

Re: Memory map?

8000-----
UART
800F-----
8010-----
ROM
FFFF-----

The UART does not know that its address actually begins at 0x8000 – you will be required to have some logic which looks at the requested address on the address bus and decides that that address falls within the range of the UART and so the appropriate control signals to the UART should be asserted to access it. This is made simpler in an FPGA because you have plenty of logic available to do this.

I have FPGA with a processor. Now i want to connect this processor with Custom IP (on FPGA) using Processor Local Bus. I know custom IP has bunch of registers which will receive the data and send the data. These registers i will create and to my understanding they are in peripheral. My question is

Are these registers really physically there or they are just the map. SO when Processor will want to write to one of them it will write to the memory and Peripheral will know something is written to it?

I'm not sure but I think your problem is in directing the processor to your registers. The processor will have its own idea of how its memory map is laid out and you need to fit in around that. If, when you run your program and it wants to read from one of your custom IP registers, you need to make sure that your address decode logic provides that register's contents on the data bus when this address is requested. How that happens is up to you but if you put some thought into where in the processor's addressable memory space you want it to be it should be some fairly simple logic expression.

I'm not sure about what you mean with "Are these registers really physically there or they are just the map" but I think that they must be physically there but you get to them by accessing their place in the map. I.E. on my processor the first location in flash is at 0x44000000. If I read from that then the address decode logic knows I want to access flash and will wiggle the appropriate lines to get the flash to return its first word of data (address 0x00000000 according to the flash).