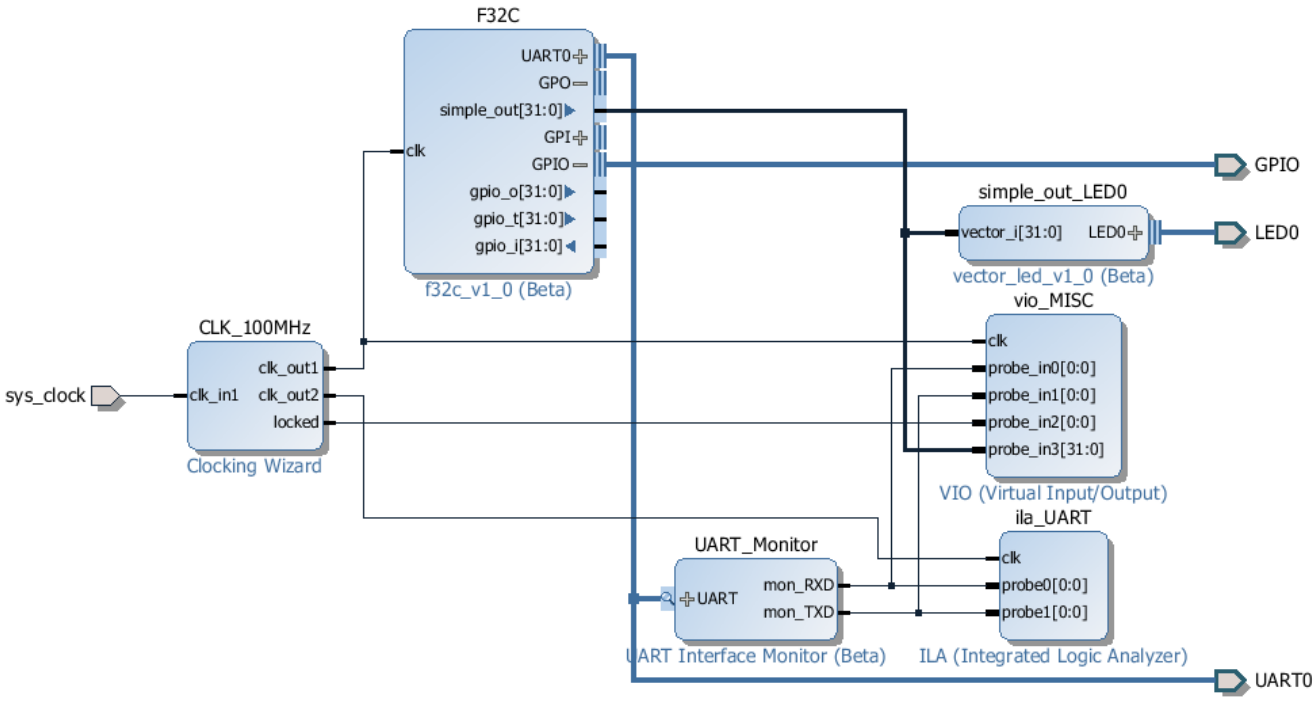
F32C Vivado IPI Support

F32C BRAM SoC

This is mainly IP Core wrap around "bram glue" generic F32C SoC RTL, Changes to original RTL include change of the GPIO IP Core to export the tri-state bus to top-level as 3 signals, no inout ports used any more.



Getting this Design ported to any new board usually takes less than 5 minutes. So far all newly ported design for new boards have worked first time tested with 0 time wasted in debug or troubleshooting.

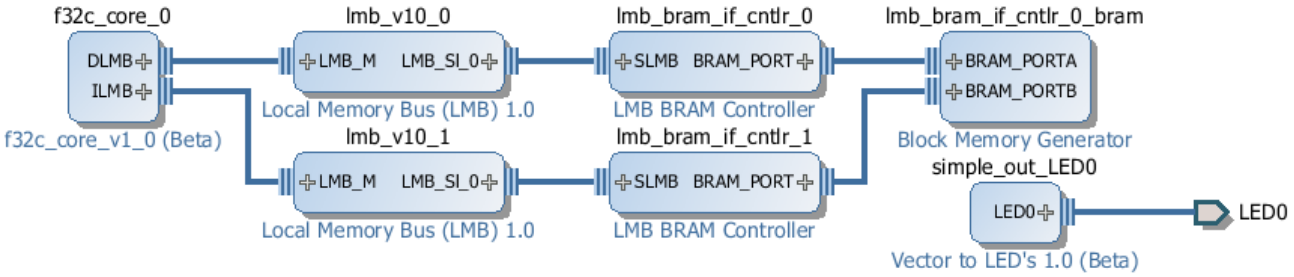
Ready to use project and bitstreams are available from the LED Blinky Tutorial.

F32C Core

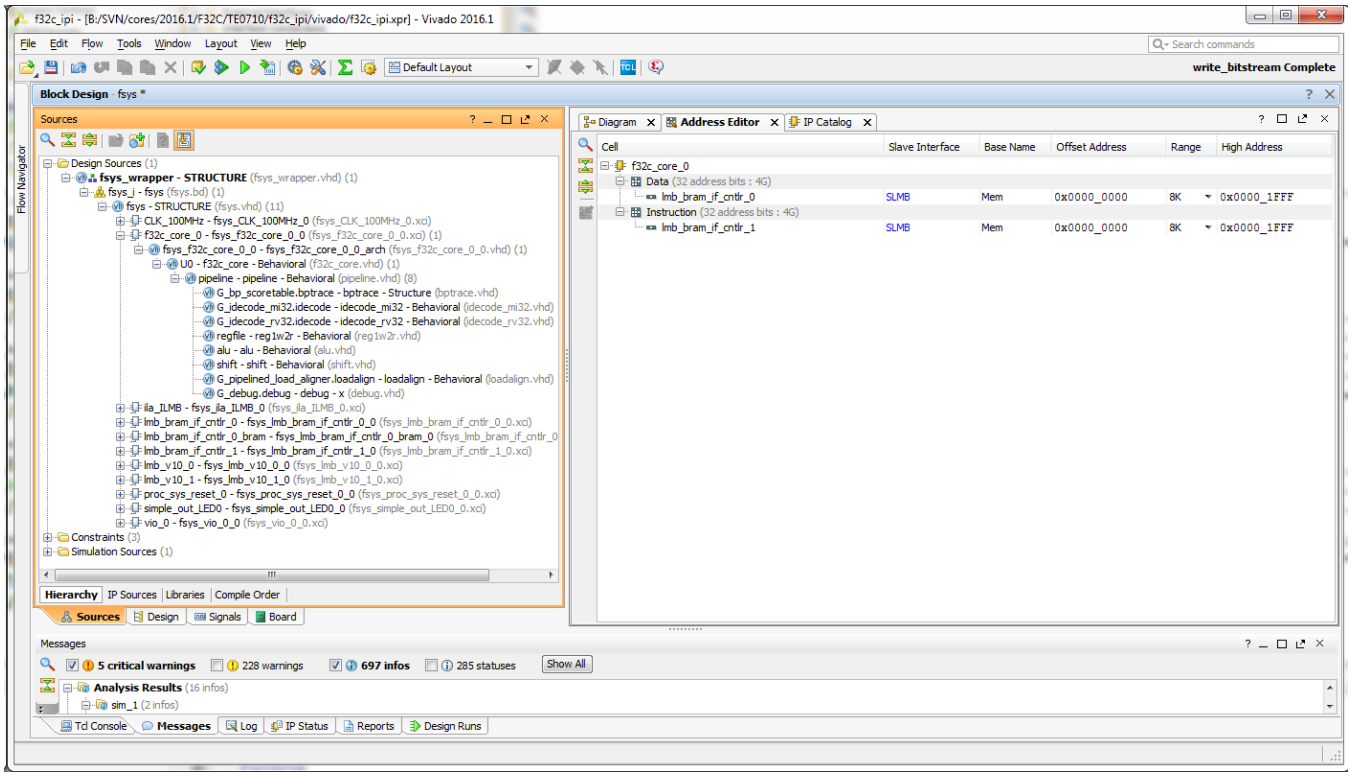
This is work to convert F32C into fully usable IP Catalog Processor IP Core.

LMB Bus

LMB Bus is the simplest bus available in Vivado IP Catalog, adding it to the "wrap" around F32C Core is fairly simple.

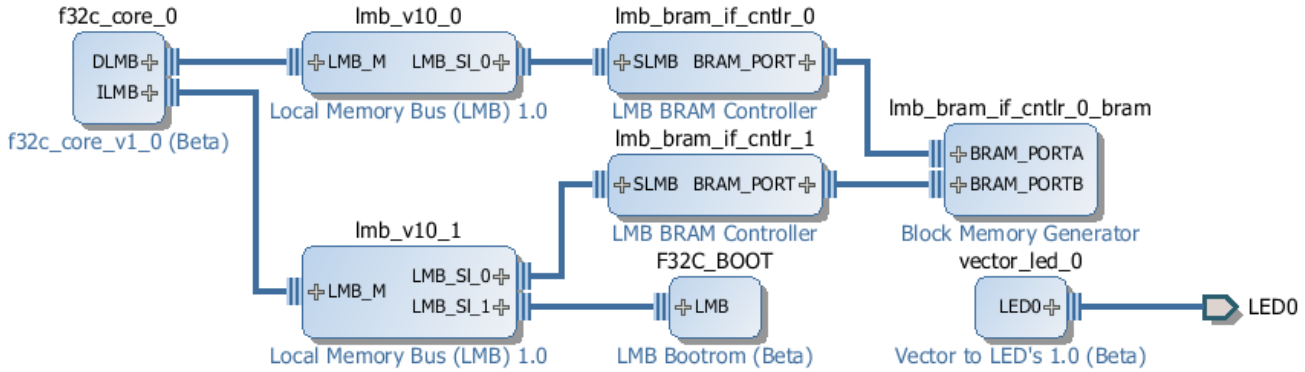


Data and Instruction buses are exported as LMB Bus, then IPI Standard IP cores are used to add BRAM. The size of the BRAM can be selected in the Address Editor:

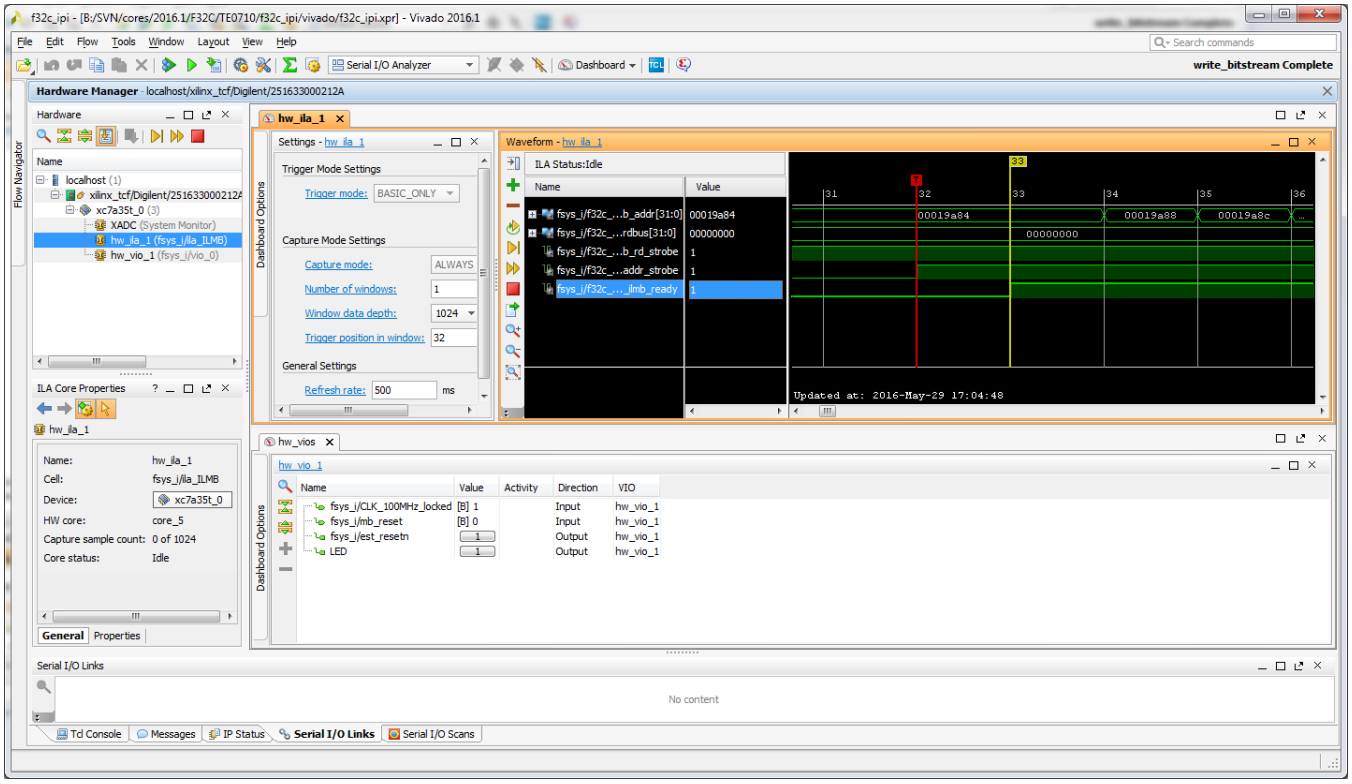


Setting F32C BRAM size in Vivado IPI Address Editor.

Problem: it is not possible to assign an ELF files to the BRAM that we add in such way not directly in Vivado IPI. As workaround several options exist, we can create an BOOTROM IP Core that can be connected to the LMB Bus holding the bootcode or we can create a BOOSTRAP IP Core that writes the bootrom code into the main BRAM and then releases F32C reset. Of course it is possible to merge the ELF into bitstream with console commands or TCL scripting.

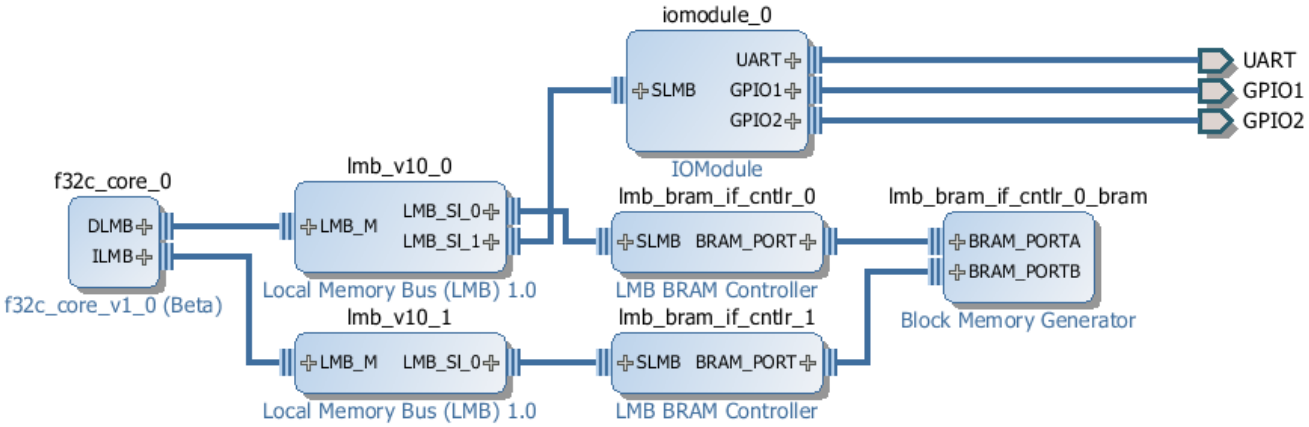


LMB Bootrom IP Core added with fixed code for the bootstrap.



Debug session in Vivado Labtools, when CPU Reset is asserted Instruction address bus freezes, but after release there seems to be no reset, F32C Core tries to continue from last PC Address.

WIP to be figured out why.



Example F32C SoC using IOModule from Vivado IP Catalog, again the addresses for BRAM and the IOModule are assigned automatically and can be changed as needed in Address Editor.

AXI Bus