ZyCAP User Guide

1 Introduction

ZyCAP is a custom ICAP controller for high speed and efficient partial reconfiguration on Xilinx's Zynq SoCs. It has a hardware IP core, which can be used within the Xilinx's XPS environment to integrate with the Zynq processing system (PS). The ZyCAP driver takes care of low level reconfiguration operations, bitstream caching and bitstream memory management.

2 Hardware integration

The ZyCAP hardware is available as a *pcore*, which can be imported to any Xilinx's XPS project. The source code for the IP is located at Zycap/hw/pcores /zycap_v1_00_a/. This IP can be added to the XPS project by copy pasting this directory to the XPS pcore directory or by using the Create or Import Peripheral wizard. ZyCAP has two AXI interfaces, of which the S_AXI_LITE interface has to be connected to a GP port and the M_AXI_M2ICAP interface has to be connected to an HP port. The *icap_intr_out* signal has to be connected to the PS as an interrupt signal.

3 Software integration

The software driver source code for ZyCAP is located at Zycap/hw/pcores/ zycap_v1_00_a/driver. These files must be added to the SDK project to access the ZyCAP hardware. The software application should use the header file *zycap.h* to access the ZyCAP APIs. Table 1 gives the ZyCAP APIs. Refer to the example application code Zycap/sw/example_app/zynq_pr.c for detailed use cases.

4 Example Application

An example application is provided to verify the effectiveness of ZyCAP in reconfiguration management. The application can equally run on Xilinx's ZC702 evaluation board and on the Zynq community supported Zedboard. The top level software application for the example is provided in the Zycap/sw/example_app directory. The software executable for this application has to be generated the SDK by importing the ZyCAP driver. Prebuilt bitstreams for the example are

Table 1: ZyCAP APIs.

API Call with Brief Description
`Init_Zycap(* intr_controller)`
Initialise the Zycap controller, memory allocate for PR bitstreams A pointer to the interrupt controller has to be passed as an argument to the API.
`Config_PR_Bitstream(bitstream_name, intr_sync)`
Reprogram using a bitstream. If the intr_sync option is set, the API returns immediately after initialising the reconfiguration. The Sync_Zycap() API should be used in this case before accessing the reconfigured peripheral.
`Prefetch_PR_Bitstream(bitstream_name)`
Prefetch the PR bitstream from SD card to DRAM
`Sync_Zycap()`
Synchronise Zycap reconfiguration interrupt

provided in the Zycap/hw/bitstreams directory. The two partial bitstreams *config1.bin* and *config2.bin* has to be initially copied to the on-board SD card. The partial bitstreams correspond to two custom peripherals implemented on Zynq, which has a single register. In config1, any data written to this register is incremented by 1 before storing. In config2, the data is decremented by 1 before storing. The source codes for the peripheral is available in the Zycap/hw/user_logic_src directory. After powering up the board, Zynq has to be initially configured using the config1.bit full bitstream. The board has to be connected to the host PC using UART in order to view the output. Now download the software executable using the *Run* option of SDK. The application shows the effects of bitstream caching and non-blocking reconfiguration operation.

4.1 Rebuilding the example application

If users wish to rebuild the complete example application for better understanding of tool flow, they should follow steps given below

1. Xilinx tool verstions 14.6 or above has to be used for development
2. Open the XPS project located at Zycap/hw/system.xps
3. It could be seen that the ZyCAP IP is available under the *USER* tab in the IP catalogue pane
4. In addition to ZyCAP, a *REG_PERIPHERAL* IP core is also present in the IP core directory. This is the core being partially reconfigured.
5. It could be seen from the source code (Zycap/hw/pcores/reg_peripheral_v1_00_a/hdl/verilog), it is being instantiated as a *black box*, which is one of the requirements for

partial reconfiguration designs.

6. From the XPS project, notice how ZyCAP is connected with the Zynq PS using through AXI interconnects and from the *Ports* tab observe how the interrupt signal is connected.

7. Now generate netlist for the design by choosing Hardware→Generate netlist option.

8. Export the hardware to SDK using the Project→Export hardware design to SDK option by checking *export and launch SDK* option.

9. The XPS project may be closed now.

10. In the SDK, choose Standalone as the board support package and import the Zycap/sw/example_app/zynq_pr.c file into the source folder.

11. The ZyCAP driver source files located at Zycap/hw/pcores/zycap_v1_00_a/driver should be also imported to the project either by using the import→General→File System option or by choosing it as a repository.

12. Now build the project to generate the *elf* file.

13. Now open the PlanAhead project (project_1.ppr) located at Zycap/hw/PlanAhead.

14. The netlists for the black box modules are pre-generated and stored in the Zycap/hw/user_logic_src directory.

15. It could be seen that there are two configurations in PlanAhead corresponding to the two implementations of the *REG_PERIPHERAL*.

16. The floorplanning for the PR regions is already done. Users are free to change the floorplan, but the size of the resulting bitstreams has to be used in the zynq_pr.c application file for accurate performance measurement.

17. Now run the two configurations in PlanAhead to generate the full and partial bitstreams.

18. Use the promgen command *promgen -b -w -p bin -data_width 32 -u 0 pr_bitstream.bit -o pr_bitstream.bin* to convert the partial bitstreams into .bin format, which can be now stored on the SD card. The names of the .bin files generated should be less than 8 characters in size and should match the names used in the zynq_pr.c application software.

19. Now from SDK, download the config1.bit full bitstream and then download the software executable.