

# 1 DPRC – FPGA Dynamic Partial Reconfiguration controller with DMA AHB interface

## 1.1 Overview

The Dynamic Partial Reconfiguration Controller (DPRC) enables self-run-time partial reconfiguration of Xilinx Virtex-4/5/6 and 7 Series FPGAs through the AMBA AHB and the Internal Configuration Access Port (ICAP).

Dynamic Partial Reconfiguration extends the native flexibility of FPGAs making it possible to dynamically change selected portions of a circuit while the rest of the design is left unchanged and fully functional. The ability to time-multiplex hardware modules at run-time enables designing complex Systems-on-Chip (SoCs), thus reducing cost, logic resources, and power consumption.

DPR is performed by mean of partial bitstreams, which are binary files storing the physical configuration of a selected logical module inside the FPGA device. These files are generated resorting to the Xilinx PlanAhead or Vivado tools during the Partial Reconfiguration design flow, and can be delivered to the ICAP interface to reconfigure a portion of the FPGA's internal configuration memory.

DPRC can be configured through VHDL generics to operate in four different operating modes:

- *Synchronous (Sync)*,
- *Asynchronous (Async - Figure 1)*,
- *Dependable DPR with CRC (D<sup>2</sup>PRC-CRC)*,
- *Dependable DPR with EDAC (D<sup>2</sup>PRC-SECDED)*.

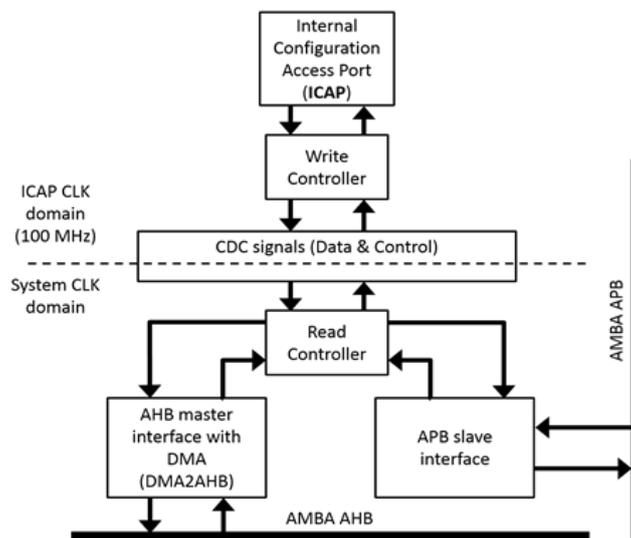


Figure 1. DPRC configured in Asynchronous mode.

In all operating modes, DPRC is connected on both AMBA AHB and APB buses. The APB interface provides access to control and status registers, that are used to control and monitor the entire partial reconfiguration process at run-time. The AHB master interface, with direct memory access capabilities, is used to retrieve partial bitstream data from a memory-mapped peripheral connected to the AHB bus. This interface is implemented using the *DMA2AHB* core included in GRLIB. AHB read operations are managed through a finite state machine (Read Controller). Bitstream data are sent to an asynchronous FIFO. Finally, another state machine (Write Controller) reads data from the FIFO and delivers them to the ICAP. Write controller works in the ICAP clock domain (100 MHz is the maximum allowed ICAP frequency) and it implements the “Non-Continuous Data Loading with Free-Running CLK” protocol [3-6].

If the bus/system clock is lower than 100 MHz, Sync mode should be preferred to the Async mode. In this case, since the bottleneck of the data transfer is the AHB bus, the ICAP can be clocked using the bus/system clock, avoiding synchronization FIFO and registers for CDC signals. This leads to a reduced DPRC complexity, and consequently a significant reduction of FPGA's resources needed to implement the controller.

In Dependable mode with CRC (D<sup>2</sup>PRC-CRC), the more complex controller includes a 32-bit Cyclic Redundancy Check (CRC) module, which assists the Read Controller, checking the bitstream data prior to loading them into the ICAP. This can be useful when there is the possibility that bitstream data can be corrupted. In fact, if a faulty bitstream is loaded into the configuration memory, the system may become inoperable, and the FPGA device must be reconfigured from the scratch. However, when using dynamic partial reconfiguration in 7 Series Xilinx FPGAs this mechanism is already embedded in the ICAP (the so-called *PerFrameCRC* functionality [3]), thus the Async or Sync operating modes are sufficient to deal with faulty bitstreams.

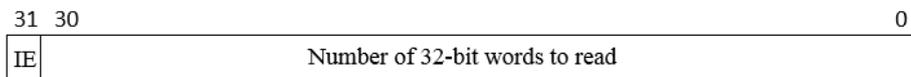
In Dependable mode with EDAC (D<sup>2</sup>PRC-SECDED), the Read Controller is instead assisted by a Single Error Correction Double Error Detection module. With respect to D<sup>2</sup>PRC-CRC mode, the reconfiguration process is stopped only if a double uncorrectable error is detected on a bitstream data word, while all single errors are corrected without halting the flow.

## 1.2 Operations

Either working in Sync, Async or D<sup>2</sup>PRC modes, DPRC shows five 32-bit registers on the APB address space.

Table 1. DPRC APB registers

APB address offset	Register
0x0	Control Register
0x4	Address Register
0x8	Status Register
0xC	Timer Register
0x10	Reconfigurable Modules Reset Register



30: 0 Size of the bitstream to be loaded (in terms of 32-bit words) from the memory. In this DPRC version, only the 20 least significant bits (19:0) of this field are taken into account by the internal controller. This leads to a maximum DMA transfer size of 2<sup>20</sup> 32-bit words (4 MBytes).

31 Interrupt Enable. Set this bit to enable interrupt. An interrupt is generated at the end of the reconfiguration process (when the Status Register is updated). The interrupt signal is active-high and lasts one AHB clock cycle.

Figure 2. Control register.



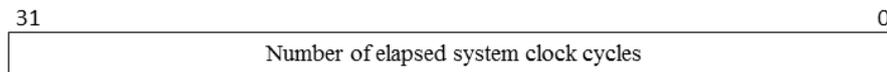
31: 0 Address of the partial bitstream to be loaded.

Figure 3. Address register.



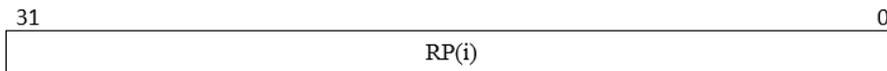
- 31: 24 RESERVED.
- 23: 4 NUMBER OF CORRECTED ERRORS. This field is only valid when working in D<sup>2</sup>PRC-SECDED mode. It shows the number of bitstream errors corrected by the SECDED engine.
- 3: 0 STATUS.
- “1111” Partial Reconfiguration Ended Successfully (No errors);
- “0100” Reconfiguration aborted due to AHB transfer error;
- “0001” Reconfiguration aborted due to CRC error;
- “0010” Reconfiguration aborted due to Double Error Detected;
- “1000” Reconfiguration aborted due to ICAP error.

Figure 4. Status register.



- 31: 0 Bus clock cycles elapsed from the start to the end of the last reconfiguration process.

Figure 5. Timer register.



DPRC supports up to 32 different reconfigurable partitions in the design. Each bit of this register is associated to a bit (in the same position) of the *rm\_reset* DPRC output signal. Each bit of the *rm\_reset* signal can be associated to the synchronous reset signal of a reconfigurable partition, acting as “true” reset signal for reconfigurable modules implemented in that partition. This signal can also act as enable or reset signal of the decoupling logic that should be located at the outputs of the reconfigurable module (see Section 6).

- 31: 0 RP(i): set to ‘1’ to give a reset to the associated partition during the reconfiguration process.

Figure 6. Reset Register.

### 1.2.1 Partial Bitstream data format

Before starting a reconfiguration, the partial bitstream to be loaded must be already available and accessible through the AHB bus. To allow correct operations, it must be pre-processed and formatted depending on the adopted DPRC operating mode. A software application is also provided to automate the bitstream formatting task in all operating modes.

When working in *Sync* or *Async* modes, it is sufficient to count the number of 32-bit words composing the partial bitstream file generated by Xilinx tools, add ‘1’ to the resulting value, and insert it (represented on 32-bits) as first word of the processed bitstream.

When working in *D<sup>2</sup>PRC-CRC* mode, additional steps are required to generate a “CRC-protected” version of the partial bitstream. The provided software application splits the bitstream file in “words blocks”, computes a 32-bit CRC for each block, and reassembles the bitstream file interleaving the computed CRC signatures at the end of each block (see *Figure 7*).

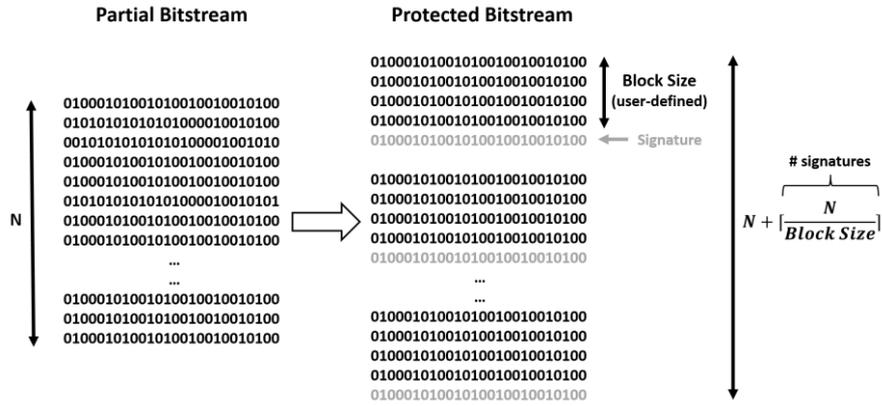


Figure 7. Partial bitstream processing for  $D2PRC-CRC$  mode.

The user can choose through a VHDL generic the number of 32-bit words composing a block to allow a fine-grained or a coarse-grained checking, thus trading-off the level of dependability against bitstream corruption with respect to the bitstream download time. The CRC polynomial implemented by the hardware CRC module is  $1+x^3+x^{14}+x^{18}+x^{29}$  (0x90022004), which guarantees detection of 100% on up to 32 bits burst errors, and detection of 100% on up to five random bit errors in a single CRC block (depending on the actual chosen size).

Instead, when working in  $D^2PRC-SECDED$  mode, the provided software application splits the bitstream in 4 32-words packets. Each 32-bit word in a packet is encoded exploiting a SECDED Hamming code, thus resulting in a 39-bit encoded word (each encoded word is also extended to 40 bits adding a '0' in the most significant bit). Finally, the processed bitstream is assembled as shown in Figure 8.

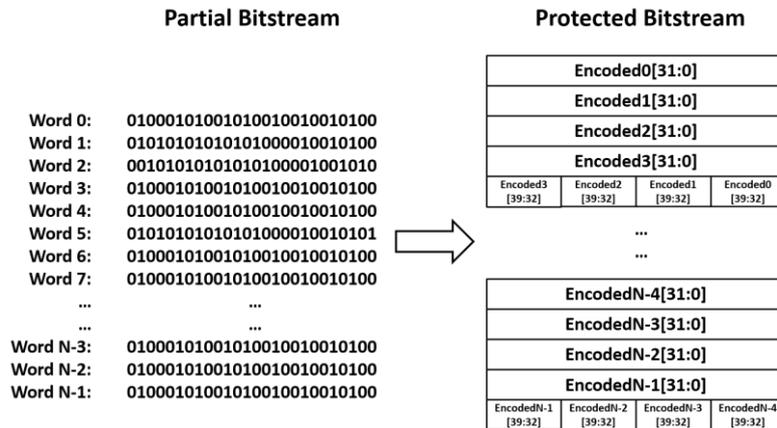


Figure 8. Partial bitstream processing for  $D2PRC-SECDED$  mode.

The C++ source code of the software utility (*dprc\_sw*) is located in `<glib root>/software/dprc/`. The utility takes as inputs one or more raw partial bitstream files (in `.rbt` format) generated by the Xilinx PlanAhead or Vivado software tools. The raw bitfile can be generated by including the `-b` (PlanAhead) or `-raw_bitfile` (Vivado) option when running the bitstream generation process. The output of this program is a header file. Each partial bitstream is processed depending on the chosen DPRC mode and declared in the header as a unidimensional array named `bitstream<i>`, where `<i>` represents the index of the input bitfile. The software utility can be compiled and launched by issuing in a Unix shell:

```
g++ -Wall -fexceptions -O2 -c <glib root>/software/dprc/main.cpp -o <glib root>/software/dprc/main.o
```

```
g++ -o <glib root>/software/dprc/dprc_sw <glib root>/software/dprc/main.o
```

```
./dprc_sw <path of the input bitstream file 0 (.rbt)> <path of the input bitstream file 1 (.rbt)> ... <path of the input bitstream file N (.rbt)> <Number of 32-bit words per block> <path of the output file (.h)>
```

The same applies when using Windows operating systems.

**\*Note:** do not run *dprc\_sw* utility on Linux operating systems if `.rbt` bitstream files have been generated using a Windows version of Xilinx tools.

The parameter `<Number of 32-bit words per block>` is used to select between operating modes. In particular, if the parameter is set to "0", the input files are processed in order to be used in *Sync* or *Async* modes; if the parameter's value is negative, SECDED encoding is performed, while if the value is a positive integer in the range 2 to 496, CRC signatures are computed. In this last case, the value represents the size of a CRC block (in terms of 32-bit words).

The following steps must be executed to setup and start a partial reconfiguration process.

1. Verify if the four least significant bits of the **Status Register** are set to 0xF, meaning that DPRC is available and no errors occurred during the last reconfiguration;
2. Write in the **Address Register** the address of the partial bitstream to be loaded;
3. Set the corresponding bit in the **RM Reset Register** in order to assert a synchronous reset to the target reconfigurable module during reconfiguration;
4. (optional) Set **Control Register** bit 31 to enable interrupt generation;
5. Write in the **Control Register** the number of 32-bit words composing the partial bitstream to be loaded (i.e., the number of words composing the processed bitstream). This write operation clears the **Status Register** and starts the reconfiguration process. The DMA module starts a locked undefined length burst transfer request on the AHB. If working in Dependable CRC mode, at run-time, the DMA module retrieves the partial bitstream data, while the CRC module computes on-the-fly the signatures. At the end of each CRC block, the run-time computed signature is automatically compared with the one embedded at design-time and, if they mismatch, an error signal is raised and the reconfiguration process is halted. If no errors occur, the block is sent to the ICAP. A similar approach is carried out when DPRC works in Dependable-EDAC mode. However, the reconfiguration process is stopped only if an uncorrectable error is detected on a bitstream data word. After writing the Control Register, the reconfigurable modules synchronous reset is asserted until the end of the process. If an error occurs, the `rm_reset(i)` signal associated to the reconfigured partition remains asserted until a new reconfiguration ends successfully. This ensures to not activate a faulty reconfigurable module.
6. Wait until the **Status Register** signals the end of the reconfiguration process or an error (look at the 4 least significant bits). If interrupt has been enabled, it signals that the **Status Register** has been updated.

In addition, the **Timer Register** can be read to monitor the partial reconfiguration execution time (in terms of bus clock cycles).

### 1.3 Vendor and device identifiers

The core has vendor identifier 0x09 (Various Contributions) and device identifier 0x001. For a description of vendor and device identifiers, see GRLIB IP Library User's Manual.

### 1.4 Configuration options

Table 3. shows the configuration options of the core (VHDL generics).

Table 3. Configuration options

Generic	Function	Allowed Range	Default
hindex	AHB master index	0 – NAHBMST-1	2
pindex	APB slave index	0 – NAPBSLV-1	13
paddr	ADDR field of the APB BAR	0 – 16#FFF#	13
pirq	Index of the interrupt line	0 – NAHBIRQ-1	0
technology	Target FPGA technology	'virtex4', 'virtex5', 'virtex6', 'virtex7', 'artix7', 'kintex7', 'zynq7000'	'virtex4'
crc_en	Dependable CRC mode selection	0 – 1	0
edac_en	Dependable SECEDED mode selection	0 – 1	0
words_block	Number of 32-bit words in a CRC block (for Dependable mode)	2 – 496	10
fifo_dcm_inst	Sync / Async mode selection	0 – 1	0
fifo_depth	Number of bits used to address the data buffer	6 - 9	1
cfg_clkmul	clkraw input multiplier	-	1

cfg_clkdiv	clkraw input divisor	-	1
raw_freq	clkraw input frequency in KHz	-	50000
clk_sel	Select between clkraw (synthesized) and clk100 as ICAP clock, when configured in async or d2prc mode	0 - 1	0

To select *Dependable CRC* operating mode 'crc\_en' generic must be set to '1' while 'edac\_en' to '0'. To select *Dependable SECEDED* operating mode 'crc\_en' generic must be set to '0' while 'edac\_en' to '1'. To select *Sync* mode 'crc\_en', 'edac\_en' and 'fifo\_dcm\_inst' must be set to 0. To select *Async* mode 'crc\_en' and 'edac\_en' must be set to 0, while 'fifo\_dcm\_inst' must be set to '1'.

In *Async* or *Dependable* operating modes, a clock generator and an asynchronous FIFO are instantiated in the DPRC core. The clock generator is based on *clkgen* component, while FIFO is based on *syncram\_2p* component, both included in GRLIB. The *clkgen* is used to generate the 100 MHz clock for the ICAP clock domain, starting from an external clock connected to the *clkraw* DPRC input port, while FIFO is used to transfer data between the two clock domains. *cfg\_clkmul* and *cfg\_clkdiv* VHDL generics are used as inputs of *clkgen* to synthesize the 100 MHz clock. However, if the system already includes an auxiliary 100 MHz clock, *clkssel* can be set to bypass *clkgen* and use directly an external 100 MHz clock, which can be connected to *clk100* DPRC input port.

It is worth to note that *fifo\_depth* VHDL generic represents the number of bits used for addressing the *syncram\_2p* data buffer. Consequently, the true FIFO depth, in terms of 32-bit words, is equal to  $2^{fifo\_depth}$ . Moreover, if Dependable CRC mode is selected, FIFO must be sized in order to accommodate at least one CRC block. Therefore, the following equation must be always satisfied when defining *words\_block* and *fifo\_depth* generics:

$$words\_block \leq 2^{fifo\_depth} - 16$$

## 1.5 Signal descriptions

Table 4. shows the interface signals of the core (VHDL ports).

Table 4. Signal descriptions

Signal name	Field	Type	Function	Active
rst	N/A	Input	Reset	Low
clk	N/A	Input	System Clock	-
clkraw	N/A	Input	Raw Clock input (to synthesize 100 MHz)	-
clk100	N/A	Input	100 MHz Clock input (if already available)	-
ahbmi	*	Input	AHB master input signals	-
ahbmo	*	Output	AHB master output signals	-
apbi	*	Input	APB slave input signals	-
apbo	*	Output	APB slave output signals	-
rm_reset	N/A	Output	Reconfigurable modules reset	High

\*see GRLIB IP Library User's Manual

The *rm\_reset* signal is used as reset for reconfigurable modules. It allows the logic inside the reconfigurable modules to start from a known state after the reconfiguration (as suggested in the Xilinx Partial Reconfiguration User Guide [1-2]). Moreover, the same signal can be used as enable or reset signal of the decoupling logic placed between the static and reconfigurable modules.

## 1.6 Library dependencies

Table 5. shows libraries used when instantiating the core (VHDL libraries).

Table 5. Library dependencies

Library	Package	Imported unit(s)	Description
IEEE	Std_Logic_1164	All	Type declarations
IEEE	Std_Logic_Unsigned	All	Type declarations
GRLIB	AMBA	Signals	Signal definitions
GRLIB	DEVICES	Constants	Vendor and device identifiers declarations
GRLIB	DMA2AHB_Package	Components	AHB2DMA types and component declarations
TECHMAP	GENCOMP	Components	Technology constants, <i>clockgen</i> and <i>syncram</i> components declarations
UNISIM	VCOMPONENTS	Components	ICAP component declaration
TESTGROUPOPOLITO	DPRC_PKG	Components	Components declarations

## 1.7 Instantiation

The following example shows how the core can be instantiated.

```
library ieee;
use ieee.std_logic_1164.all;
library grlib;
use grlib.amba.all;
use grlib.devices.all;
library techmap;
use techmap.gencomp.all;
library testgrouppolito;
use testgrouppolito.dprc_pkg.all;

entity dprc_ex is
  port (
    clk : in std_ulogic;
    rstn : in std_ulogic;

    ..... -- other signals
  );
end;
architecture rtl of dprc_ex is

  -- AMBA signals declarations
  . . .

begin
```

```

-- AMBA Components are instantiated here
. . .

-- Dynamic Partial Reconfiguration Controller
p1 : dprc
    generic map(hindex => 2, pindex => 10, paddr => 10, pirq => 10, technology => virtex4,
crc_en => 1, edac_en => 0, words_block => 10, fifo_dcm_inst => 0, fifo_depth => 5, cfg_clkmul => 6,
cfg_clkdiv => 5, raw_freq => 50000, clk_sel => 0)
    port map( rst => rstn, clk => clk, ahbmi => ahbmi, ahbmo => ahbmo(2), apbi => apbi, apbo =>
apbo(10), rm_reset => rm_reset_sig, clkraw => clk_board, clk100 => '0');

end;

```

## 1.8 Known Limitations

- DPRC assumes that no other AHB masters are granted during the whole bitstream data transfer. It requests a single AHB locked burst transfer of undefined length;
- The maximum allowed size for a processed bitstream is 4 MBytes (i.e.,  $2^{20}$  32-bit words);
- In D<sup>2</sup>PRC-SECDED mode, DPRC assumes that the first word of the unprotected partial bitstream represents a value that is multiple of 4. The provided software application handles this limitation replicating from 1 to 3 times the last word in the bitstream. These words does not alter the functionality represented by the bitstream itself since they represent dummy words for the ICAP (usually 0xFFFFFFFF or 0x20000000);
- in Virtex-4 devices programmed using the iMPACT tool, the first partial reconfiguration will fail with status code 0x8 (Configuration error), since, at startup, ICAP does not behaves as documented in [1].

## 1.9 Additional documents

- [1] “*Partial Reconfiguration User Guide – UG702(v14.5)*”, April 26, 2013 – [www.xilinx.com](http://www.xilinx.com)
- [2] “*Vivado Design Suite User Guide – Partial Reconfiguration – UG909(v2015.2)*”, June 24, 2015 – [www.xilinx.com](http://www.xilinx.com)
- [3] “*7 Series FPGAs Configuration User Guide – UG470(v1.8)*”, August 22, 2014 – [www.xilinx.com](http://www.xilinx.com)
- [4] “*Virtex-6 FPGA Configuration User Guide – UG360(v3.8)*”, August 28, 2014 – [www.xilinx.com](http://www.xilinx.com)
- [5] “*Virtex-5 FPGA Configuration User Guide – UG191 (v3.11)*”, October 19, 2012 – [www.xilinx.com](http://www.xilinx.com)
- [6] “*Virtex-4 FPGA Configuration User Guide – UG071(v1.11)*”, June 9, 2009 – [www.xilinx.com](http://www.xilinx.com)
- [7] J. Andersson, S. Di Carlo, P. Prinetto, P. Trotta, “*A portable open-source controller for safe Dynamic Partial Reconfiguration on Xilinx FPGAs*”, in Proc. of the 25th International Conference on Field Programmable Logic and Applications (FPL), September 2015.