**CO262 - Computer Organization and Architecture**

# A2 - Design the RISC-V Datapath in Verilog

- Objective: Design datapaths to execute R-type, Load/Store, and branch-if-equal instructions.
- Hardcode the control inputs for the datapath designs.
- You may ignore the blocks that are not required.
- **Deadline:** 9AM, January, 11, 2019.
- **Submission guidelines:** Team assignment. Team size <= 2. Pack code, screenshots, testcases, etc. in an archive. Mail to co262.nitk@gmail.com.

1. **R-type Instruction Datapath.** Use the Register File and ALU (datapath + control) and implement the datapath for an R-type RISC-V instruction. The behavioural code for this block should update the output register accurately. The entire block is combinational. The results should be verified by the testbench by reading out the output register. You may use the block diagram in Fig. 4.19, Page 258.
2. **I-type ALU Instruction.** Implement the datapath for an I-type RISC-V ALU instruction (Eg. `addi x5, x6, -4`). Input to the block: The RISC-V encoded, 32-bit I-type ALU instruction. The design of this datapath is similar to the R-type with two major differences. The immediate operand is output from an *Immediate Generation* block and fed to the second input to the ALU.
3. **Memory Access Instruction Datapath**. Implement the datapath for the load nd store instruction execution (Eg. `ld x5, 8(x6)`). The ALU operation input can be hardwired to the appropriate value (or can be fed from a constant register) such that the ALU performs effective address calculation at all times. The Data memory should be instantiated from A1.Q1. The datapath is shown in Fig. 4.20 (Page 259).
4. **Branch on Equal Instruction Datapath**. Implement the datapath to execute a BEQ instruction (Eg. `beq x5, x6, Offset`). The datapath block diagram is shown in Fig. 4.21.