**METHODOLOGY AND TOOL FLOW**

Figure 1 represents the experimental setup for the profiler interfacing. A high level application code is executed on a processor in run time. The profiler is interfaced with the output of the processor and executes non-intrusively. It requires the availability of the following real time trace signals from the processor to enable profiling and does not require the application code to be modified.

1. Trace Instruction signal: Provides the run time opcode values as the processor executes the code.

2. Trace Valid Instruction signal: The information on the instruction bus is valid only when Trace Valid Instruction signal is high.

3. Trace Jump Taken signal: Is asserted only when a jump is taken.

4. Trace PC signal: Gives out the run-time value of the Program counter.

Hence, the proposed approach does not require the processor implementation to be modified to generate any signals to perform profiling. The processor used in our experimental setup is MicroBlaze [30] due to availability and accessibility of trace signals (also referred to in Section-2 for its use in other FPGA based hardware profilers). It is an embedded processor soft core which is a reduced instruction set computer (RISC) optimized in Xilinx Multiprocessor Silicon-On-Chip Field Programmable Gate Arrays (MPSoC FPGAs). 32-bit version of the processor which is available across majority of the kits is used for this implementation. As MicroBlaze is a commercial processor, the RTL implementation of the processor can’t be accessed and modified.

The real time output of the profiler post implementation can be checked by interfacing a logic analyser. As our current experimental setup is Xilinx FPGA based, Chipscope Logic Analyzer (CLA-Xilinx ISE tool based) or Integrated Logic Analyzer (ILA Xilinx Vivado Tool based) can be used for the same.

We would like to mention here that the actual working and application of the profiler is not limited to FPGA. If required, based on the application and heterogenous platform chosen, it can be implemented and interfaced as an ASIC too with a processor whose trace signals are accessible.

Xilinx has two primary design and implementation environments to program its FPGAs, namely:-

1. Xilinx Integrated System Environment (ISE) Design Suite – earlier supported only 5 & 6 series MPSoC FPGA Families (eg. Spartan 6 and Virtex 6), latest versions now support more kits.

2. Xilinx Vivado Design Suite – Supports MPSoC FPGA Family - 7 series onwards (eg. Artix 7 and Kintex 7).

Both of these flow were explored during the duration of the thesis as a part of the experimental survey. The following section describes the profiler implementation flow for both the softwares specific to the profiler design, implementation and verification.

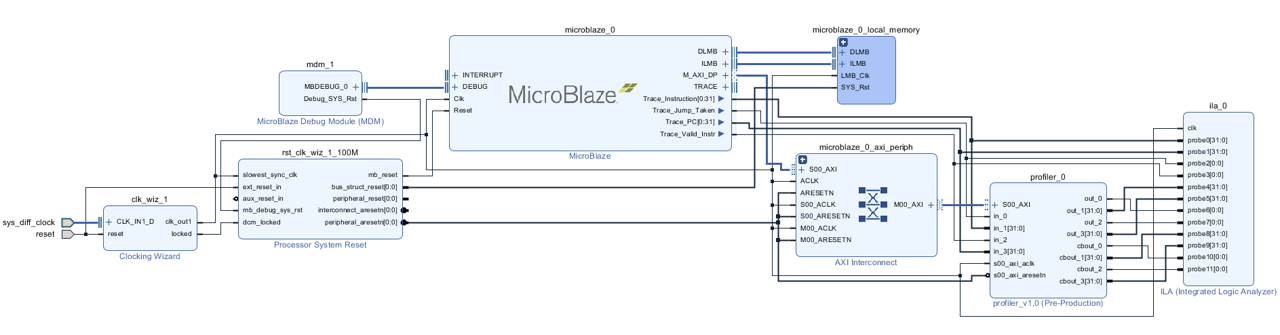
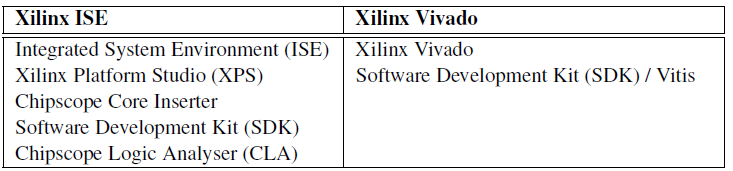


Table 1 Tool Flow



**3.1 Implementation Process**

**3.1.1 Xilinx ISE Design Suite**

Referring to Table-1, ISE Flow comprises of 5 tools as follows:-

1. Integrated System Environment (ISE) – Tool that allows the designer to do HDL code design, synthesis and implementation for a target FPGA. In the explored tool flow, the profiler is coded using ISE.

2. Xilinx Platform Studio (XPS) – The profiler verilog files are then input to the XPS as a user defined custom IP. XPS allows selection of other customized heterogenous platform components such as processors, logic analyser, GPIOs and communication bus interface.

3. Chipscope Core Inserter – This is another separate tool which is invoked in order to select the signals of the XPS platform to be viewed in the Chipscope Logic Analyser. The chosen configuration is then inserted back into the XPS to complete the final platform configuration.

4. Software Development Kit (SDK) – The platform is then synthesized and the generated netlist and bitstream are forwarded to the SDK Environment. in order to program the processor (Microblaze in our case). This allows us to create an application project for our C/C++ application code to be executed on top of this platform. The SDK compiles the entire project to generate the .elf (Executable Link Format) file, the binary file and memory mappings corresponding to the hardware platform for programming the entire project onto the FPGA.

5. Chipscope Logic Analyser (CLA) – This tool is invoked by the user in parallel with the SDK, after the FPGA is switched on and programmed with the platform

configuration plus the selected application binary via SDK. CLA detects the programmed FPGA core. The CLA can be optionally configured to trigger on

specific transition values of the mapped signals such as reset, program counter, instruction opcode etc. in order to view meaningful data on the output waveform

window of a fixed buffer depth. Once the triggers are set, the programmed FPGA is triggered to execute via SDK and the real time execution data can be seen on

the CLA’s output waveform window.

3.1.2 Xilinx Vivado Design Suite

Xilinx Vivado Design Suite has a more integrated approach as compared to Xilinx ISE Design Suite. The overall flow though remains the same but it requires only two

separate tools to be invoked for this implementation.

1. Xilinx Vivado – This helps design the profiler in HDL, package it into a custom IP, establish connectivity with other IPs and implement the final heterogenous

platform. The platform components in our case include the profiler IP, MicroBlaze soft IP and the Integrated Logic Analyser (ILA). The output through the ILA can

be viewed within Vivado itself via the hardware manager.

2. Software Development Kit (SDK) / Vitis – SDK (until Vivado 2019.1) or Vitis (Vivado 2019.2 onwards) is used the program the processor and execute the

application code as explained in detail in the above section.

3.2 Verification Flow

In order to verify the functionality of the profiler during the RTL design process, a platform containing only the Microblaze and the Xilinx Logic Analyzer IP Core without

the profiler is created to record the run-time execution traces of Microblaze. A comprehensive set of test cases (C programs) inclusive of benchmark codes is created

to cater to different coding styles found in typical embedded systems application codes, and to check for corner case behavior of the profiler code. These test cases are executed

on the MicroBlaze synthesized in the target FPGA, and the run-time execution traces are captured through the Logic Analyzer. The captured traces are then rendered into

a Verilog Test Bench for verification of the profiler Verilog RTL code in a simulator. The .elf file as mentioned in Section- 3.1.1 above helps us view the program counter,

instruction opcode and assembly code translation with respect to the C code. This file for multiple C codes along with the Microblaze instruction set is analysed to design

the instruction decoder and control block of the profiler (explained more in Chapter-4). Once the profiler functionality is verified, it is added to FPGA fabric to test its execution

comprehensively on a much larger set of test cases and benchmarks as explained in Section-3.1 above.