Navigator, click **Create Block Design** under IP Integrator.
      2. Name the block **system** and click **OK**.
      3. Click on the  button.
      4. Once the IP Catalog is open, type zy into the Search bar, and double click on **ZYNQ Processing System** entry to add it to the design.
      5. Click *Run Block Automation*,and click **OK** to accept the default settings.
      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.

* 1. Configure the I/O Peripherals block to only have UART 1 support. Deselect the TTC device.
     1. Click on the *MIO Configuration* panel to open its configuration form.
     2. Expand the *I/O Peripherals* on the right.
     3. Uncheck *ENET 0*, *USB 0*, and *SD 0*, *GPIO (GPIO MIO)*, leaving *UART 1* selected.
     4. In the **MIO Configuration** panel, expand the **Application Processing Unit** and uncheck the **Timer 0**.
     5. Click **OK**.

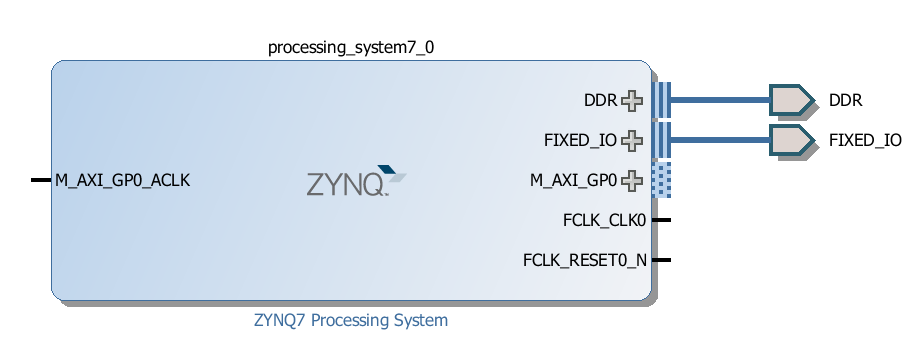


Figure 2. ZYNQ Processing System configured block

1. Add FIR Core to the System Step 3
   1. Instantiate the provided FIR core twice naming the instances as fir\_left and fir\_right. Validate the design.
      1. Click the  button and search for **fir** in the catalog.
      2. Double-click on the **fir\_top\_v1\_0** to add the IP instance to the system
      3. Select the *fir\_top\_1* instance and change its name to **fir\_left** in its property form.
      4. Click the  button and search for **fir** in the catalog.
      5. Double-click on the **fir\_top\_v1\_0** to add the IP instance to the system
      6. Select the *fir\_top\_1* instance and change its name to **fir\_right** in its property form.
      7. Click on **Run Connection Automation**, and select **All Automation** to select *fir\_left* and *fir\_right***.**
      8. Click on *s\_axi\_fir\_io* for both *fir\_left* and *fir\_right* and confirm that they will be automatically connected to the Zynq *M\_AXI\_GP0* port
      9. Click **OK** toconnect the two blocks to the *M\_AXI\_GP0*  interface.

The design should look similar to shown below:

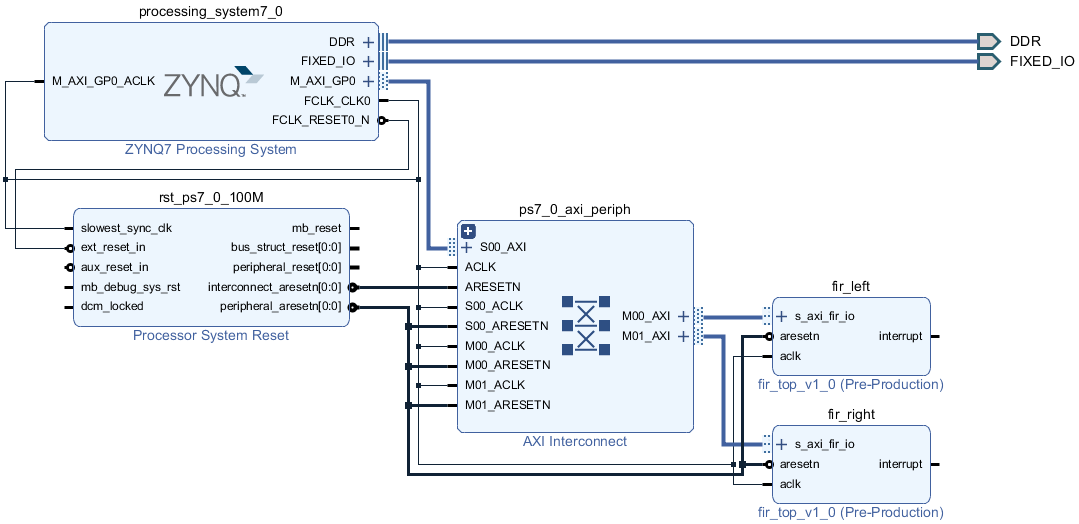


Figure 3. The completed design

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

* + 1. Select the *Diagram* tab, and click on the  (Validate Design) button to make sure that there are no errors.

Ignore warnings.

1. Generate the Bitstream Step 4
   1. Create the top-level HDL of the embedded system, and generate the bitstream.
      1. In Vivado, select the *Sources* tab, expand the *Design Sources,* right-click the *system.bd* and select **Create HDL Wrapper** and click **OK**.
      2. Click on the **Generate Bitstream** in the *Flow Navigator* pane to synthesize and implement the design, and generate the bitstream.
      3. Click **Save** to save the design and **Yes** to run the necessary processes. Click **OK** to launch the runs.
      4. When the bitstream generation process has completed click **Cancel.**
2. Export the Design to Vitis Step 5
   1. Export the design to Vitis, create the software BSP using the standalone operating system and enable the profiling options.
      1. Export the hardware configuration by clicking **File > Export > Export Hardware…**
      2. Tick the box to *Include Bitstream*, and click **OK**
      3. Launch Vitis.
   2. In Vitis, create an empty application project named demoProfile, and import the provided demoProfile.c file.
      1. Create a new Workspace by entering **workspace\_demoProfile**.
      2. Select **Create Application Project.**
      3. In the *Project Name* field, enter demoProfile as the project name, leave all other settings to their default’s and click **Next.**
      4. For the Platform, select the **Create a new platform from Hardware (XSA)** tab. Then click the plus sign and navigate to the directory containing your Vivado project. Select **design\_1\_wrapper.xsa**. This is the export of your hardware description. Then click **Next**.
      5. For the Domain, leave the CPU, OS, Language and boot components as defaulted, click **Next**.
      6. Select the **Empty Application** template and click **Finish.**
      7. Click on **Navigate to BSP settings** to change the settings in the board support package.

A screenshot of a computer

Description automatically generated

Figure 4. Navigate to BSP Settings

* + 1. Click **Modify BSP Settings**

A screenshot of a computer screen

Description automatically generated

Figure 5. Modify BSP Settings

* + 1. Notice **Standalone\_bsp\_0** in the **Project name** field and click **Finish** with default settings.

A Board Support Package Settings window will appear.

* + 1. 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**.

A screenshot of a computer

Description automatically generated

Figure 6. Enable profiling in the board support package

* + 1. Select the **Overview > drivers > cpu\_cortexa9** and add **–pg** in the *extra\_compiler\_flags* *Value* field.

A screenshot of a computer

Description automatically generated

Figure 7. Adding profiling switch

* + 1. Click **OK** to accept the settings and create the BSP.

1. Create the Application Step 6
   1. Create the *demoProfile* application using the provided demoProfile.c, fir.c, fir.h, fir\_coef.dat, and xfir\_fir\_io.h files.
      1. Select **demoProfile > src** in the project view, right-click, and select **Import Sources.**
      2. Click **Browse** next to the From Directory**.**
      3. Browse to the folder containing the demoProfile.c file that you downloaded with this demo and click **Select Folder**.
      4. In the panel on the right, check the box next to **demoProfile.c,** **fir\_coef.dat, fir.c, fir.h,** and **xfir\_fir\_io.h,** and click **Finish.**
      5. Right click on the *demoProfile\_system* and select **Build Project**.
      6. A screenshot of a computer program

         Description automatically generatedOpen the *demoProfile.c* file and scroll to the main function at the bottom. Notice the following code:

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.

1. Run the Application and Profile Step 7
   1. Program the PL section and run the application using the user-defined SW\_PROFILE symbol.
      1. Power ON the board.
      2. Select **Xilinx Tools > Program FPGA** and click on **Program**.
      3. Right click on the *demoProfile Project [standalone on ps7\_cortexa9\_0]*, and select **C/C++ Build Settings**.
      4. 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**, and click **OK**.

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

A screenshot of a computer

Description automatically generated

Figure 8. Add user-defined symbol

* + 1. Under the **ARM v7 gcc compiler** group, select the **Profiling** sub-group, then check the **Enable Profiling** box, and click **Apply and Close**. If you are asked to rebuild, select **Yes** to rebuild.

A screenshot of a computer

Description automatically generated

Figure 9. Compiler setting for enabling profiling

* + 1. Right click on the demoProfile\_system and select Build Project. Once the build is finished, select Xilinx > Program FPGA and click on Program.
    2. Right click on the demoProfile project and Select Run As > Run Configurations… and double click on Single Application Debug GDB to create a new configuration.
    3. Click on the newly created demoProfile Debug configuration, and select the Application tab.
    4. Click on the *Advance Options* **Edit…** button.
    5. 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, and click **OK**.

A screenshot of a computer

Description automatically generated

Figure 10. Profiling options

* + 1. Click the **Run** button to download the application and execute it.

The program will run.

* 1. Analyze the results.
     1. When execution is completed, the Gmon File Viewer dialog box will appear showing *demoProfile.elf* as the corresponding binary file. Click **OK**.
     2. Click on the **Sort samples per function** button ().
     3. 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) or 3.166 (Zybo) microseconds were spent per call.

A screenshot of a computer

Description automatically generated

Figure 11. Sorting results

* + 1. Go back to the *Run Configuration*, and change the sampling frequency to **1000000** (1 MHz) and profile the application again.
    2. When execution is completed, click **OK** and the gprof viewer will be updated.
    3. 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 5.483 microseconds in the figure below.

A screenshot of a computer

Description automatically generated

Figure 12. 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.

* 1. Profile the application using the hardware FIR filter IP by removing the user defined SW\_PROFILE symbol.
     1. Select the *demoProfile* application, right-click, and select **C/C++ Build Settings**.
     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.

A screenshot of a computer

Description automatically generated

Figure 13. Deleting the user-defined symbol

* + 1. Click **Apply**, and then click **Apply & Close.** If prompted to rebuild now, select **Yes**.
    2. Right click on the demoProfile\_system and select Build Project. Once the build is finished, select Xilinx > Program FPGA and click on Program.
    3. Select Run > Run Configurations and click the Run button to profile the application again and click OK when profiling completes.

A screenshot of a computer

Description automatically generated

Figure 12. Deleting the user-defined symbol

* + 1. Close Vitis and Vivado programs by selecting **File > Exit** in each program.
    2. Turn OFF the power on the board.