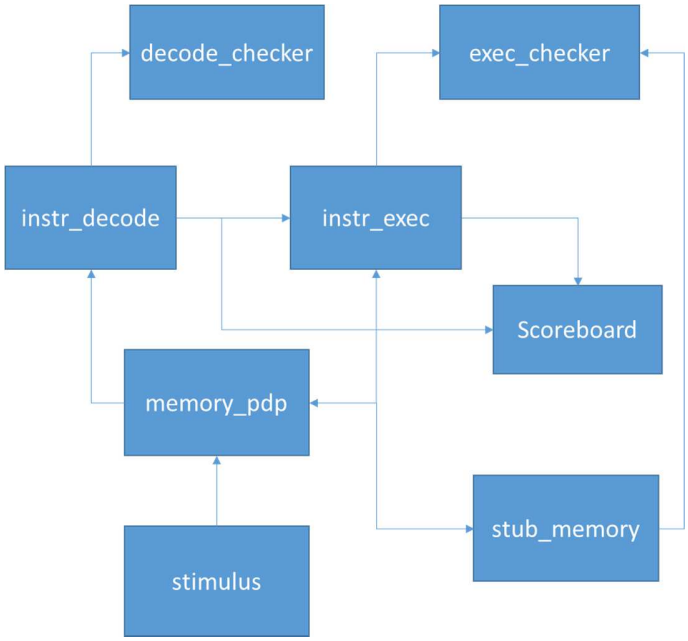
ECE 510: Pre-Silicon Validation

Validation of PDP-8 Instruction Set Simulator

Kanithahalli Sadananda, Manas
instr_exec unit-level verification
memory_pdp post-simulation verification
Chip-level verification
BONUS TASK for instruction sequence coverage

Karanth Kota, Vishak
instr_decode unit-level verification
Documentation

Chip-Level Verification:



BEST FEATURES:

1. Assembly Level Testing as shown in ***test.as*** and ***test.mem*** file
2. Can track the stage of instruction (instruction fetch, instruction decode, instruction execution). This is part of top.sv file which is commented out due to large amount of logging this causes. The messages are logged in ***instr_stage.log***
3. ***On-the-fly*** checking for every instruction in chkr_instr_exec.sv
4. ***Reuse of chkr_instr_exec.sv*** module from the instr_exec unit-level verification. No modification done whatsoever!
5. ***Post-simulation validation of memory module*** by comparing ***out.mem*** with ***stub.mem***
6. ***Scoreboard*** for instruction coverage as shown (see screenshot below)
7. Assertions to ensure correct state transitions
8. All checkers written tin SystemVerilog
9. **BONUS TASK:** Coverage for the instruction sequence CLA_CLL > TAD > TAD > DCA > HLT> JMP. This is part of the ***top.sv*** file

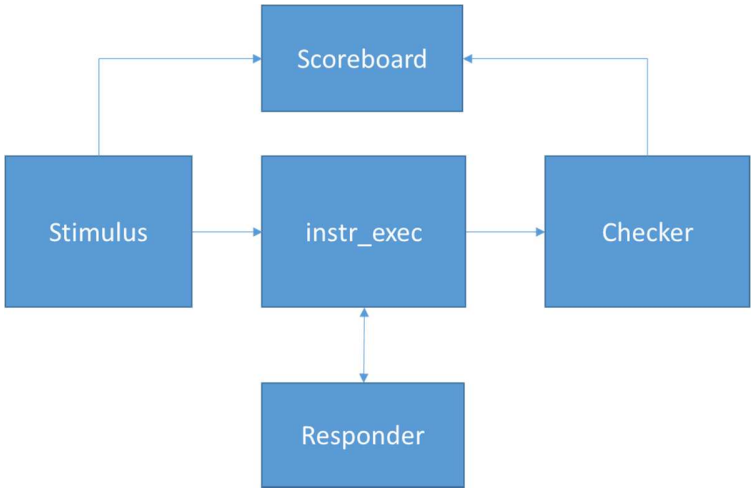
The full chip level verification is done by giving out a test case which consists of all the opcodes. The individual signals of each modules are checked for their correctness with their individual checking components.

The “test.as” is a file which is converted from the assembly level file to the opcode included file “test.mem”. This file is read by the memory module. The instructions and the address are mentioned in the very beginning of file. The obtained opcodes are given to the decode unit, later to the execute unit.
The “out.mem” file is a file written by the memory module which has the outputs given by the execute unit to the memory module. The PDP-8 Instructions which were implemented in the design are as follows.

The “stub.mem” is a file written by the stub memory module that wiill be used to validate the memory module post-simulation by comparing the ***stub.mem*** with ***out.mem***
The “scrbrd.log” is a file written by the ***chkr_instr_exec*** module which consists of the coverage data for instructions as shown

```
1 AND_count:        20548
2 TAD_count:        41098
3 JMP_count:        20542
4 JMS_count:        20546
5 DCA_count:        61642
6 ISZ_count:        20551
7 CLA_CLL_count:        20549
```

instr_exec module Unit-level Verification

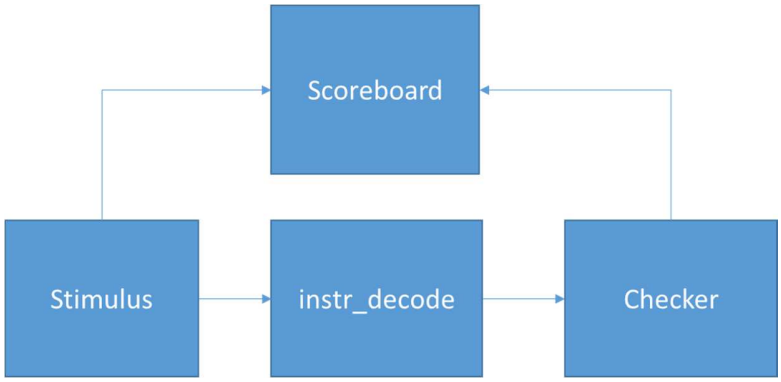


BEST FEATURES

1. Both constrained and unconstrained randomized stimulus
2. Directed test cases as part of the stimulus prior to randomized stimulus
3. Versatile checker module that is fully re-used at chip-level verification without any modifications whatsoever!
4. On-the-fly checking in checker module
5. Responder module mimicking a stub memory module for data
6. Scoreboard displayed after simulation run showing instruction ***coverage data*** as shown

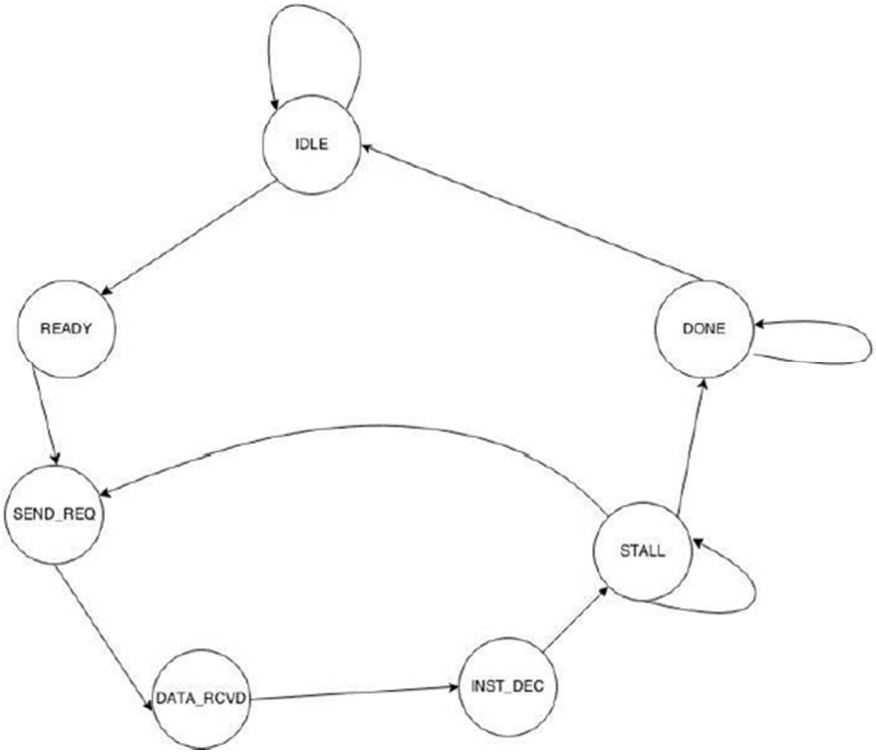
```
Beginning Stimulus for Directed Testing of Memory Reference Instructions
Beginning Stimulus for Directed Testing of Microinstructions
Beginning Stimulus for Randomized Testing of Memory Reference Instructions
Beginning Stimulus for Randomized Testing of Microinstructions
AND:        6511
TAD:        4171
JMS:        1981
DCA:        2923
JMP:        1825
ISZ:        2295
CLA_CLL:        7273
```

instr_decode Unit-level verification



Verification Plan: On-the-fly, Constrained-Randomized testing.
The decoder is connected to a testing module, which has a stimulus generator and a checker. The stimulus generator is responsible for generating random instructions to the DUT. The Checker contains a behavioral model of the DUT. The following block diagram shows the implementation of the verification plan.

Decode Unit FSM
The Decode unit contains the following Finite State Machine



Execute Unit FSM

The Execute unit contains the following Finite State Machine