UPaRC Quick Start Guide

Robin Bonamy, Sébastien Pillement and Hung Manh Pham
University of Rennes 1 - CAIRN-IRISA
sebastien.pillement@irisa.fr, robin.bonamy@irisa.fr

March 20, 2012

Introduction

UPaRC is a hardware intellectual property (IP) core dedicated to Xilinx FPGAs.

This IP provides very high speed reconfiguration rate including power consumption consideration. This document includes file details to set up UPaRC. This software was tested on Virtex 5 and Virtex 6 architectures, using the Xilinx ISE 12.4. The name provided here may change depending on the user change and the software version.

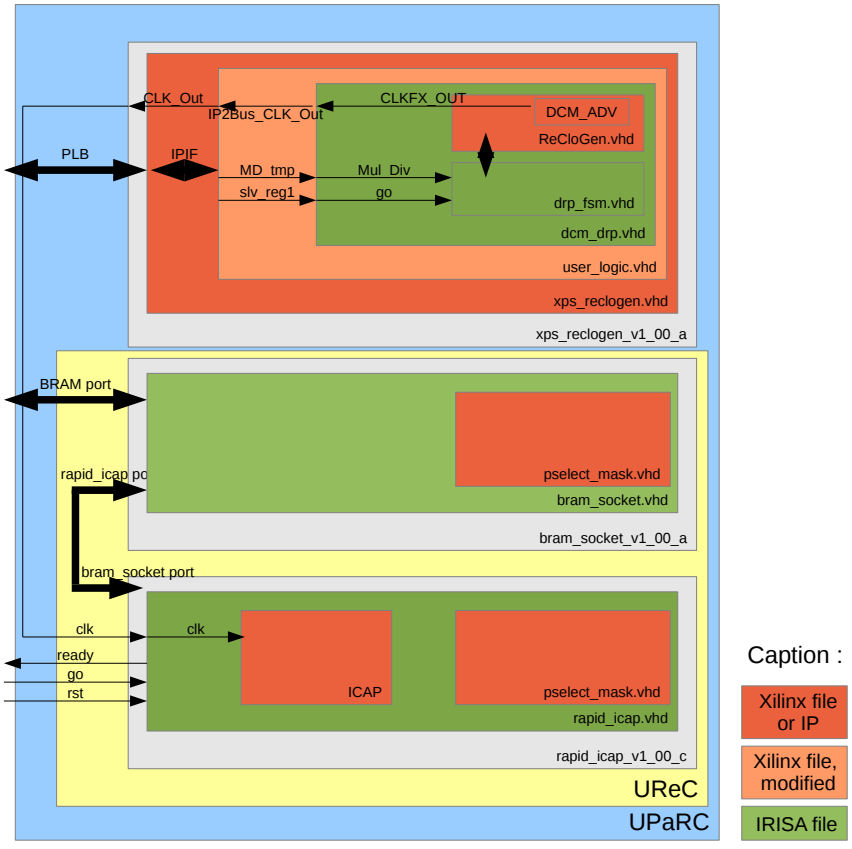

Figure 1: Hierarchical schematic of UPaRC files.

The Figure 1 presents the hierarchy of blocks involved in UPaRC instantiation. Rapid ICAP

and bram socket IPs are mandatory to instantiate the reconfiguration controller (UReC). Reclogen is an IP to manage reconfiguration clock frequency and is used to face power considerations, it is not mandatory in case you won't manage power during runtime. Finally some software tools are provided in order to format the bitstream to be handled by UReC.

The pcores directory contains all the vhdl and parameters files required for the synthesis of UPaRC. The software directory contains the two above mentioned programs.

Provided IPs are designed to be used in Xilinx EDK. pcore provided directory is intended to replace or complete existing pcore directory in your project tree. Then three IPs can be added to the project as user core. You MUST add desired size of (dual port) BRAM to be used for UPaRC in EDK project. One port must be connected to bram_socket_v1_00_a and the other one can be used to preload bitstream at runtime, typically using a *MicroBlaze*. UPaRC stat signal can be controlled by a GPIO.

UPaRC can be used outside EDK project but it is not recommended if you are not comfortable with BRAM connection.

The following sections explain the role and the usage of each block and required file modifications.

1 xps_reclogen_v1_00_a

Reclogen is an IP designed to dynamically modify clock speed delivered to the reconfiguration port (ICAP). This IP is controlled by a microprocessor core through multiplication and division ratio registers addressed by a PLB bus. Clock is generated using a FPGA DCM dedicated module.

1.1 xps_reclogen.vhd

xps_reclogen.vhd is the file generated through the *create IP wizard* to interface microprocessor with reclogen with two registers and no memory.

add `IP2Bus_CLK_Out : out std_logic`
at the end of the xps_reclogen port section
add `IP2Bus_CLK_Out => CLK_Out`
in the user_logic instantiation.

1.2 user_logic.vhd

user_logic.vhd file is a Xilinx generated file (from the same wizard as previous) but is modified as follows :

`use xps_reclogen_v1_00_a.dcm_drp;`
in the libraries section.
add port for clock out : `IP2Bus_CLK_Out : out std_logic`
and add the dcm conection module at the end:

```
dcm_drp_ins : entity xps_reclogen_v1_00_a.dcm_drp
   port map ( Rst       => Bus2IP_Reset,
              CLK       => Bus2IP_Clk,
              Mul_Div   => MD_tmp,
              go        => slv_reg1(31),
              CLKFX_OUT => IP2Bus_CLK_Out
            );

  SWAP_HALF_WORD: process(slv_reg0)is
  begin
```

```
    for i in 0 to 15 loop
      MD_tmp(i) <= slv_reg0(31-i);
    end loop;
  end process SWAP_HALF_WORD;
```

1.3 dcm_drp.vhd

dcm_drp.vhd is a IRISA file. It is used to interface reclogen state machine with DCM. So it includes the drp_fsm.vhd file.

1.4 Reclogen.vhd

Reclogen.vhd is a Xilinx generated file using coregen for a DCM module. It is used for a DCM connection.

1.5 xps_reclogen_v2_1_0.mpd

mpd files are required in Xilinx IP cores especially to set IOs.

Modify the xps_reclogen_v2_1_0.mpd to add

```
PORT CLK_Out = "", DIR = O
```

to have the CLK_Out port in EDK.

1.6 xps_reclogen_v2_1_0.pao

pao files are used to include other vhdl files than the xps.. file.

Finally add

```
lib xps_reclogen_v1_00_a dcm_drp vhdl
lib xps_reclogen_v1_00_a drp_fsm vhdl
lib xps_reclogen_v1_00_a ReCloGen vhdl
```

to the pao file.

2 rapid_icap_v1_00_c

Rapid ICAP is the IP which controls the reconfiguration procedure. Configuration data are read through bram socket which is presented in next section.

2.1 rapid_icap.vhd

rapid_icap.vhd is an IRISA file.

This is the core of UPaRC. This file includes the finite state machine required to handle reconfiguration and includes interface to ICAP. It requires Xilinx pselect_mask.vhd file which can be found in the proc_common pcore in the Xilinx setup path.

2.2 rapid_icap_V2_1_0.pao

Add `lib rapid_icap_v1_00_c pselect_mask vhdl`

to the rapid_icap_V2_1_0.pao.

2.3 to the rapid_icap_V2_1_0.mpd

Declare the input/ouput of the rapid_icap in the .mpd file:

```
# Signals
PORT clk = "", DIR=IN, SIGIS=CLK
PORT rst = "", DIR=IN
PORT go = "", DIR=IN
PORT ready = "", DIR=OUT

PORT BRAM_Rst = BRAM_Rst, DIR = O, BUS = PORTA
PORT BRAM_Clk = BRAM_Clk, DIR = O, BUS = PORTA
PORT BRAM_EN = BRAM_EN, DIR = O, BUS = PORTA
PORT BRAM_WEN = BRAM_WEN, DIR = O, VEC = [0:3], BUS = PORTA
PORT BRAM_Addr = BRAM_Addr, DIR = O, VEC = [0:31], BUS = PORTA
PORT BRAM_Din = BRAM_Din, DIR = I, VEC = [0:31], BUS = PORTA
PORT BRAM_Dout = BRAM_Dout, DIR = O, VEC = [0:31], BUS = PORTA

END
```

3 bram_socket_v1_00_a

This IP provides interface between rapid ICAP and the BRAM IP from Xilinx.

3.1 bram_socket.vhd

bram_socket.vhd file is an IRISA file.
it requires pselect_mask.vhd too.

3.2 bram_socket_V2_1_0.pao

Add `lib bram_socket_v1_00_a pselect_mask vhdl`
to the bram_socket_V2_1_0.pao file.

3.3 bram_socket_V2_1_0.mpd

Declare also the connection and the pad of bram_socket_V2_1_0 in the correct file (i.e. the .mpd): add

```
# Signals
PORT Rst = Rst, DIR = I, SIGIS = RST
PORT Clk = Clk, DIR = I, SIGIS = CLK
PORT EN = EN, DIR = I
PORT WEN = WEN, DIR = I, VEC = [0:3]
PORT Addr = Addr, DIR = I, VEC = [0:31]
PORT Din = Din, DIR = O, VEC = [0:31]
PORT Dout = Dout, DIR = I, VEC = [0:31]

PORT BRAM_Rst = BRAM_Rst, DIR = O, BUS = PORTA
PORT BRAM_Clk = BRAM_Clk, DIR = O, BUS = PORTA
PORT BRAM_EN = BRAM_EN, DIR = O, BUS = PORTA
PORT BRAM_WEN = BRAM_WEN, DIR = O, VEC = [0:3], BUS = PORTA
PORT BRAM_Addr = BRAM_Addr, DIR = O, VEC = [0:31], BUS = PORTA
PORT BRAM_Din = BRAM_Din, DIR = I, VEC = [0:31], BUS = PORTA
```

```
PORT BRAM_Dout = BRAM_Dout, DIR = O, VEC = [0:31], BUS = PORTA

END
```

4 UPaRC bitstream conversion file

Two programs, `bit2uparc` and `bit2mem`, are provided to convert the bitstreams to load to UReC. As UReC is a very simple controller with a single start signal, the size of the handled bitstream is required at the beginning of the transfer. These converters replace Xilinx bitstream header by only the bitstream's size. Bitstream content is not modified.

Source code is provided and so need to be compiled. Using gcc and Linux for example, `gcc bit2uparc.c -o bit2uparc` and `gcc bit2mem.c -o bit2mem`.

Since bitstream must be preloaded to BRAM buffer before partial dynamic reconfiguration, either data are preloaded with the FPGA configuration, at startup, using the BRAM data bitstream section or data are loaded during runtime by the reconfiguration manager.

Only output file is different between two software.

4.1 bit2uparc

`bit2uparc` translates a Xilinx generated bitstream file to UPaRC reconfiguration data. File generated contains the bitstream length in the first 32bits. Then configuration data are directly copied from the original bitstream file.

- offboard:
  - run `./bit2uparc partial.bit partial.uparc`
- onboard:
  - directly load the complete `partial.uparc` file to UPaRC BRAM.
  - handle UPaRC `start` signal

4.2 bit2mem

`bit2mem` translates Xilinx bitstream files into files used to preload BRAM memory. It does the same modifications that `bit2uparc` do but output file is in mem format to add it to Xilinx bitstream.

- offboard:
  - run `./bit2mem partial.bit partial.mem`
  - run `data2mem -bd partial.mem -bt system.bit -ob download.bit`
- onboard:
  - handle UPaRC `start` signal

Conclusion

This document has presented files and modifications required to set up UPaRC and enhance reconfiguration performance. In case of any problem feel free to contact us.