RePaBit:

Relocatable **Pa**rtial **Bit**streams

Manual v1.0

Jens Rettkowski, Konstantin Friesen

1. Integration of RePaBit

To create a relocatable implementation design within the Vivado Design Suite 2016.1 environment, the Tcl scripts of RePaBit must be included. To do this, the RELOC_TCL folder and the init.tcl file are copied to the file path *C:\Users\"User Name"\AppData\Roaming\Xilinx*. The init.tcl file is called by the Vivado Design Suite design tool when it is started.

2. Construction of Design

The design must be divided into a static and reconfigurable parts. The static logic must be placed in a single hardware module. To do this, a single IP core must be created that instantiates all the entire static logic/IP cores.

In case of a Zynq SoC, the static IP core incorporates the processor system and other IP cores of the static logic. All connections to the reconfigurable IP cores and to IO pads must run via the created static IP core. Connection lines of the static IP core that lead to several reconfigurable IP cores must be routed individually out of the static IP core.#

If the reconfigurable IP cores have a clock input, the clock lines must also be routed individually from the created static IP core to the reconfigurable IP cores. If only the reconfigurable IP cores are connected to IO pads, these connections must also run via the static IP core.

The IP cores of the reconfigurable hardware modules can be prepared as under the Partial Reconfiguration Design Flow. A Black_Box attribute must be defined within the VHDL code of a reconfigurable IP core, which defines the reconfigurable component within this IP core (marked red, Fig. 3). Figure 3 shows an example of the VHDL code of a reconfigurable IP core. The actual reconfigurable component within the VHDL code is highlighted with the color blue in Fig. 1. The Black_Box attribute (highlighted in red, Fig. 1) refers to this reconfigurable component. In order to load different modules into the netlist for the reconfigurable IP cores after synthesizing the whole circuit design, netlists of the respective reconfigurable hardware modules (marked blue, Fig. 1) have to be created. For this purpose, the names and the number of ports of the components must be identical. The content of these components may be different. Netlists of the reconfigurable hardware modules must be created in the out-of-context mode of the Vivado Design Suite environment. These netlists can then be flexibly loaded into the netlist of the entire circuit design.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;
use IEEE.std_logic_arith.all;
use IEEE.std_logic_unsigned.all;
-- Uncomment the following library declaration if instantiating
-- any Xilinx leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity led_core_clust is
   Port ( reg_in : in STD_LOGIC_VECTOR (2 downto 0);
        reg_out : out STD_LOGIC_VECTOR (2 downto 0));
end led_core_clust;

architecture Behavioral of led_core_clust is

component rp is
  Port          (
    reg_in  : in std_logic_vector(2 downto 0);
    reg_out : out std_logic_vector(2 downto 0)
  );
end component;

attribute black_box : string;
attribute black_box of rp : component is "yes";

begin

reconfig_rpLED : rp
   port map (  reg_in => reg_in,
               reg_out=> reg_out
          );
end Behavioral;
```

Fig. 1 Black Box Attribut

After a static IP core and the reconfigurable IP cores have been created, the complete design must be created within a new project. Within this project, the created IP cores (static IP core and reconfigurable IP cores) must be integrated and connected. The connection of the static IP core to the IO pads also takes place here. Here the necessary constraints are defined within a .ucf file for the IO-Pads. After the complete design has been created, this design must be synthesized to create a netlist.

2. Design Flow of Relocatable Designs

After the complete design has been synthesized, it is useful to create a new folder "Checkpoints" within the project folder. Within this folder all checkpoints within the project will be stored.

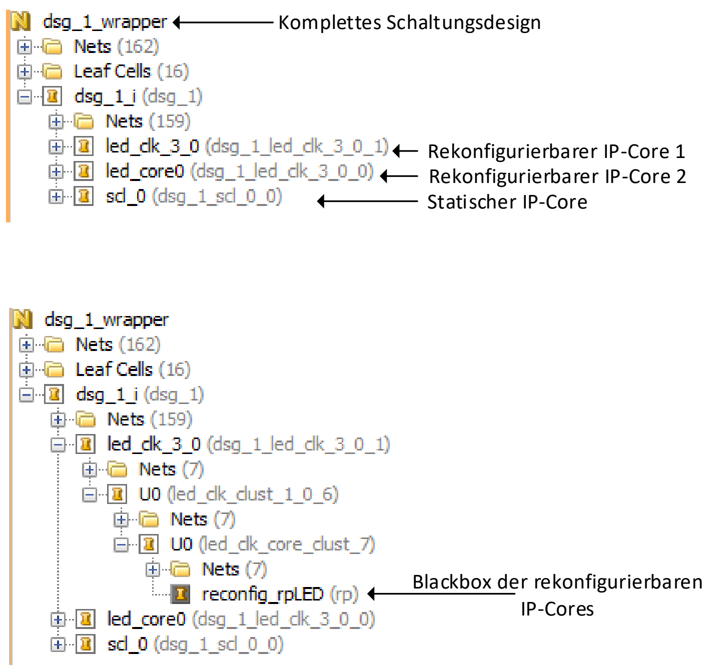
2.1: After synthesis, a checkpoint of the synthesized design is created. To do this, use the Tcl console command

write_checkpoint "Project-Path"/Checkpoints/synth.dcp

2.2: Close the project under **File>Close Project**

2.3: Opening the previously saved checkpoint through the Tcl function ***open_checkpoint "Project-Path"/Checkpoints/synth.dcp***

Within the checkpoint "Project path"/Checkpoints/synth.dcp several instances can be seen in the netlist of the complete circuit design. Once the instance of the static IP core and the instances of the reconfigurable IP cores. Fig. 2 shows an example netlist with three instances (static IP core and reconfigurable IP cores).


```
dsg_1_wrapper
  Nets (162)
  Leaf Cells (16)
  dsg_1_i (dsg_1)
    Nets (159)
    led_clk_3_0 (dsg_1_led_clk_3_0_1)
    led_core0 (dsg_1_led_clk_3_0_0)
    scl_0 (dsg_1_scl_0_0)
```

```
dsg_1_wrapper
  Nets (162)
  Leaf Cells (16)
  dsg_1_i (dsg_1)
    Nets (159)
    led_clk_3_0 (dsg_1_led_clk_3_0_1)
      Nets (7)
      U0 (led_clk_clust_1_0_6)
        Nets (7)
        U0 (led_clk_core_clust_7)
          Nets (7)
          reconfig_rpLED (rp)
    led_core0 (dsg_1_led_clk_3_0_0)
    scl_0 (dsg_1_scl_0_0)
```
Komplettes Schaltungsdesign
Rekonfigurierbarer IP-Core 1
Rekonfigurierbarer IP-Core 2
Statischer IP-Core
Blackbox der rekonfigurierbaren IP-Cores

Fig. 2: Netlist Example

The partitions of the reconfigurable IP cores must be defined within the floor plans. For this purpose, partitions are drawn in the floorplan as under the Partial Reconfiguration Design Flow. Since fixed interface sides must be defined for a relocatable implementation design, adjacent tile types of the partitions must also be considered. It is possible to choose between a left and right interface page to the static logic.

<u>Remarks on Partition Placement:</u>

1.) The partitions must have the same footprint.

2.) Partitions can only be drawn as rectangle.

3.) Partitions must have the height of a clock region.

4.) No INT Tiles which are connected to CLB, BRAM or DSP tils are allowed to be placed outside the partition.

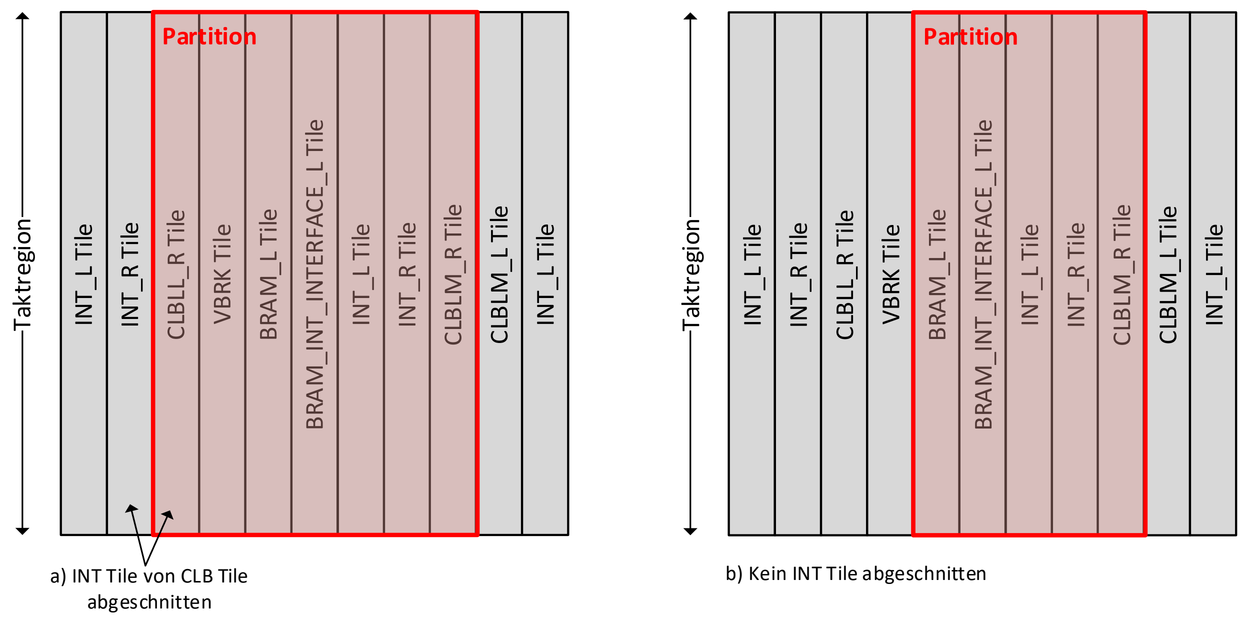


Fig. 3: a) INT Tiles separated by the partition; b) INT Tiles not separated by the partition

INT Tiles are not allowed to be separated py partition borders. (*Fig. 3a*)).

5.) In addition to the partitions, identical arrangements of the tile types must exist so that an identical interface page can be created.

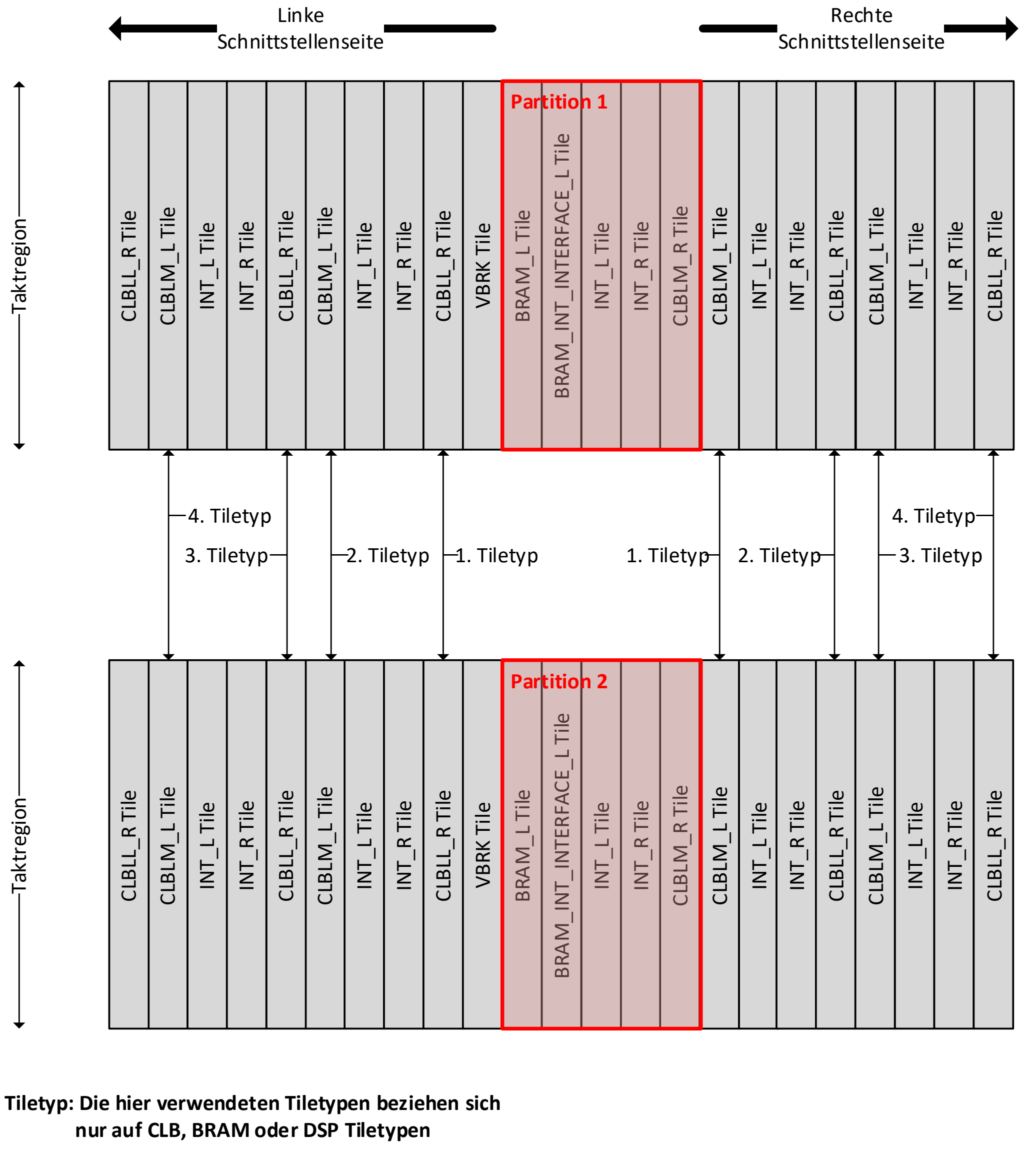


Fig.4: Structure of Tile Types

6.) Within the relocatable implementation design it is possible to define a left or right interface side of the reconfigurable partitions to the static logic. Fig. 4 shows the arrangement of different tile types in the case of a left and a right interface side to the static logic of two reconfigurable partitions with a pin count greater than 400 and less than 800. The 1st tile type must always be identical in the case of DSP or BRAM tiles. If the 1st tile type is of type "CLB", the tile types may be of type "CLBLM" or "CLBLL". However, the placement direction must be identical (Ex: CLBLM_L is compatible with CLBLL_L, but not with CLBLL_R). The tile types 2., 3. and 4. may represent different "CLB" tile types, but must have the same placement direction. The tile types between the 1st tile type (1st tile type is excluded) and the 4th tile type (4th tile type taken) are allowed to be different at the adjacent tile type arrangements of the reconfigurable partitions. However, the order of the "CLB" tile types 2nd, 3rd and 4th of the different reconfigurable partitions must be identical. Depending on how many pins a reconfigurable IP core has, a certain number of "CLB" tile types must be observed next to the partitions (starting with the 1st tile type). For a number of 0<=pins<=400, the "CLB" tiletype order must be identical up to the 3rd tiletype. For a number of 400<pins<=800, the "CLB" tiletype order must be identical up to the 4th tiletype. For a number of 800<pins<=1200, the "CLB" tiletype order must be identical up to the 5th tiletype. This is done analogously for a larger number of pins of the reconfigurable IP cores. Between the "CLB" tiletypes 2. and n. (n=3, 4, 5, 6...; depending on how many pins) different tile types may be placed. If there are connections to IO pads within the design, the partitions must ensure a certain minimum distance to the IO pads. Of the last required tiletype of the interface side, there must be a "CLB" tiletype between the IO pad and the last required tiletype of the interface side.

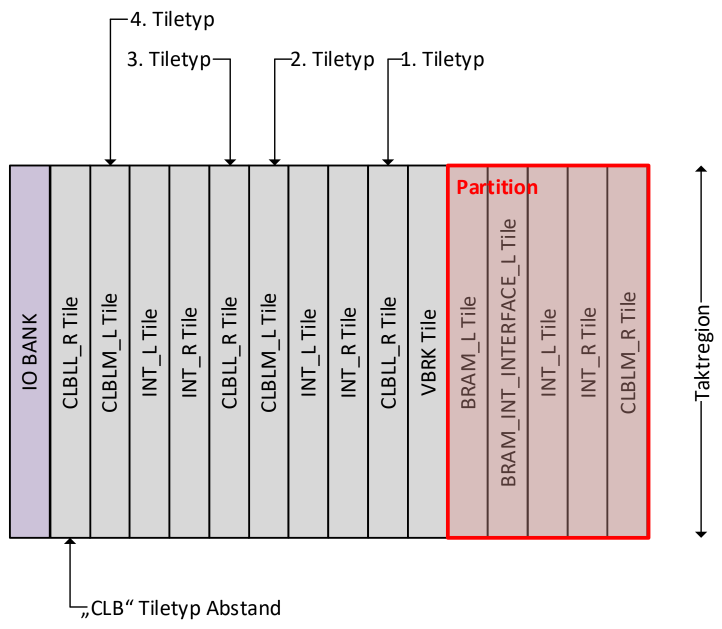


Fig. 4: Minimum distance

Fig. 4 shows an interface on the left side with necessary tile types for a reconfigurable IP-Core.

7.) A partition must not be placed between two different clock regions.

Functions to search for partitions

To determine identical footprints with a matching interface side within the floorplan, a reconfigurable partition can be initially drawn within the floorplan. This initially placed reconfigurable partition can be used to determine compatible partitions within the floorplan.

H.1) Call the Tcl function init_plattform -part xc7z020 in the Tcl console to initialize specific parameters of the xc7z020 SoC.

H.2) Calling the Tcl function get_compatible_pblock_pattern -pblock "Name of the initially placed partition" -interface_side "Desired interface side (LEFT/RIGHT)". **Example:** ***get_compatibel_pblock_pattern –pblock pblock_1 –interface_side LEFT***

The get_compatible_pblock_pattern function returns the position of matching partitions as coordinates within the console. Different partition placements are distinguished with curly brackets by the return values.

Partitions are defined by a lower left corner and an upper right corner. Coordinates within the floorplan can be determined by the properties ROW (Y-coordinate) and COLUMN (X-coordinate) of the tiles within the floorplan. The draw_rect function can be used to draw the partitions automatically. To do this, the function is called in the Tcl console as follows:

H.3) **draw_rect –create –x_ll** *"X-coordinate of the lower left corner of a P-block". –* **y_ll** *"Y-coordinate of the lower left corner of a P-block"* **–x_ur** *"X-coordinate of the upper right corner of a P-block". –***y_ur** *"Y-coordinate of the upper right corner of a P-block"* **–name** *"partition name"*

Example: draw_rect -create -x_ll 129 -y_ll 103 -x_ur 158 -y_ur 53 -name pblock_2

To determine the coordinates of a partition the function get_rect can be called within the console:

H.4) **get_rect -pblocks** “Name of partitions placed” **–exact**

Example: get_rect –pblocks pblock_1 –exact

As return value the function returns the name of the partition, the X-coordinate of the lower left corner, the Y-coordinate of the lower left corner, the X-coordinate of the upper right corner and the Y-coordinate of the upper right corner of the partition.

The function get_compatible_pblock_pattern returns compatible partition placements, but it must be checked whether these partition placements have sufficient distance to the IO pads. Additionally it must be checked if connection lines to the desired interface side of the different partition placements are possible because connection lines are not routed through the partitions.

<u>DRC Functions to check correct placement of partitions:</u>

To check the placement positions of the reconfigurable partitions, different DRC functions can be used which, based on output within the console, indicate to the user the validity of the partition placements within the floorplan.

D.1) By calling the Tcl function report_reloc_drc within the Tcl console, various DRC checks are performed on the partition placements and the interface pages.

Warnings are displayed in yellow within the console. Error messages are output within the console marked red. Warnings about the interface pages should be heeded by the user, since a possible incompatibility between the interface pages of the different partitions can be detected. If the warnings refer to interface pages within the floorplan that are not desired, these warnings can be ignored.

If the DRC functions do not reveal any error messages, you can move forward with creating a relocatable implementation design.

<u>2.4:</u> After the partitions have been placed within the floorplan, the reconfigurable IP cores (still black boxes at this point) from the netlist must be assigned to the partitions.

<u>2.5:</u> Now the static IP core must be declared as static within the netlist. To do this, select the static IP core within the netlist. Under the Properties tab, the property "RELOC_STATIC_PART" is added to the instance of the static IP core by clicking the Plus button and confirming with the OK button (Fig. 10). Now the property "RELOC_STATIC_PART" must be set to TRUE within the properties by clicking on it.

<u>2.6:</u> Now the reloc_design function is called within the Tcl console:

reloc_design –checkpoint “Path for Checkpoint” **-interface_side** “LEFT/RIGHT”

-ref_cell “Instance of reconfigurable IP-Core”

Example:

reloc_design

-checkpoint
C:/Xilinx/ProjectVivado/New_ISPR/Sources/rp_sources/rp1/led_clocked_rp1.dcp

-interface_side LEFT

-ref_cell dsg_1_i/led_clk_3_0/U0/U0/reconfig_rpLED

First the path of the checkpoint (netlist) of the largest reconfigurable IP core is specified under the -checkpoint option. The -interface_side option is used to define the interface side for the reconfigurable partitions with LEFT or RIGHT. The -ref_cell option is used to specify the instance name of a reconfigurable IP core (black box in netlist) from the netlist as reference core.

Remark:

It should be noted that even with valid placements of various reconfigurable partitions, Vivado Design Suite's Router and Place might not find valid routing paths.

2.7: After a relocatable implementation design has been created by the reloc_design function, a new checkpoint of the static circuit design must be created. To do this, a new checkpoint must be created in the console under the project path.

write_checkpoint „Project-Path"/Checkpoints/static_impl_design.dcp

2.8: Close Project: **File>Close Project**

2.9: Open checkpoint that has been previously closed:

open_checkpoint „Project-Path"/Checkpoints/static_impl_design.dcp

Now, different netlists can be loaded for the reconfigurable IP cores within the static relocatable implementation design and partial bitstreams can be created. The opt_design, place_design and route_design functions can be used to implement the netlists. The write_bitstream function can be used to create bitstreams of the whole circuit design and partial bitstreams.