RISC – V RV 32 Vivado IP integration user guide

This guide explains the process of performing design rule apropriate system integration for the RISC – V RV 32 IP in the AMD Vivado design suite.

The RISC – V soft procesor IP comes as a source code included non enrypted packade Vivado IP integrator ready IP.

To start designing with the core the user must include a new IP repository in to the Vivado design via the IP integrator and position the path to the RISC-V ip location.

The core comes with predefined Vivado friendly interfaces for connecting with other systems in the block design

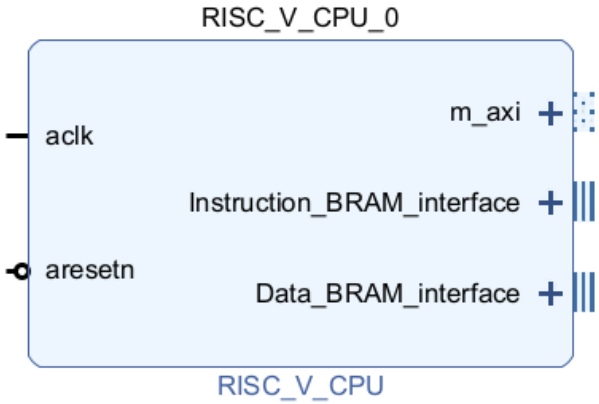

FIGURE 1

The core as shown in Figure1 implements single ended inputs for clocking and resets, and has to be provided with a 100 MHZ clock at the aclk port and a active low reset signal connected to the aresetn input. That is best implemented with a clocking wizard and a procesor system reset as shown in Figure2

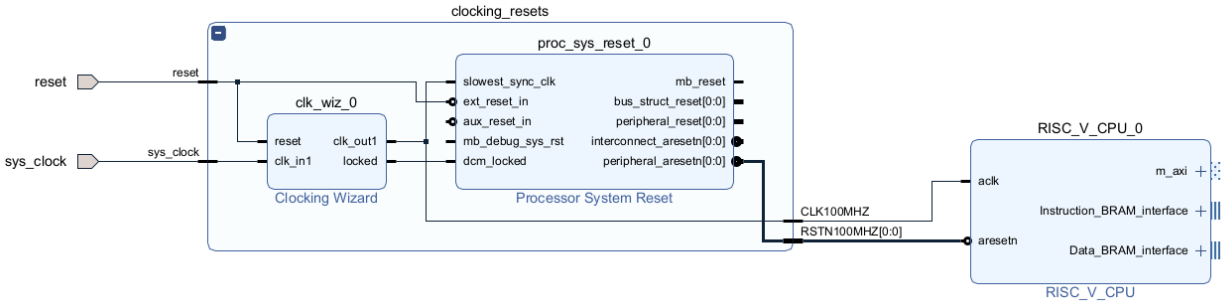

FIGURE 2

The core must be provided with local instruction and data memory in order to execute code. The core itself doesn't instantiate memory resources, but it has local memory interfaces packaged to comply with Vivado BRAM interface and provides one for Instructions and one for Data.
The core must be provided with a BRAM IP and that is best done via the Block Memory Generator IP that is a part of the basic IP catalogue.

```
set blk_mem_gen_0 [ create_bd_cell -type ip -vlnv xilinx.com:ip:blk_mem_gen:8.4 blk_mem_gen_0 ]
set_property -dict [list \
  CONFIG.Assume_Synchronous_Clk {false} \
  CONFIG.Byte_Size {8} \
  CONFIG.Coe_File {Path to .coe file} \
  CONFIG.Enable_A {Always_Enabled} \
  CONFIG.Fill_Remaining_Memory_Locations {true} \
  CONFIG.Load_Init_File {true} \
  CONFIG.Memory_Type {True_Dual_Port_RAM} \
  CONFIG.Read_Width_A {32} \
  CONFIG.Register_PortA_Output_of_Memory_Primitives {false} \
  CONFIG.Register_PortB_Output_of_Memory_Primitives {false} \
  CONFIG.Use_Byte_Write_Enable {true} \
  CONFIG.Write_Width_A {32} \
  CONFIG.use_bram_block {Stand_Alone} \
] $blk_mem_gen_0
```

The code snippet offers an easy copy paste in the Vivado tcl console to generate the block ram IP with the
IMPORTANT!!! The Path to .coe file part marked with red must be updated to the absolute path of a .coe file implementing targeted machine code

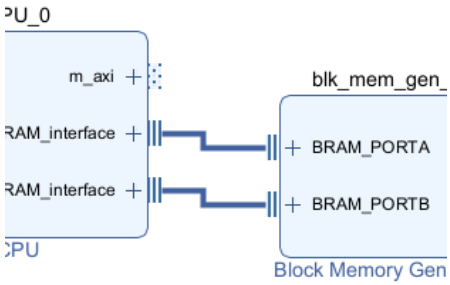

FIGURE 3

The following connections as shown in Figure3 must be done to connect the BRAM.

The core implements a AXI4-Lite master port for data access to axi peripherals.
To enable proper use of this peripheral interface the user must first configure the program memory range in the RISC_V_CPU IP by opening its customization GUI.

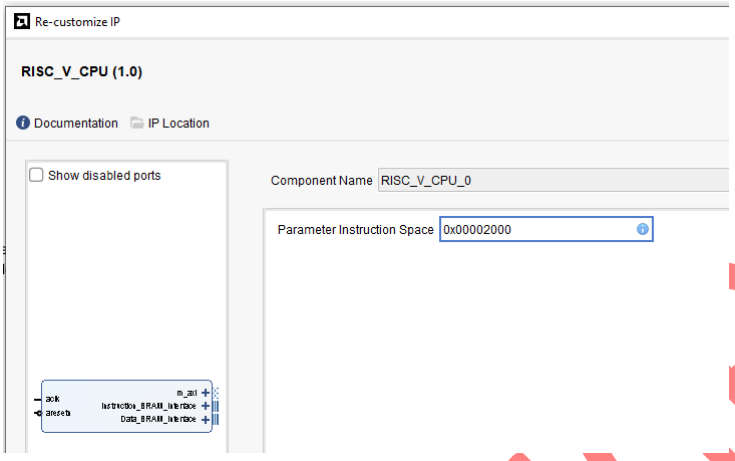

FIGURE 4

In this parameter a range of memory locations is give to the processor to successfully decode between the native data port and the AXI interface, its important to keep this size equivalent to the size of the program memory configured in the Block Memory Generator. As shown in Figure

Configuration of AXI peripherals follows the standard;
master->interconnect->subordinate scheme and Vivado offers many utilities to aid in the design of AXI peripheral systems.

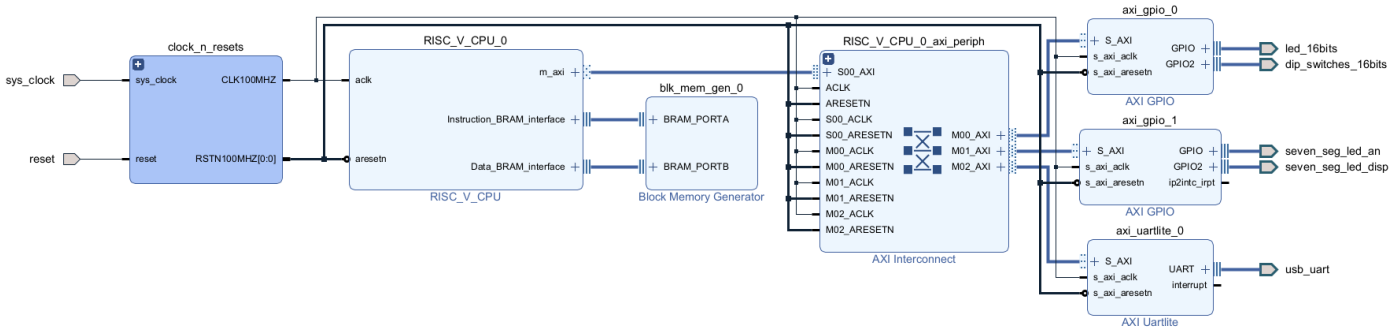

FIGURE 5

Figure5 shows an example design for integrating the RISC-V core in the Digilent Basys3 FPGA board and implementing the UART, test leds, switches and seven segment displays