Vexriscv SoC with UART & Multiple RAMs

This example integrates 2 AXI based rams onto a Vexriscv SoC.

Instructions:

Copy your demo folder from litex installation directory `litex/litex/soc/software/demo` and paste it inside your project directory. Use the main.c file provided in this example in the demo application and replace it with the main.c file located inside your newly copied demo folder in project directory.

1. Simulation

We can simulate this IP using litex_sim tool in litex.

Run the following command to generate your SoC:

```
litex_sim --integrated-main-ram-size=0x10000 --cpu-type vexriscv --ram-
init=demo.bin --sim-debug --multiaxiram
```

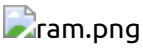
Run the following command to generate .bin file from .py file:

```
python3 ./demo/demo.py --build-path=build/sim
```

Before running the simulation, you have to create the binary of your application code residing in demo. The python script below converts the application code to demo.bin, which is later loaded on to the ram.

```
litex_sim --integrated-main-ram-size=0x10000 --cpu-type vexriscv --ram-
init=demo.bin --sim-debug --multiaxiram
```

Output:


ram.png

2. Hardware

Connect your Digilent Arty a7-100 board with your machine. We will be using the same design which we used in simulation to verify on the board. The following board file written in python creates the same SoC and later build and load it onto the Arty board.

Note: Before using this GPIO design on hardware, we need to replace the board files with the necessary changes provided in the `board files` directory.

Run the following command to generate your SoC:

```
./../../litex_installation/litex-
boards/litex_boards/targets/digilent_arty.py --integrated-main-ram-
size=0x10000 --variant=a7-100 --cpu-type=vexriscv --multiaxiram --build --
load --uart-name=serial
```

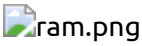
Run the following command to generate .bin file from .py file:

```
python3 ./demo/demo.py --build-path=build/digilent_arty
```

The litex_term tool load the board with the application binary through the comm port.

```
litex_term /dev/ttyUSB1 --kernel=demo.bin
```

Output:


ram.png

Application

This test invokes both memories and writes and read dat subsequently and randomly, inorder to make sure that the integration is properly done. Secondly, it stresses the Interconnect as both slave IPs are being communicated with simultaneously.