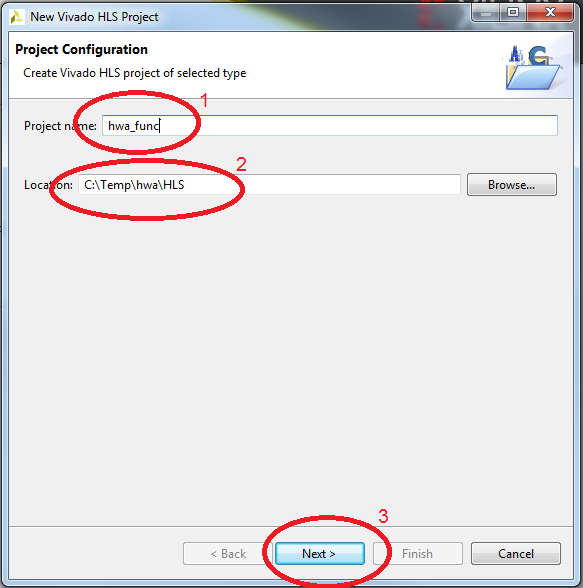
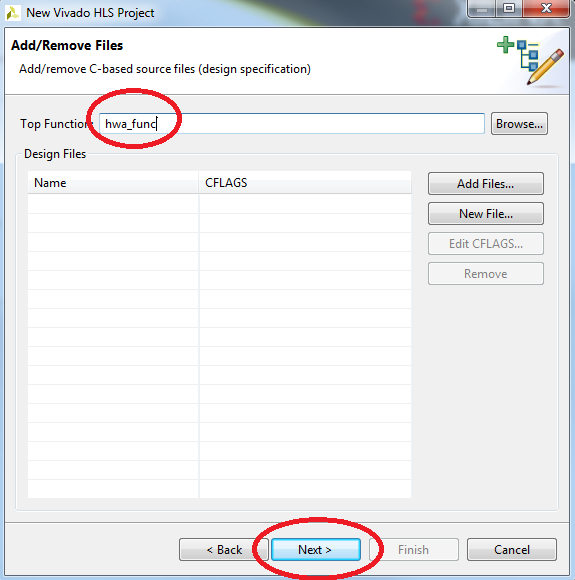
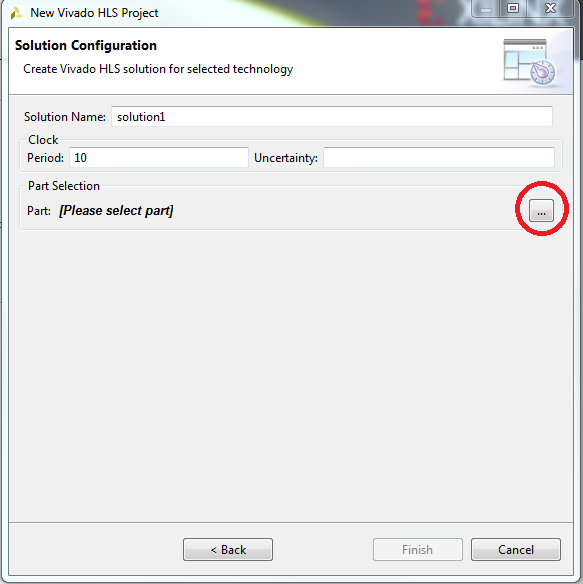
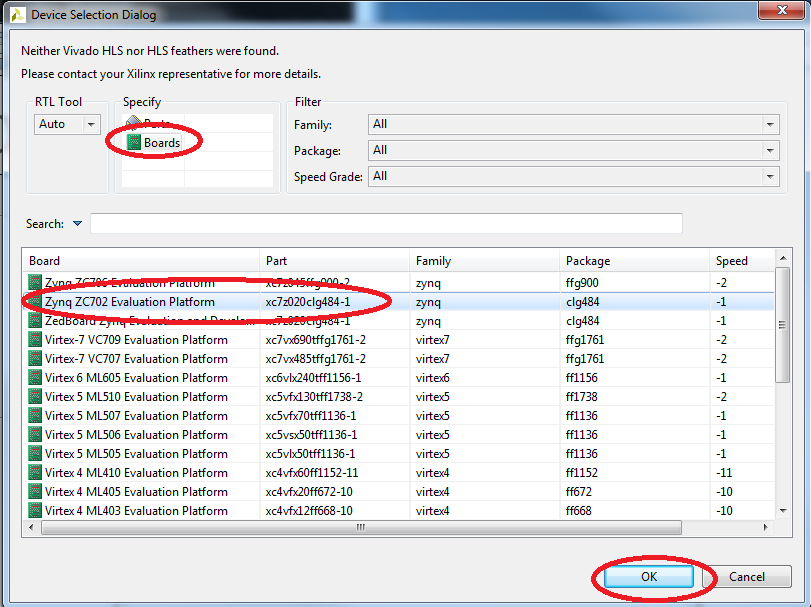
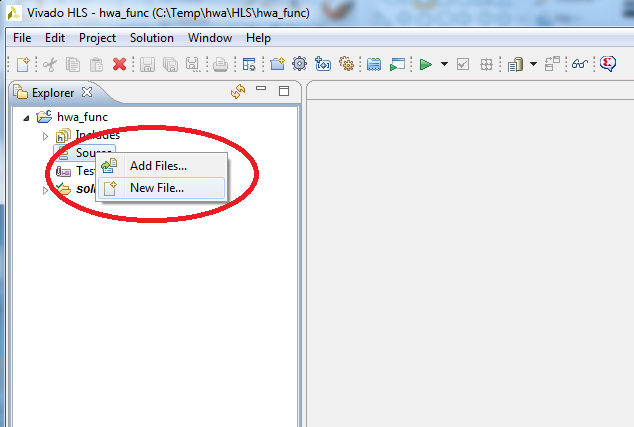
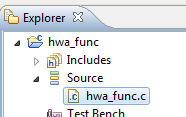
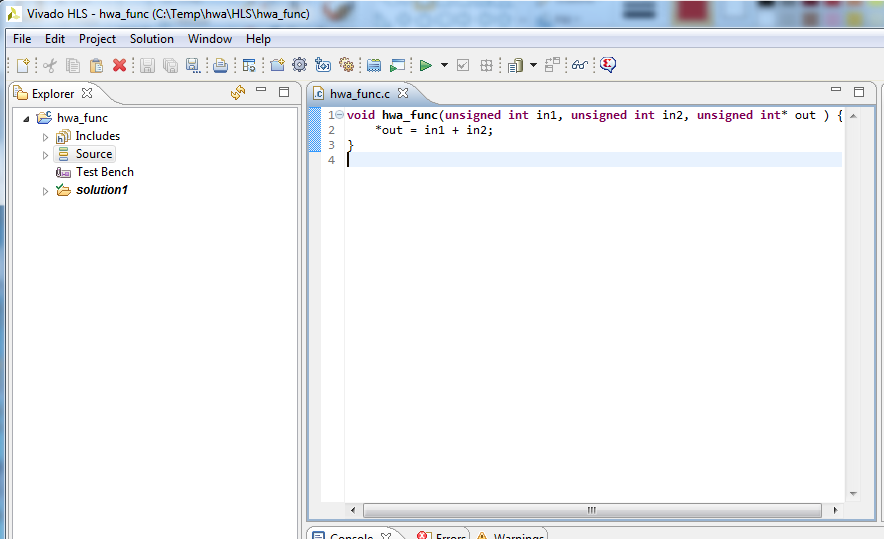
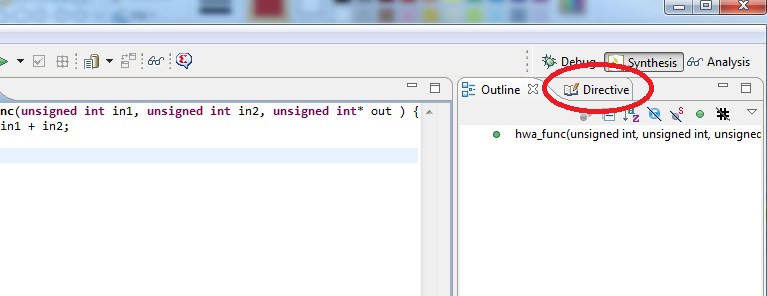
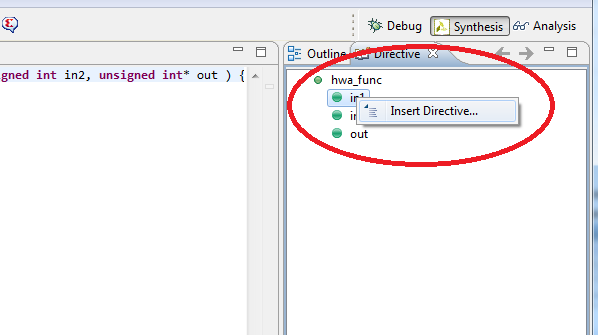
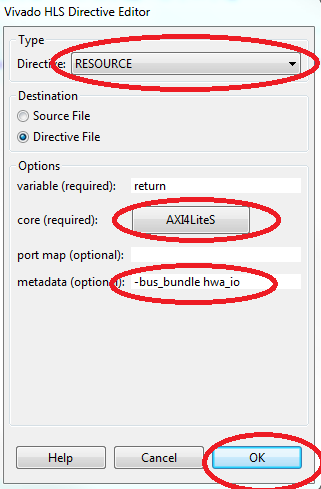
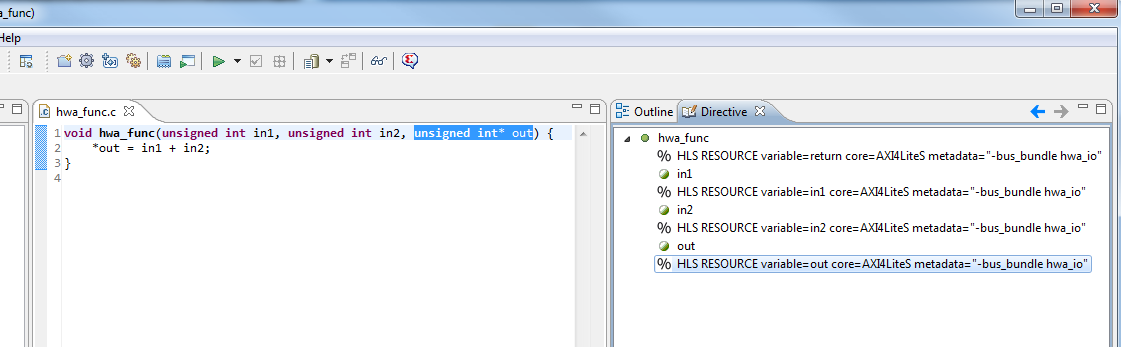
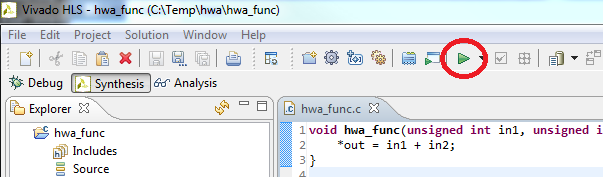
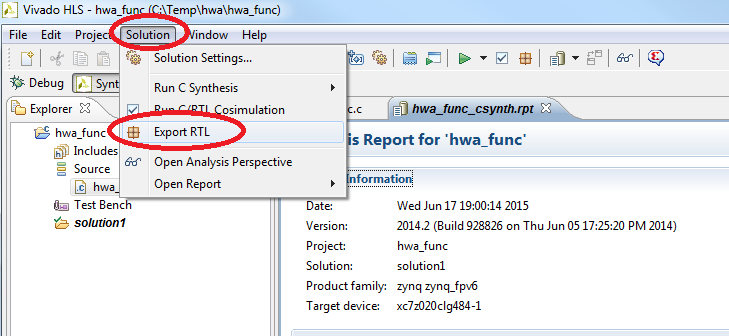
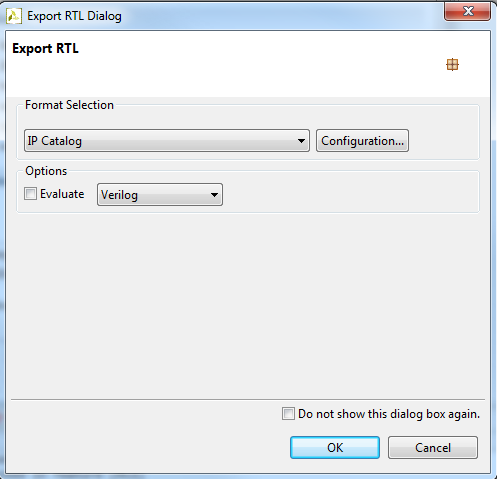
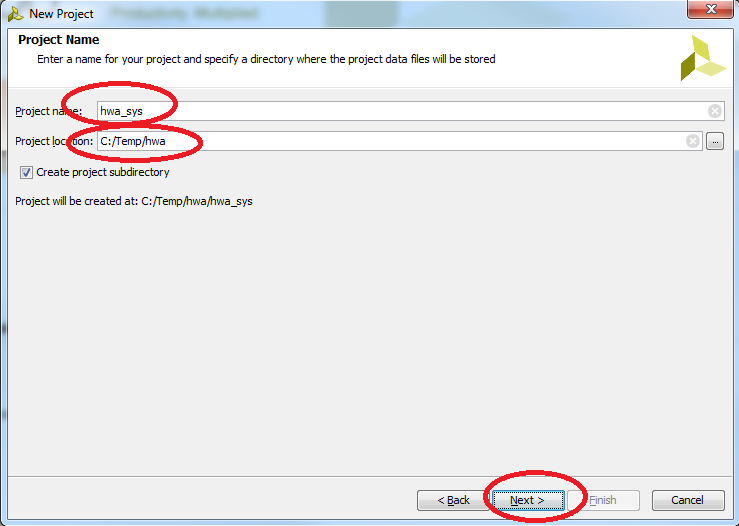
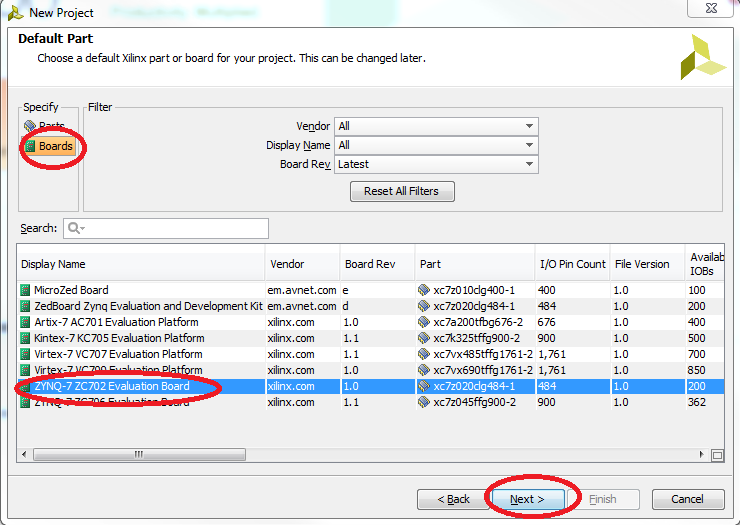
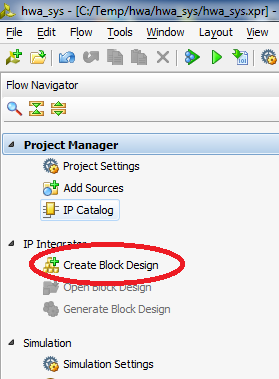
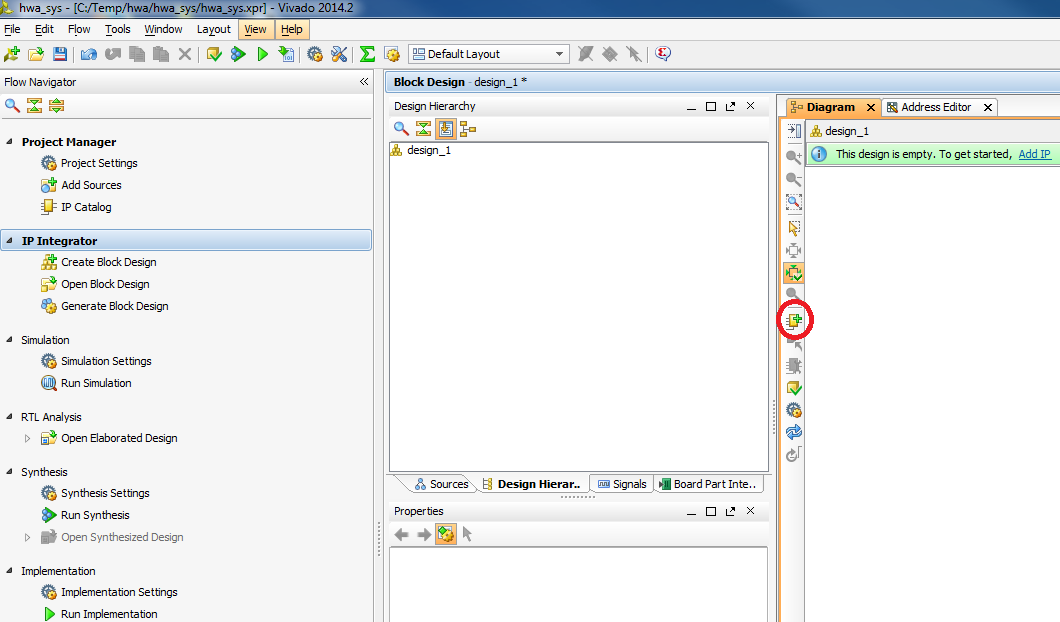
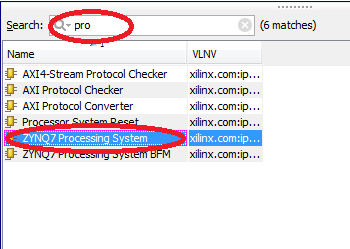
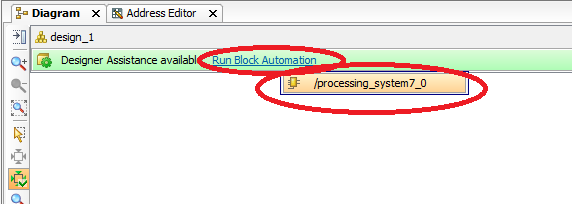
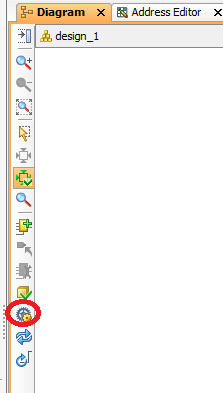
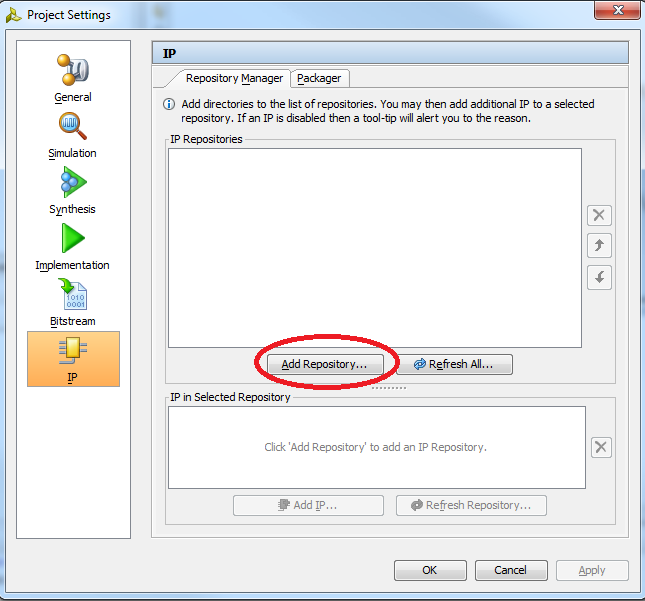
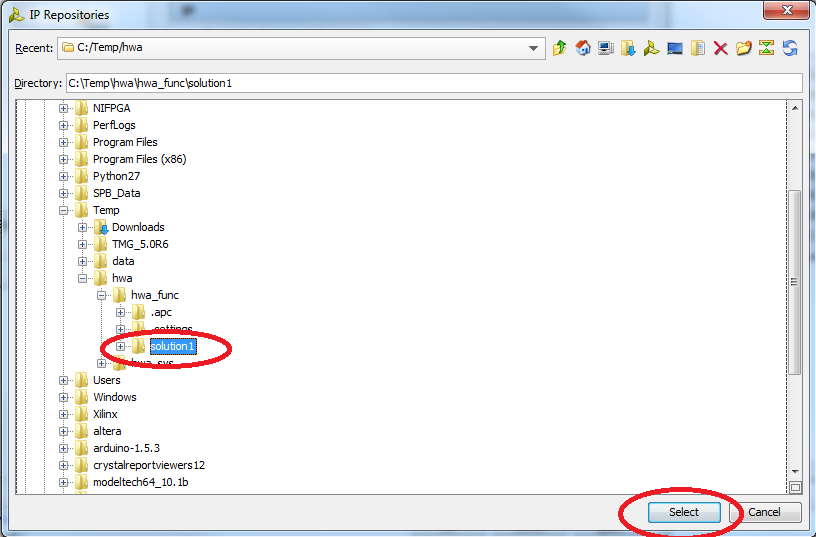
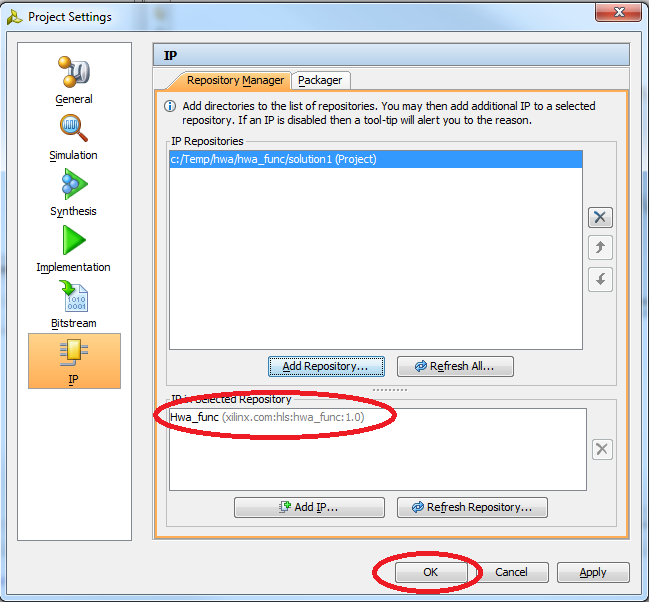
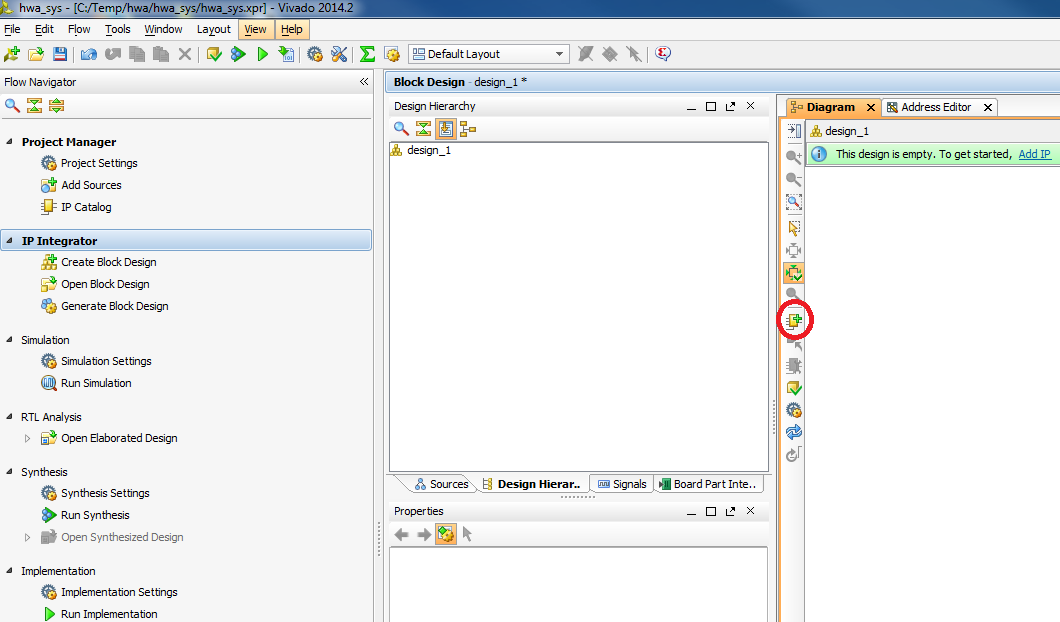
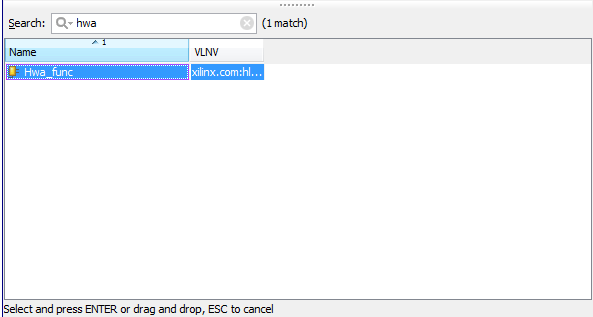
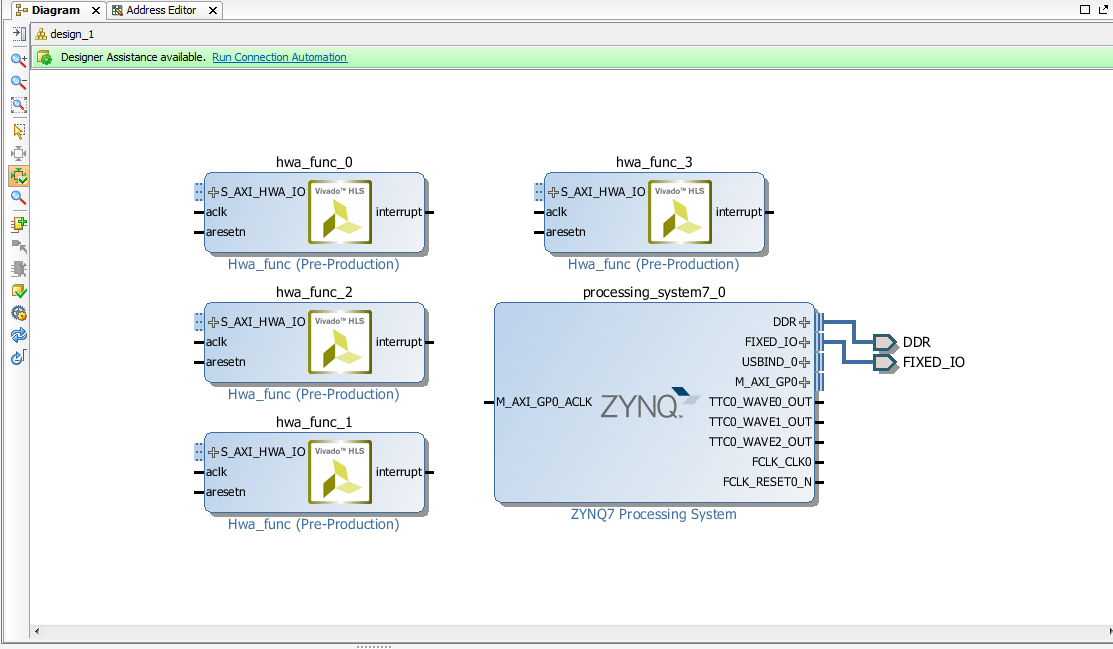
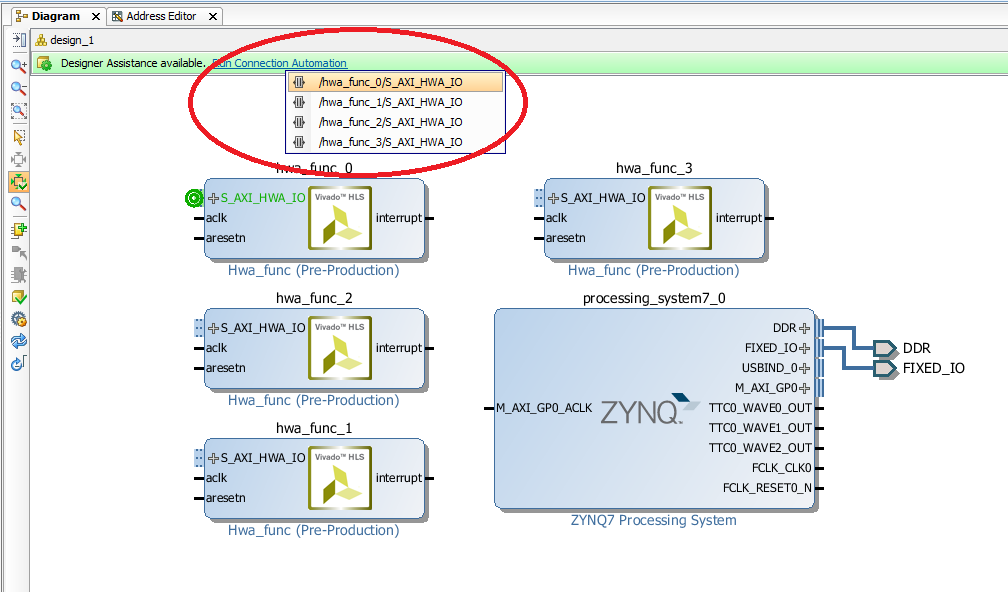
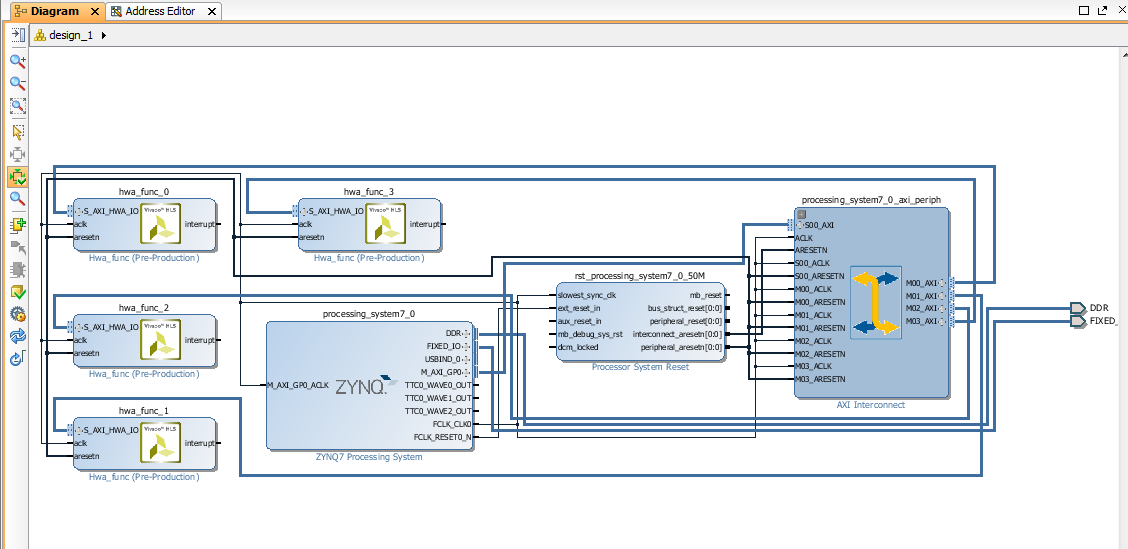
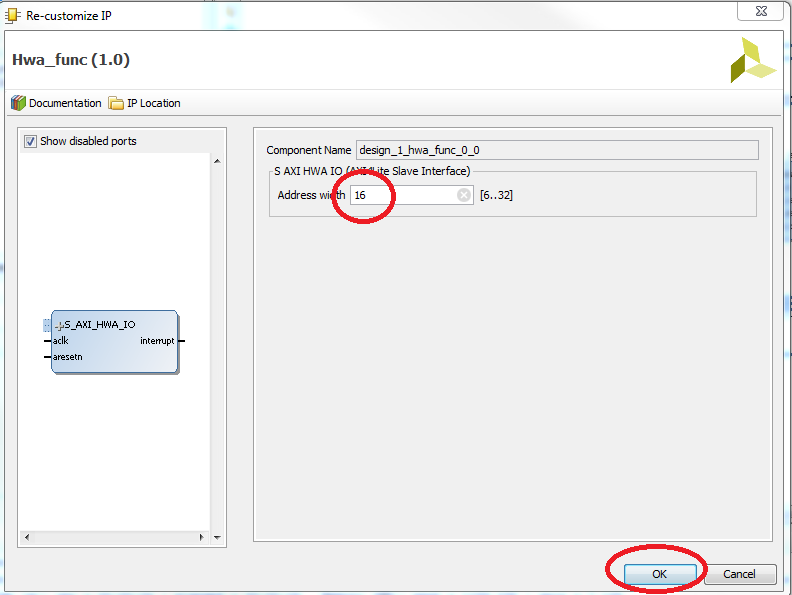
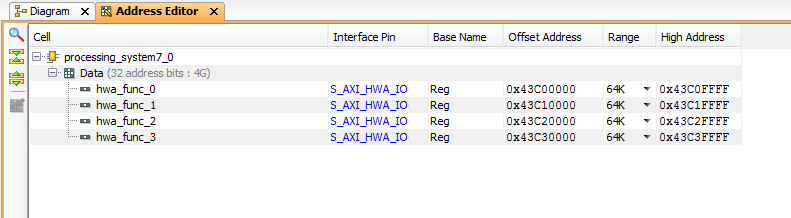
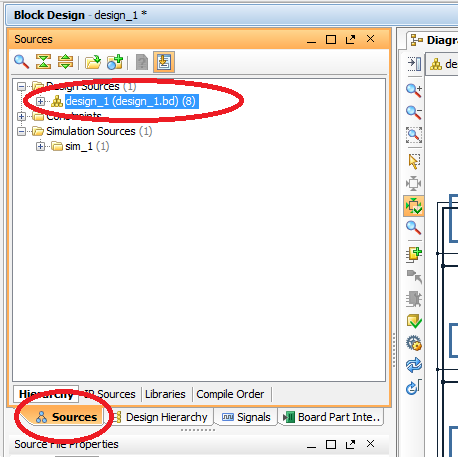
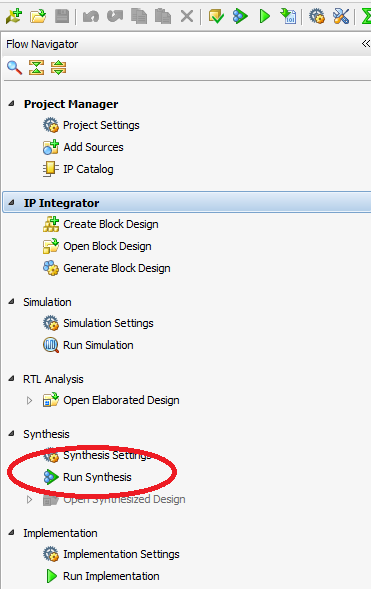
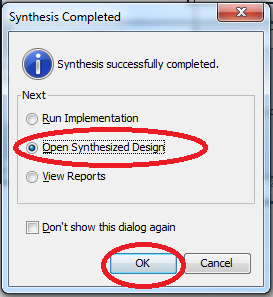
Make it yourself guide for HW DLL system based on Xilinx partially reconfigurable FPGA

First thing first, we will need to create a HW accelerated template that the base of the system will be loaded with. This will also be the template for creating new accelerators.

Note: all screenshots taken from Vivado version 2014.2

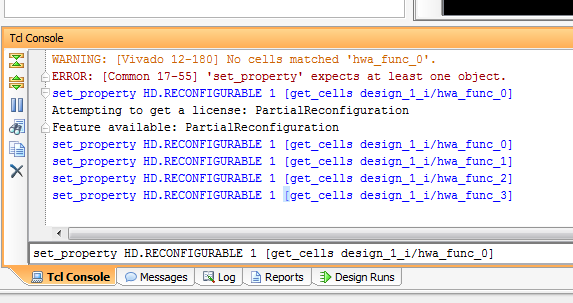
1. Start Vivado HLS and click create new project  
   
2. Enter project name (we choose "hwa\_func"), set a location and click next  
   
3. Write the top function (same as project name for us) and click next twice.  
   
4. Select the board:
   1. Click the board selection box  
      
   2. Click on boards and Zynq ZC702 from the menu and click ok.  
      
5. Click finish.
6. Now we need to add implementation, for this right click on source and select add new file…  
   
7. Select file name (we selected hwa\_func.c)
8. Edit the file by selecting source and double clicking on it  
   
9. Write your function:
   1. It should return void.
   2. All inputs are of type unsigned int
   3. All outputs are of type unsigned int\* (pointer to unsigned int)
   4. You can have up to 8190 I/O parameters  
      WORNING: you can't change the input parameters value in your function!  
      
10. Insert directives so that we can use this function as HW:
    1. Click the directive tab  
       
    2. Right click on a variable/main function and select insert directive.  
       
    3. Select RESOURCE in the directive dropdown menu.
    4. Select AXI4Lite in core by clicking the box next to core.
    5. Add "-bus\_bundle hwa\_io" in the metadata.  
       
    6. Repeat this for every parameter as well as for the main function, the end result should look like this:  
       
11. Run c synthesis by clicking the green arrow  
    
12. When the synthesis will finish it will open a synthesis report.
13. Now go to solution and click Export RTL  
    
14. Click OK.  
    
15. Wait until it's done and close Vivado HLS.

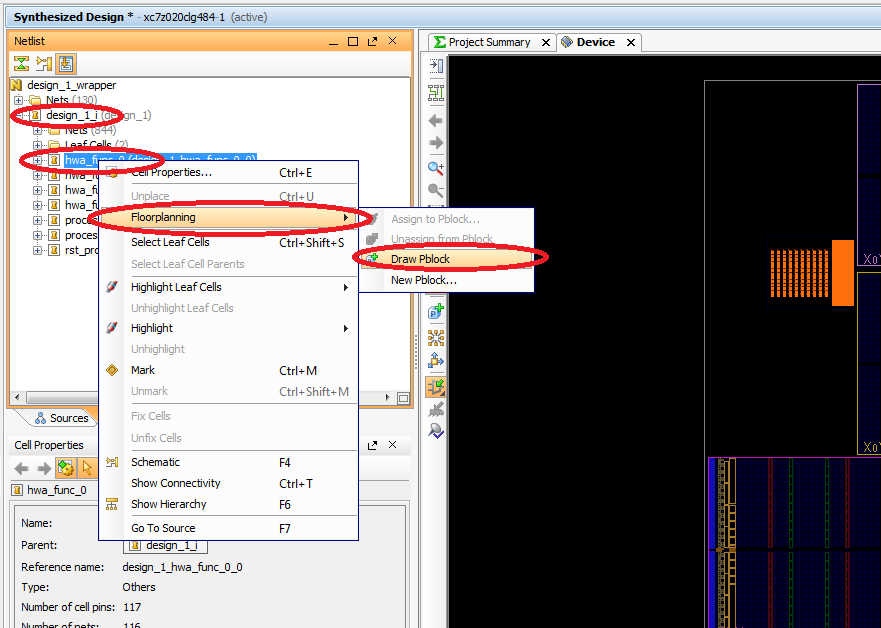
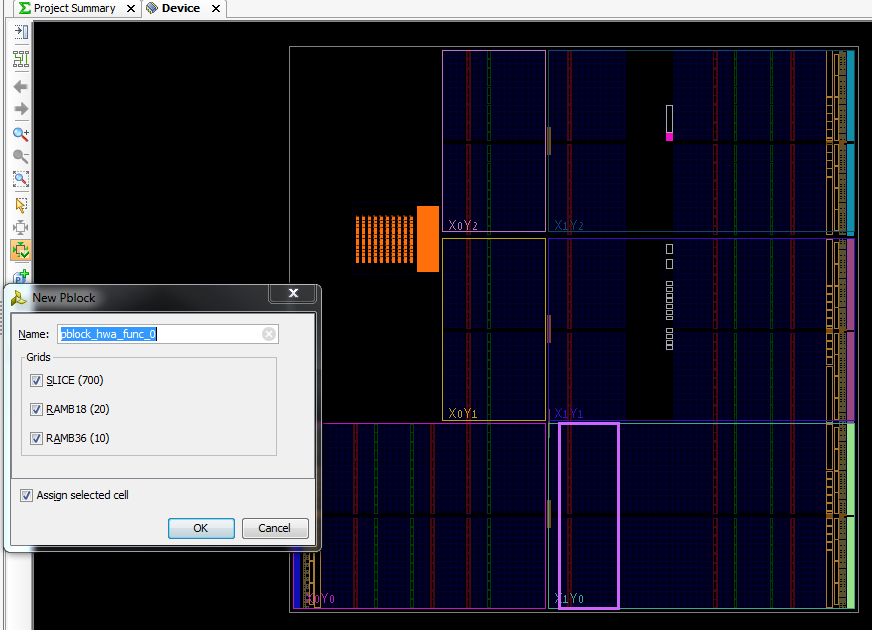
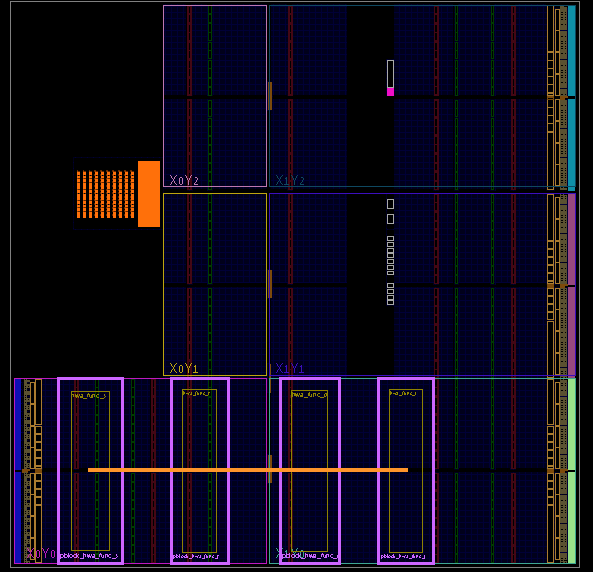
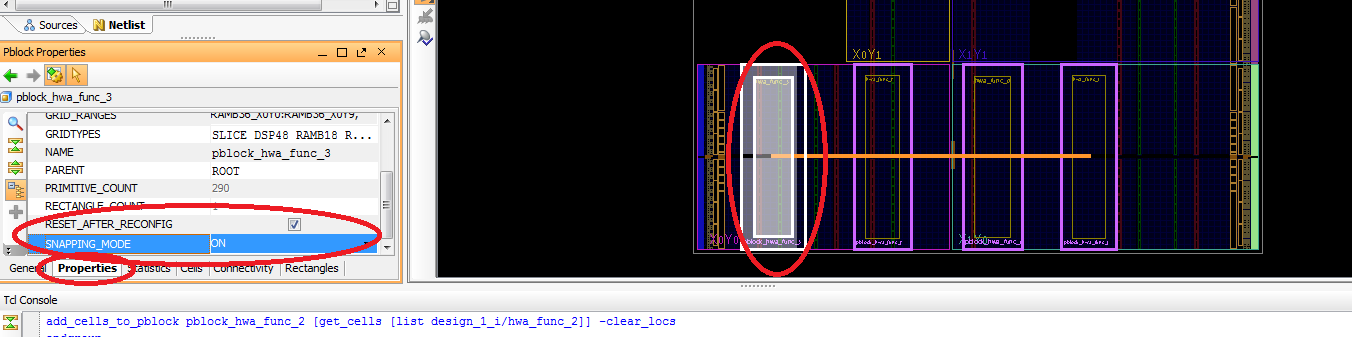
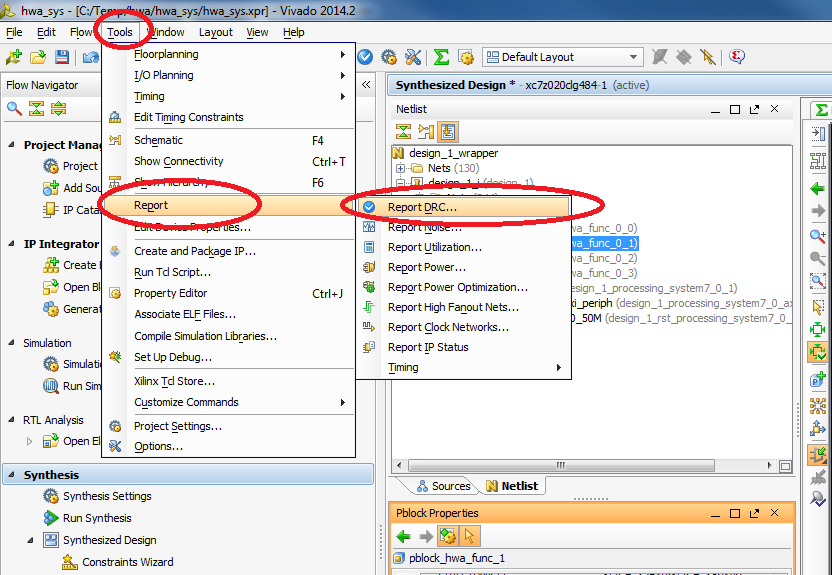
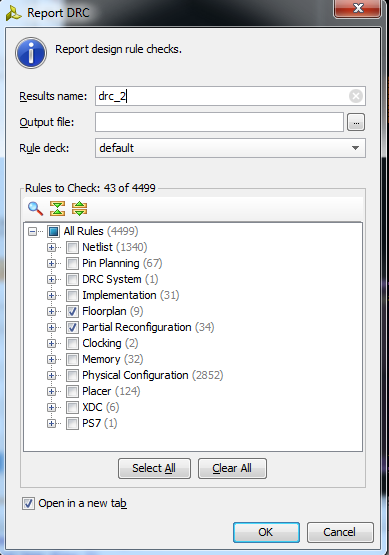
You now have all you need to start the design.

1. Run Vivado and click create a new project  
   
2. Click next.
3. Set project name to hwa\_sys, set project location and click next 5 times  
   
4. Select the board by clicking on boards and selecting the ZYNQ-7 ZC702, click next  
   
5. Click finish.
6. Click on create block design and clock OK  
   
7. Click "add IP" icon  
   
8. In the menu that opens write in the search bar "pro" and double click on ZYNQ7 processing system  
   
9. Click on the "run block automation" on the green bar and select the processing system and click OK.  
   
10. Now we need to add our HWA to the library.
    1. Click on the project settings icon  
       
    2. Click "add repository"  
       
    3. Navigate to the solution1 folder in your HLS project folder and click select  
       
    4. Make shore that you see the hwa\_func and click ok  
       
11. Add the HWA module as many times as you wish, every module is going to be reconfigurable so this determines the number of PR partitions:
    1. Click add ip icon  
       
    2. Write hwa in the search bar and double click the hwa\_func  
       
    3. You can copy paste from the block diagram view instead of adding the IP.
12. For this guide we will build only 4 PR blocks  
    
13. Run connection automation for every added block  
    
14. Final system:  
    
15. At the moment the modules have the minimum number of address pins needed to manage the 3 IO parameters we gave them, since we don’t want to restrict to this number of pins we need to set the address bus length to the maximum 16.
    1. Double click on every HLS module in the diagram view
    2. Set the address length to 16 and click ok.  
       
16. Note the addresses of all modules, you will need them when writing the software:
    1. Click address editor and extended processing\_system->data  
       
17. Click on source tab in sources.
18. Right click on design\_1.bd and select "create HDL wrapper"  
    
19. Click ok.
20. Click run synthesis  
    
21. When it's done, select open synthesized design and click ok in the popup window  
    
22. Set the hwa cells as HD.RECONFIGURABLE by running the next code in the TCL console line by line:  
    set\_property HD.RECONFIGURABLE 1 [get\_cells design\_1\_i/hwa\_func\_0]

set\_property HD.RECONFIGURABLE 1 [get\_cells design\_1\_i/hwa\_func\_1]

set\_property HD.RECONFIGURABLE 1 [get\_cells design\_1\_i/hwa\_func\_2]

set\_property HD.RECONFIGURABLE 1 [get\_cells design\_1\_i/hwa\_func\_3]  


1. Now for the floorplan, do this for every module:
   1. In the netlist view extend design\_1\_i
   2. Right click on the module name
   3. Select Floor planning
   4. Select draw pblock  
      
   5. On the layout you will see 6 clock domains, 4 big ones and 2 small ones.
   6. Draw a square in any clock domain, they should touch the top and the bottom of the domain and should not touch each other.  
      note the resources this pbloc has and click ok.  
      
   7. End result:  
      
2. Now we need to set the parameters needed for every PR blocks:
   1. Click on the block.
   2. Click on the properties tab in the pblock properties window
   3. Scroll down
   4. Set RESET\_AFTER\_RECONFIG checkbox
   5. Set snapping mode to "on"  
      
   6. Do this for every block
3. To check your design for errors go to tools->Report->report DRC  
   
4. Select only Floorplan and PR rules and click ok  
   
5. Run in TCL console:
   1. Opt\_design
   2. Place\_design
   3. Route\_design
6. Now that the design is complete we need to write a checkpoint, future designs will be compared to this checkpoint to see if the design is compatible with our PR configuration.  
   run in the TCL shell:  
   write\_checkpoint <project folder>/top\_route\_design.dcp
7. Now we need a static route checkpoint for future use:
   1. Run:  
      update\_design -cell design\_1\_i/hwa\_func\_0 -black\_box  
      update\_design -cell design\_1\_i/hwa\_func\_1 -black\_box  
      update\_design -cell design\_1\_i/hwa\_func\_2 -black\_box  
      update\_design -cell design\_1\_i/hwa\_func\_3 -black\_box
   2. Run:  
      lock\_design -level routing
   3. Run:  
      write\_checkpoint <project folder>/static\_route\_design.dcp  
      to make the checkpoint.
8. Let's read back the cell checkpoint to build the bitstreams:
   1. Go to file->open checkpoint
   2. Select the top\_route\_design.dcp file created before
   3. Close the previous design without saving it.
   4. Run:  
      write\_bitstream <project folder>/bitstram.bit

There will be 5 bitstreams created, the main one that has the entire system and one bit stream for every pblock.

To make an SD card and boot with your design follow the generating boot guide.