Caravel FPGA Labs

3 Labs to experiment with Verilog & HLS

4/6/2023

Jiin Lai

3 Labs

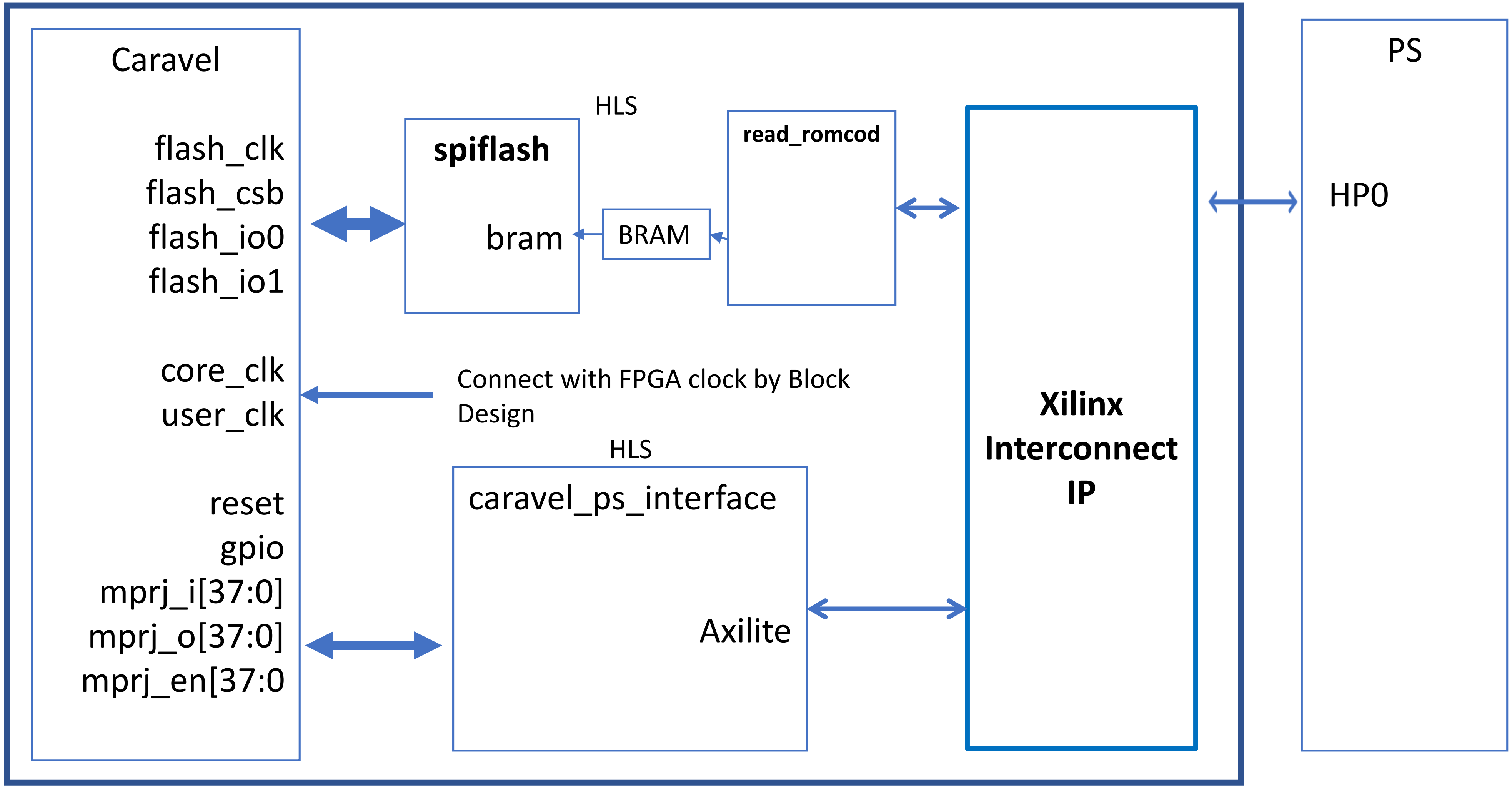
- Lab1: AXI master read/write BRAM
- Lab2 – spiflash protocol design and validation
- Lab1+2 – integrate Lab1 and Lab2
- Lab3 – Axilite access GPIO pins

Note: None of the design sources are verified. You have to verify it by yourself.

- Background
  - Vitis HLS – lab#1, 2
  - Xilinx XSIM
  - Basic Verilog & HLS coding

Design sources: https://drive.google.com/drive/folders/15WaEzAkgZPE97dyX6pHcDiDTv6b-hRhJ?usp=sharing

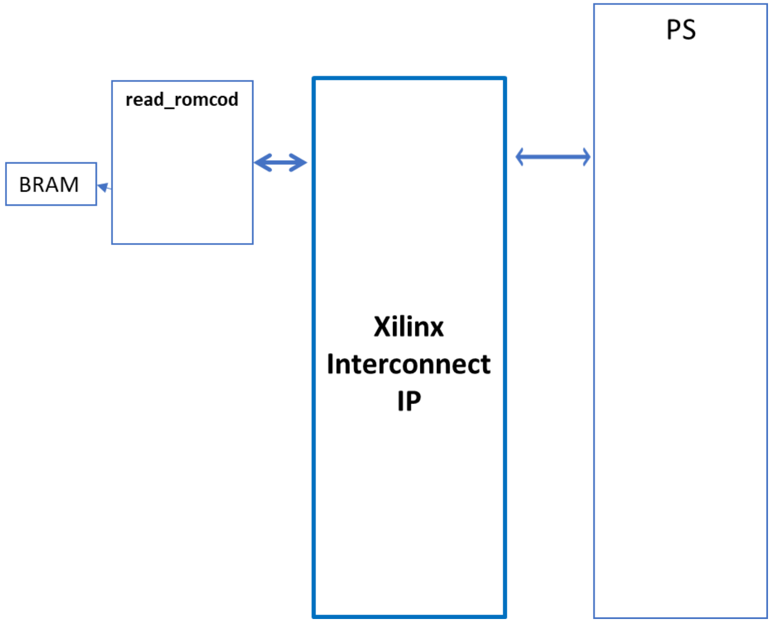
Caravel FPGA Block Diagram



Windbond spiflash model: https://www.winbond.com/hq/support/documentation/?__locale=en&line=/product/code-storage-flash-memory/index.html&family=/product/code-storage-flash-memory/serial-nor-flash/index.html&category=/.categories/resources/verilog-model/

Lab#1 : AXI Master to read/write BRAM

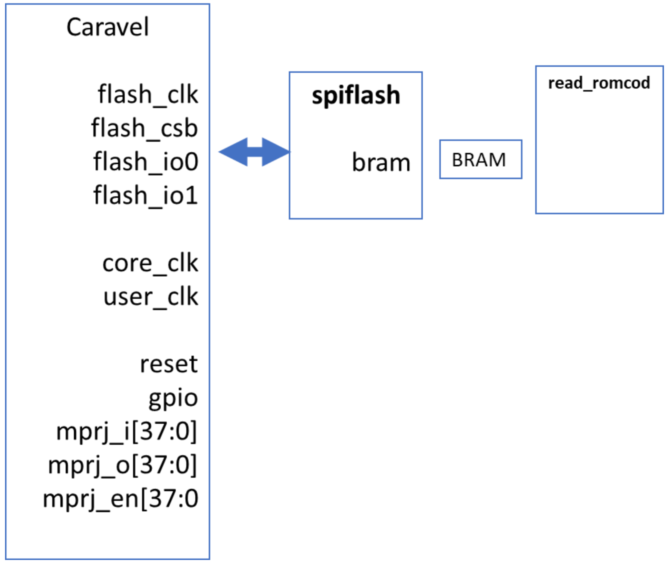
- Design source: read_romcode.cpp
- Lab Content
    1. Add another axi-master path to write to PS Memory
    2. Load program.hex (RISCV code from any of the Caravel testbench) to PS memory buffer
    3. Develop host code to load program.hex to BRAM, and read from BRAM.
    4. Compare the input and output buffer content is the same



Lab2 – spiflash protocol design and validation

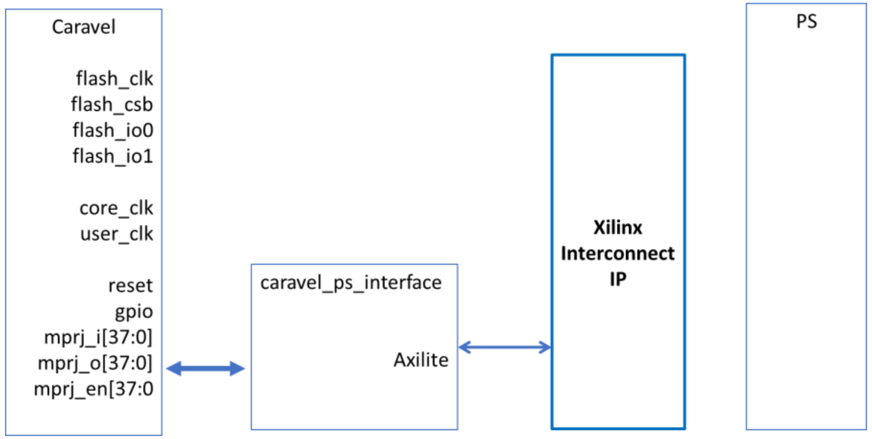
- Design source: spiflash.cpp
  - Note: this spiflash design only implements: single bit spi flash read.
- Lab Content:
  1. Develop flash controller spiflash_ctrl.v (you can find the design from Caravel) in Verilog
  2. Synthesize spiflash.cpp (this is flash memory device)
  3. Develop a simple BRAM behavior model, preload the content in testbench
  4. Integrate the spiflash_ctrl.v + spiflash.v and verify spiflash_ctrl.v can correctly read data from BRAM through spiflash.cpp

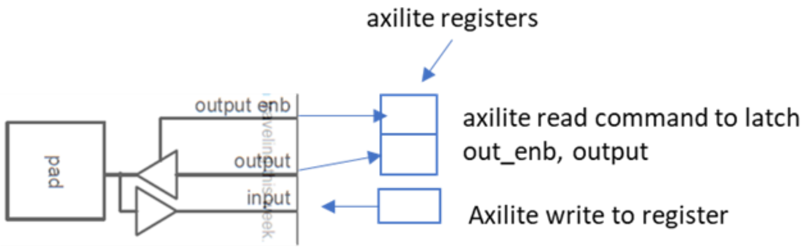
  Note: spiflash.cpp is not verified.



Lab3: Axilite access GPIO pins

- Reference Design: caravel_ps.cpp
- Lab Content
  - Design a simple module mprj_control.v
    - Use one mprj_i pin (synchronize with host code) to stage through several steps, e.g.
      - Change mprj_o pins value
      - Some of mprj pins used for loop-back, e.g.
        - mprj_o[x] = mprj_i[n]
        - Control mprj_en accordingly
      - Host use axilite to read mprj_o, mprj_en values
  - Integrate mprj_control.v & caravel_ps.v in Block design – generate bitstream
  - Develop Python host code to verify its behavior




axilite registers
pad
output enb
output
input
axilite read command to latch out_enb, output
Axilite write to register