DIMEtalk System Design – A Walk Through

Karthik Nagaraja (September 9, 2007)

There are three major components that need to be taken care of for porting an application on the Nallatech boards

- The user defined HDL file performing the operation of the core (e.g., FIR, Matrix multiply, etc…)
- The GUI-based system design in DIMEtalk
- The software control program written in C at the host end

Let us look at a stepwise procedure for getting a sample application work on the Nallatech board.

1. Create the user defined HDL file in your environment of convenience (say top_user.vhd). Note: Use dt_clk and reset to name the clock and reset ports in your designs.
2. Open up DIMEtalk
    a. Go to View → Library Manager → Add to Library → VHDL Component – Browse to top_user.vhd and click open. The user defined HDL file is now added as a part of the DIMEtalk component library (usually under the user components tab) and ready to be used as a "drag and drop" component in any design
    b. Right click on the design space (the white space area), click Create → Device. Select the H101-PCIXM board. Choose V4LX100 f1148 -10 family of FPGAs (it should be the default choice). NOTE: ALL THE BLOCKS YOU ADD TO THE DESIGN SHOULD BE BOUNDED WITHIN THE DEVICE AREA. SO, DRAG THE DEVICE AREA TO ENCOMPASS ALL THE COMPONENTS (Ref Fig.1)
    c. Adding Components – Left click on the desired component and left click again on the design area to add the component.
        i. Two important components that are to be present in every design are the clock (Click on systems tab and you will find H100 clock component) and PCI-X edge (Click on Edge tab and use PCI-X H100 component). Connect the PCIXInterface clock node on the PCI-X H100 component to the PCIXInterface clock node on the H100 clock component – Refer Fig.2

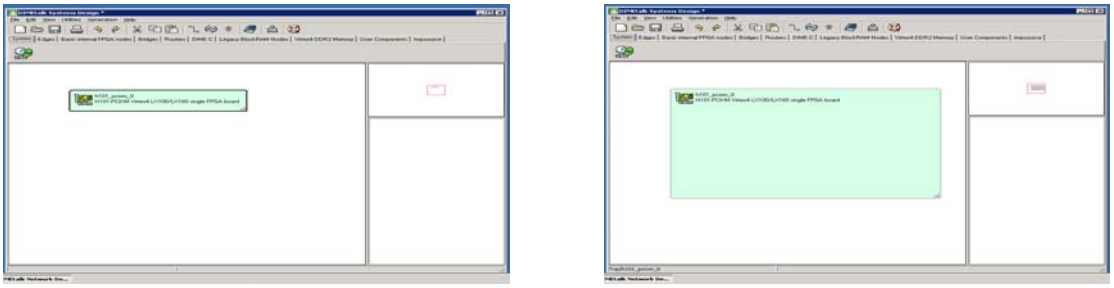

Fig.1 Expand device area and make sure all the design components are present inside the green box.

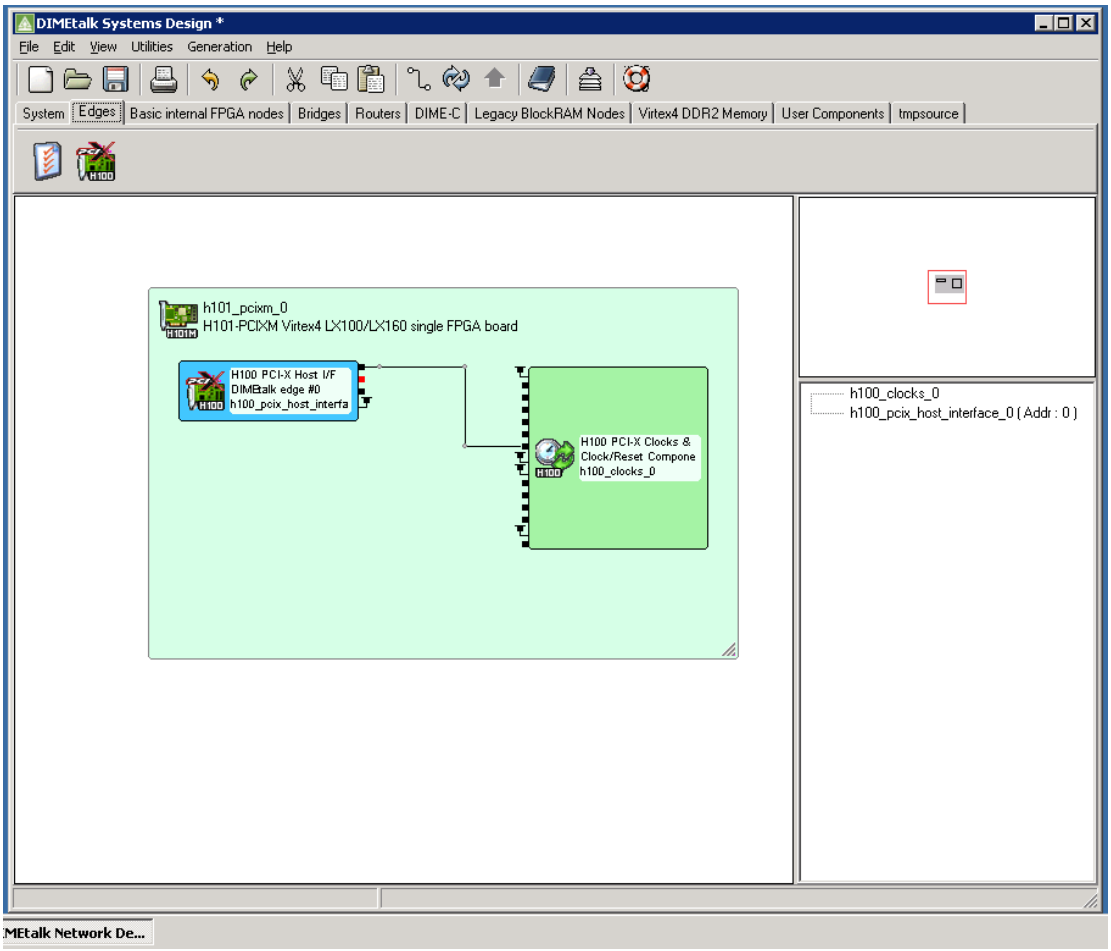

Fig.2 Connect clock node of PCI-X H100 edge component to the PCIXInterface Clock node of the H100 clock component

d. Other components that find use in most designs are the routers (under routers tab), BRAMs and memory map (under Basic internal FPGA nodes) – Usage of BRAMs in DIMEtalk depends on whether the user core (top_user.vhd) instantiates BRAMs as a component in its architecture. In general, you could define and use BRAMs as part of DIMEtalk and have ports on the top_use.vhd file going to the BRAMs or have the BRAMs defined within top_user.vhd. The former might be a better option in most designs.
    i. Routers are always used to fan out the single red port on the PCI-X edge component to multiple BRAMs and other components.
    ii. After a “drag and drop” operation on BRAMs, we can right click on the component and click “edit” to change some properties of the BRAMs. The primary thing that we would change is the No. of address lines. The default would be 12 (which means 4 bit words are used in each BRAM with a total of 2^12 addressable spaces). For 32 bit words, set No. of address lines to 9.

iii. NOTE THAT EVERY BRAM AND MEMORY MAP ADDED TO THE DESIGN IS GIVEN A **NODE NUMBER**. This number would have to be used later in the software control program at the host end to send and receive data.

e. Once the design has been created (refer Fig.3 for a sample design – This design basically lets one write and read from a BRAM), save the design and click on the Generate network code button. DIMEtalk has scripts within that call ISE and synthesizes, does P and R, applies constraints and gives you the final "bit file". This is the only thing that would be required for later use. The bit stream would be located under source→h101_pcix_m folder

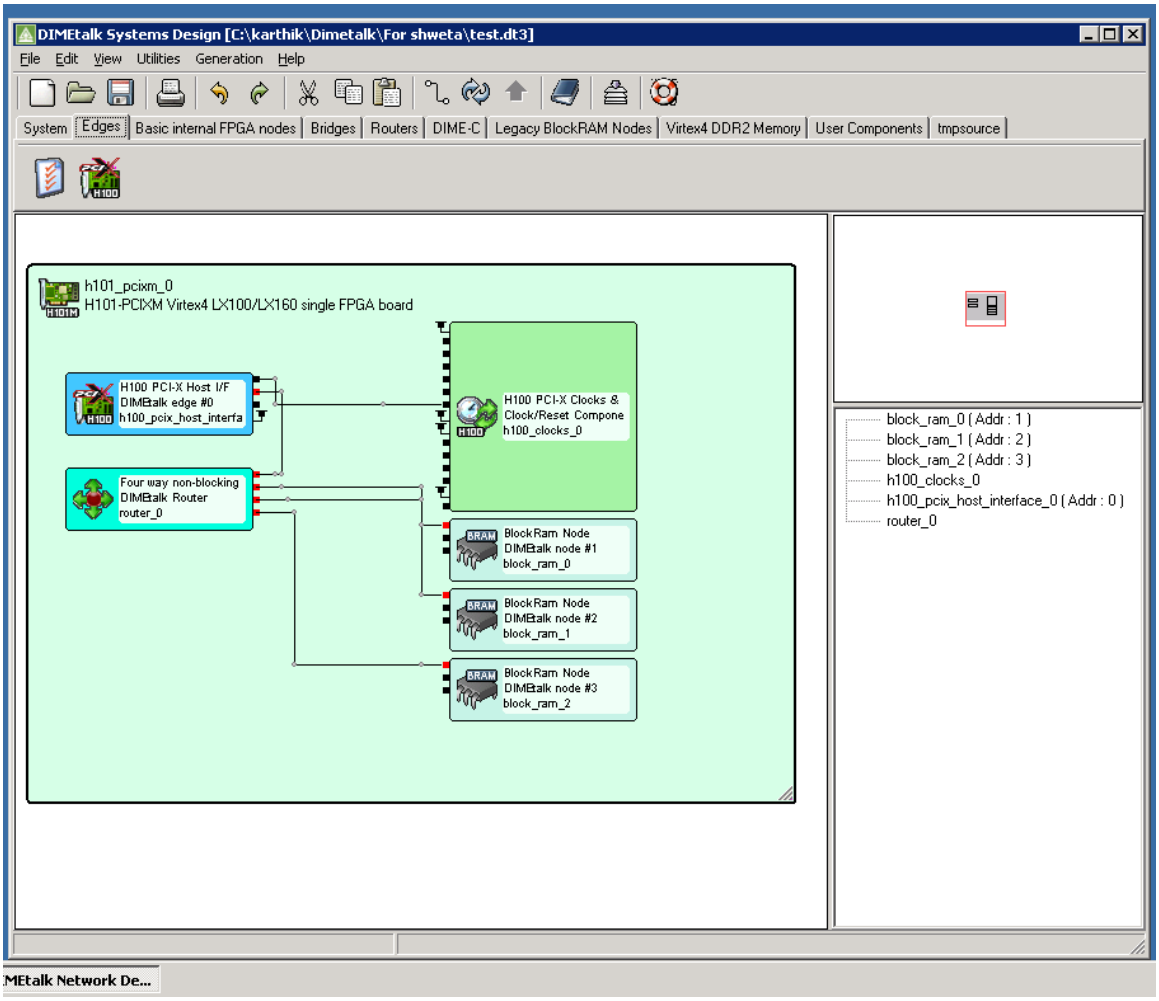

Fig. 3 Sample DIMEtalk design for reading/writing to BRAMs

3. SSH to delta.hcs.ufl.edu – the hcs account username and password should work
   a. I have attached a zip file with some header files and makefile for compiling the software control program. Extract them to a folder
   b. Copy the bit file that you generated earlier to this local folder.

c. Open up example.c and you could change the FPGA design clocks (These are the CLKA, CLKB and CLKC variables). For synchronous designs, CLKA is used by default.
d. Scroll down to the “USER CODE BEGINS HERE” and make appropriate changes. The current example.c file writes and reads from a BRAM. It should be self explanatory
e. At the command prompt, just type in “make” to compile the c code and generate the executable h100bist.exe
f. Type in “srun ./h100bist.exe” to run the code