CS411 project 2 Document

Release date: 2023.10.05

Due date : 2023.11.09

- GOAL
  - Build FPGA systolic array (8x8) system that do small size matrix multiplication
    - [Task 1, 2] Build your own matrix multiplication system with Verilog programming lan guage.
    - [Task 3] Using VIVADO, build Block Diagram to make entire system
    - [Task 4] Generate bit-stream file and communication with Python interface in Jupyter notebook.

Recommended Step

- Step 1 [Write module code]

  In this step, you must write 'matmul_system' using the Verilog language. Refer to the Example Diag ram in the assignment description PowerPoint for the internal structure of matmul_system. There is an FSM (Finite State Machine), and buffers for activation, weight, and output exist. Additionally, the re is a Systolic array. An accumulator is not mandatory. **The diagrams in the Powerpoint are for e xample answers only and do not need to be strictly followed**. There are, however, some fixed co nditions.

  1. The input/output of the PEs (Processing Elements) is fixed.
  2. The input/output of matmul_system.v is fixed.
  3. It is also determined which values should go to which address in SP_BRAM (Special purpose Block RAM)

     addr0: start signal {0: idle, 1: start}

addr4: mode {1: OS, 1: WS}

addr8: M, addr12: K, addr16: N {in all tests M,N,K <= 8 }

addr100: end signal {1: end, 0: not end}

4. You can perform functionality checks using tb_brram_combine.v

(How it works) First, you write data into the Activation, Weight BRAMs and setting parameters on SP_BRAM. After then, when you send the start signal in SP_BRAM, matmul_system should recognize it and automatically fetch data from BRAM for computation. Finally, FPGA places the result in the Output BRAM and it should set SP_BRAM's end signal indicating it has finished.

Feel free to make additional modules (.v) as needed to create the matmul_system.v.

- Step 2 [simulate and check functionality]
  - We will provide you with 'tb_pe.v' as a sample test_bench. You can use it as a reference to create your own test bench for other modules. Conducting simulation for other modules is a crucial step in completing **TASK 1**. We strongly recommend against creating 'matmul_system.v' without conducting intermediate checks and debugging.
  - **TASK 1.** You have to check if the simulation passes or not. **{matmul_system.v + BRAM}**
    - For 'matmul_system.v + 4 BRAM', you need to create a block diagram. The block diagram consists of 4 BRAM(A,W,O,SP) and matmul_system.v. Additionally, you should define ports to facilitate module testing along the BRAM components.Please refer to the provided reference image and test bench file for guidance. The simulation should follow the instructions outlined in the PowerPoint presentation.

  - Type of simulation
    - Behavior simulation **(necessary)**
    - Post-synthesis functional simulation **(necessary)**
    - Post-synthesis timing simulation **(necessary)**
    - Post-implementation functionality simulation (optional)
    - Post-implementation timing simulation (optional)

- Each simulation result may vary between Behavior simulation and Post-synthesis functional s imulation.

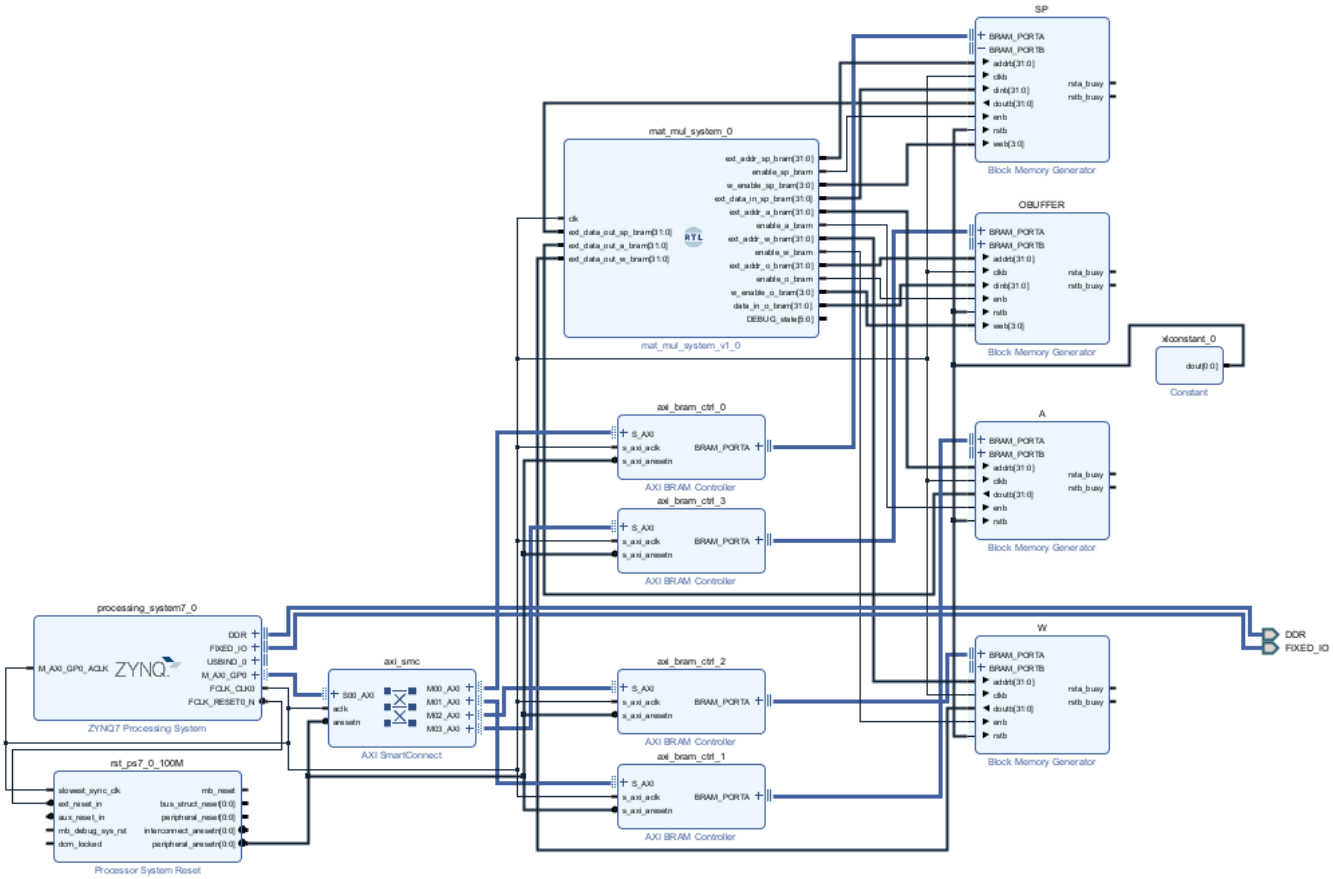
  - Behavior simulation:

    Behavior simulation involves simulating your module based on the RTL (Register Tran sfer Level) code. In this type of simulation, you are essentially testing the functional b ehavior of your design as described in the RTL code. **It is a high-level simulation th at doesn't consider the specifics of the hardware implementation, such as timing delay.**

  - Post-synthesis simulation

    Post-synthesis simulation occurs after the RTL code is synthesized using a tool like Vi vado. During synthesis, the RTL code is transformed into a netlist representation that describes the actual hardware components and their connections on the target FPGA . Post-synthesis simulation simulates the design based on this netlist, taking into acc ount the physical implementation of the design. **You need to be more concerned a bout "timing".**

- Step 3 [Build Block Diagram]
  - Entire scheme to make the system including PS of PYNQ.
  - Following [Fig 1]



[Fig 1]

- Step 4 [Make bit_stream file and Test operation with Jupyter notebook]

    - You can simply generate a 'bit stream file' as described in the 'BRAM example' provided.
    - Your task involves working with the given skeleton code for 'mamul_OS' and 'matmul_WS', b oth of which take 'mat1' and 'mat2' as inputs and should return the resulting matrix in the correct size. You are not allowed to hard coding on python in terms of matrix multiplication, which means matrix multiplication should be done within the FPGA.
    - **TASK 2**. To evaluate your code, run 'Test_OS' and 'Test_WS' without making any code chang es. Scoring policy is based on 'Test_OS' and 'Test_WS'.

Constraint

- The clock frequency is **1MHz** for stable operation.
- I/O ports of "pe.v"
- I/O ports of "matmul_system.v"
- Activation and Weight values are 1 byte size [INT8]. Output values are 4 byte size
- BRAM default setting is 4 byte size space for one element. For simplicity, You should put number [INT8] into 4 byte size BRAM space. When you get data from BRAM, you should get 1 byte of 4 bytes and put it into a buffer existing in the matmul_system.

Submission and Scores

- Submission
  - Document (4 ~ 7 page)
  - Your Verilog codes (in folder name "verilog_code")
  - VIVADO project folder (we'll generate bitstream from your VIVADO project file and test functionality)
  - jupyter notebook folder (bit_stream file, hwh_file, skeleton_code.ipynb)
- Score
  - **TASK 0** Design Document (30%)
  - **TASK 1,2** Verilog Simulation Test (Behavior + Post-Synthesis functional & timing simulation) (25%+15%)
    - Test via given tb_bram_combine.v.
  - **TASK 3,4** Test functionality in jupyter notebook (30%)
    - We will check your matmul_WS and matmul_OS function. (memory optimization check)
    - Test via Test_OS and TEST_WS. (minimum score during 10 trial)

Details of Submission Contents

- Document Contents (you should include)

1. Design part (50%)
    a. Scheme of PE, Systolic Array and explain how this work
        i. How to control the mode of pe and systolic array?
        ii. How does streaming data flow in your systolic array diagram?
    b. Scheme of Systolic array system (Project2.ppt p.10) and explain your modules one by one with this format
        i. I/O ports (each port you should mention data width and property)
        ii. Functionality (how does it work?)
2. Jupyter Notebook (30%)
    a. Suggest the "matmul_WS" and "matmul_OS" code.
    b. Explain your "matmul_WS" and "matmul_OS" function.
3. Feedback (20%)
    a. Are there any issues when you integrate your small modules into matmul_system.
    b. Other difficulties when you did project2.

- Verilog test bench TEST (Submission your Verilog code)
    - We give you test benches file of tb_pe.v, tb_bram_combine.v.
    - Check "Behavior simulation", "Post-synthesis functional simulation", "Post-synthesis timing si mulation" pass the test cases.
    - **If you pass the test of tb_bram_combine.v for three simulations, you will get a full scor e.**

- VIVADO project folder
    - You may create a Vivado project to generate a bitstream file and simulate your module.
    - You should submit your entire VIVADO project that you generated a bitstream file, since we will use it so that your block diagram can generate a bit stream file well.