In a brief way, write only what you understand about the following in your own style:

- Draw basic UVM Test bench Architecture block diagram supporting Emulation.
- What are the main criteria in developing UVM TB Emulation friendly?
- Summarize digital design flow.
- Summarize Functional Verification flow.

Note:

- ***It is allowed to read & search over internet.***
- ***It is not allowed to take a copy from any website or books.***

Computation Storage Project:

Introduction:

- In common CPU's, to make an operation (result = op1 (+/-/*/..) op2) on data saved in a memory. The host needs to
  - Send memory read command to read op1.
  - Send memory read command to read op2.
  - Send attended arithmetic command = op1 (+/-/*/..etc) op2
  - Send memory write to save result in another memory address.
- **Computation Storage architectures**, allow the host to this in one command
  - send one commend with address 1 **to read OP1**, address 2 **to read OP2**, **operation** (add/sub/..etc), write address **to store results**.
  - The DUT internally will handle the whole operation.

Module: Develop small computation storage module

- Interface:
  - Input: CLK/RESET, addA, addB, addC
  - In/Out: DQ
  - Any other needed signals
- Functionality:
  - Operations:
    1. RD_MEM_CMD: (DQ= Mem[addA])
       - Input to DUT addA
       - DUT read from Mem[addA] & out data on DQ
    2. WR_MEM_CMD: (Mem[addC]=DQ)
       - Input to DUT addC, DQ
       - DUT writes DQ @ location addC in Memory
    3. ADD_CMD: Mem[addC] = Mem[addA] + Mem[addB]
       - Input addA, addB, operation.
       - DUT reads op1 from add addA, op2 from addB, execute result = op1+ op2, save result in memory @ location addC
    4. SUB_CMD: Mem[addC] = Mem[addA] - Mem[addB]
       - Input addA, addB, operation.
       - DUT read op1 from add addA, op2 from addB, execute result = op1- op2, save result in memory @ location addC
- Language: System Verilog.
- Target platforms: Simulation/Emulation/FPGA
- Memory preloading:
  - Initialize memory via readmemh.
  - Generate preloading file using any script

Testing:

- UVM testing is preferred, system Verilog is ok.
- Test 4 operations.
- Testing: Questa
- Compare final memory contains with preloaded file - using any script -.
- Coverage module.

Documentation:

- Project structure.
- Needed sequences to be covered & Corner cases.

Note: Assume any missing data.