ECE 510 FUNDAMENTAL PRE-SILICON FUNCTIONAL VALIDATION

SPRING 2016

HARDWARE IMPLEMENTAION OF PDP8 INSTRUCTION SET ARCHITECTURE (ISA) LEVEL SIMULATOR

BY

BHARATH REDDY GODI,

SURENDRA MADDULA.

Contents:

Overview of PDP-8 Architecture:

Introduction: The PDP-8 (Programmed Data Processor) was introduced in 1965 by Digital Equipment Corporation (DEC). The original PDP-8 was built using transistor technology making it a second generation computer but a third generation model using MSI technology, the PDP8/I, was introduced in 1968.The PDP-8 was considered a *mini-computer* because of its small size and low cost. Memory consisted of 4K words, each word being 12 bits.

The Full system is divided in to below blocks:

1. Instruction decode block
2. Instruction execution block
3. Memory block
4. Clock generator block

Understanding of the full chip Design:

The Full chip system consists of a Memory Unit, Instruction fetch and decode unit, and an Instruction execution unit. The memory consists of 4096 words. Where word length of each word is 12 bit each. The memory unit reads the data from a compiled pdp8 assembly language test.

The chip will be in reset initially. It comes out of reset when reset_n signal is asserted high. Once after coming out of reset, the instruction fetch and decode unit fetches the first instruction from a base address i.e. Octal 200.

The instruction fetch and decode unit sends the fetched instruction to instruction execution unit where it does the specified operations based on instruction and reads from memory or writes from memory. After finishing with the current instruction, IFD fetches the next instruction from the memory unit and gives it to EXE unit. This process continues until all the instructions in the memory are completed. The last instruction address is same the base address just to indicate that it is the last it is the instruction.

Supported Instructions:

AND: **Logical AND.** The content of memory location specified by 'xxx' is combined with the content of the AC using a bitwise logical AND operation. The result is left in the AC.

TAD: **Two's Compliment Add.** The content of memory location specified by 'xxx' is added to the contents of the AC using two's compliment addition. The result is left in the AC, if the result overflows, the LINK bit is complimented.

ISZ: **Increment and Skip if Zero.** The content of memory location specified by 'xxx' is incremented by one and restored to memory. If the result of the increment was zero, the program counter (PC) is incremented one extra time to cause the execution of the instruction following the ISZ to be skipped.

JMS: **Jump to Subroutine.** The address of the instruction following this one is stored into the memory location specified by 'xxx' and the PC is set to the location following 'xxx'.

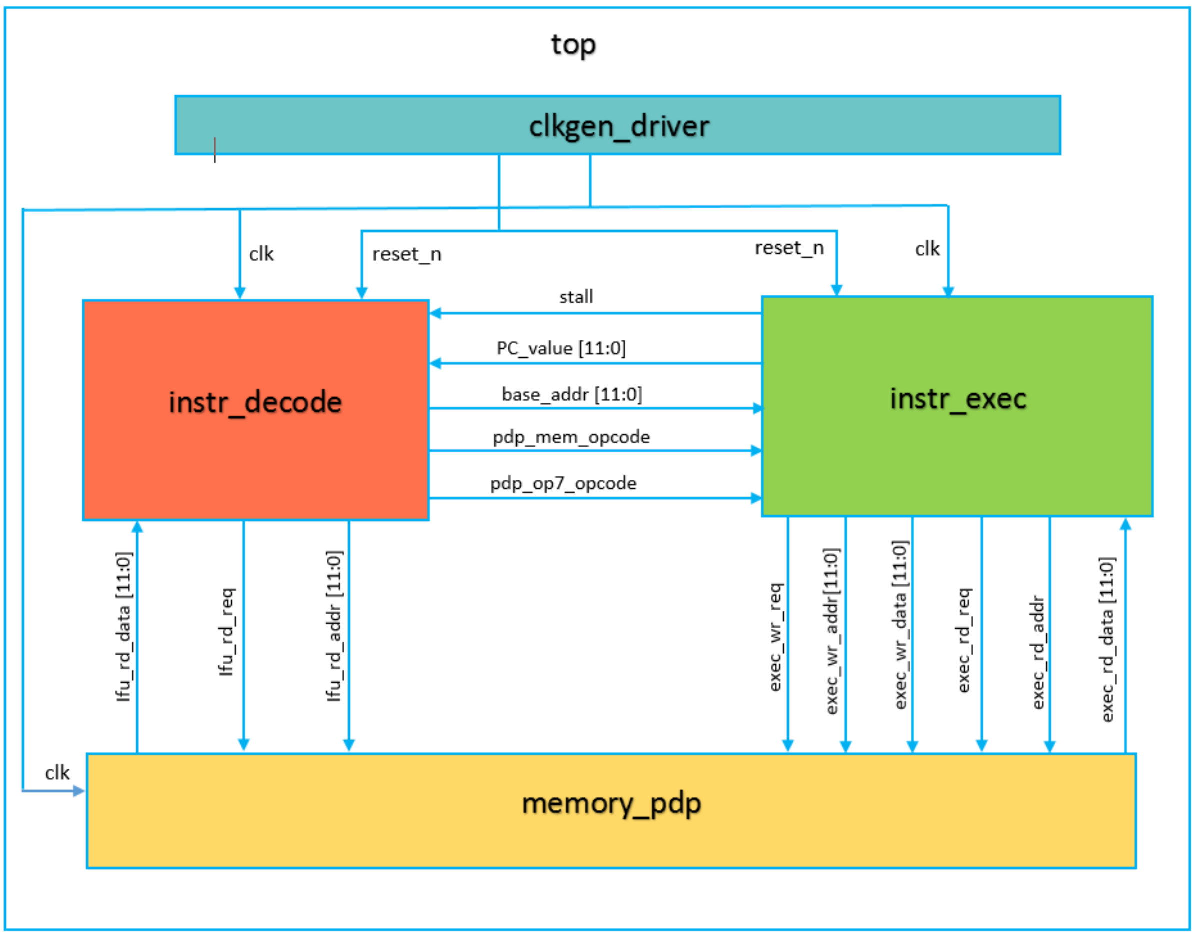
JMP: **Jump.** The PC is set to the location specified by 'xxx', causing the machine to execute its next instruction from there.

CLA_CLL: The **accumulator** and **link** will be cleared.

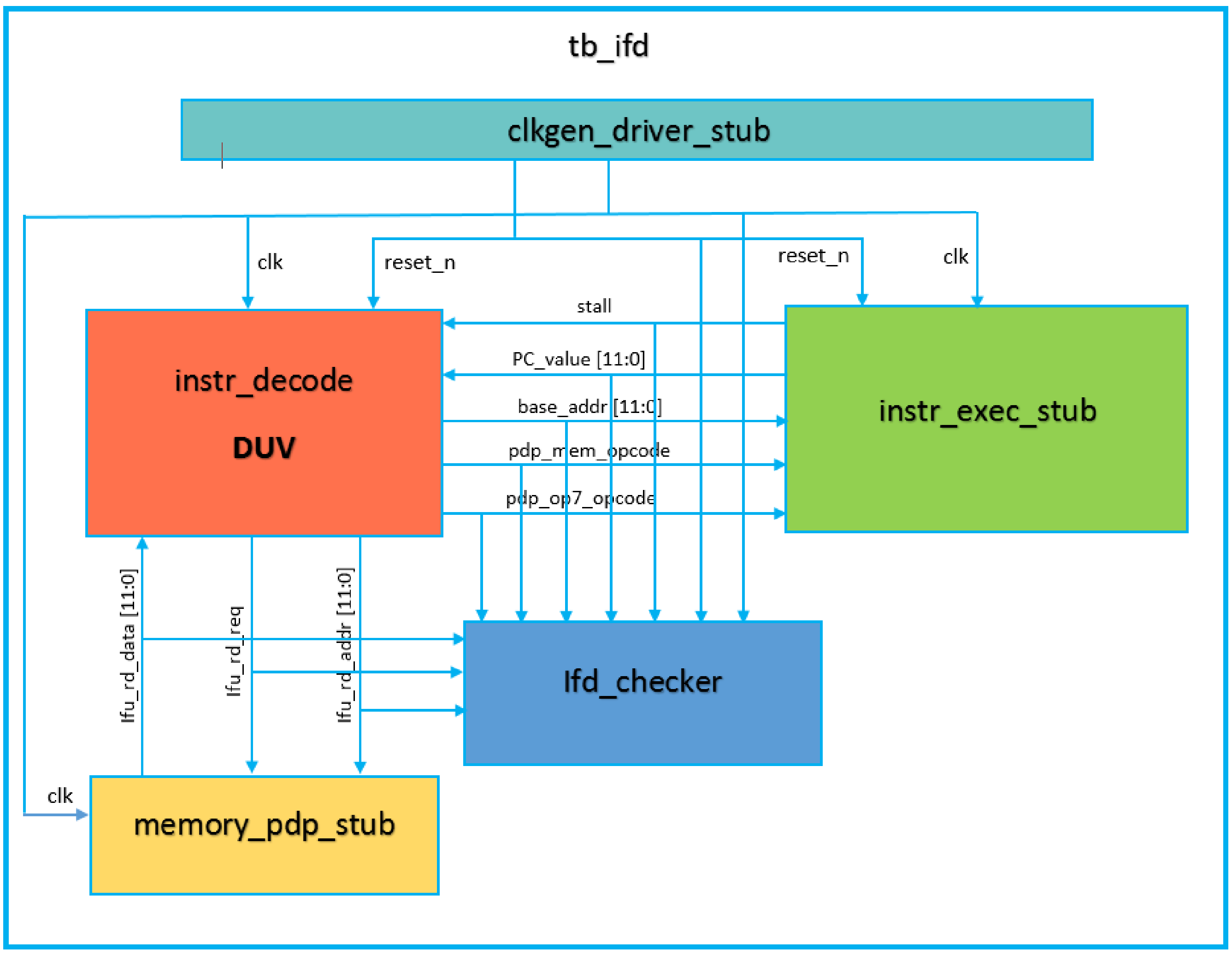
DCA: Deposit and Clear Accumulator. The content of the accumulator (AC) is stored into the memory location specified by 'xxx' and the AC is cleared to zero.

NOP: No Operation

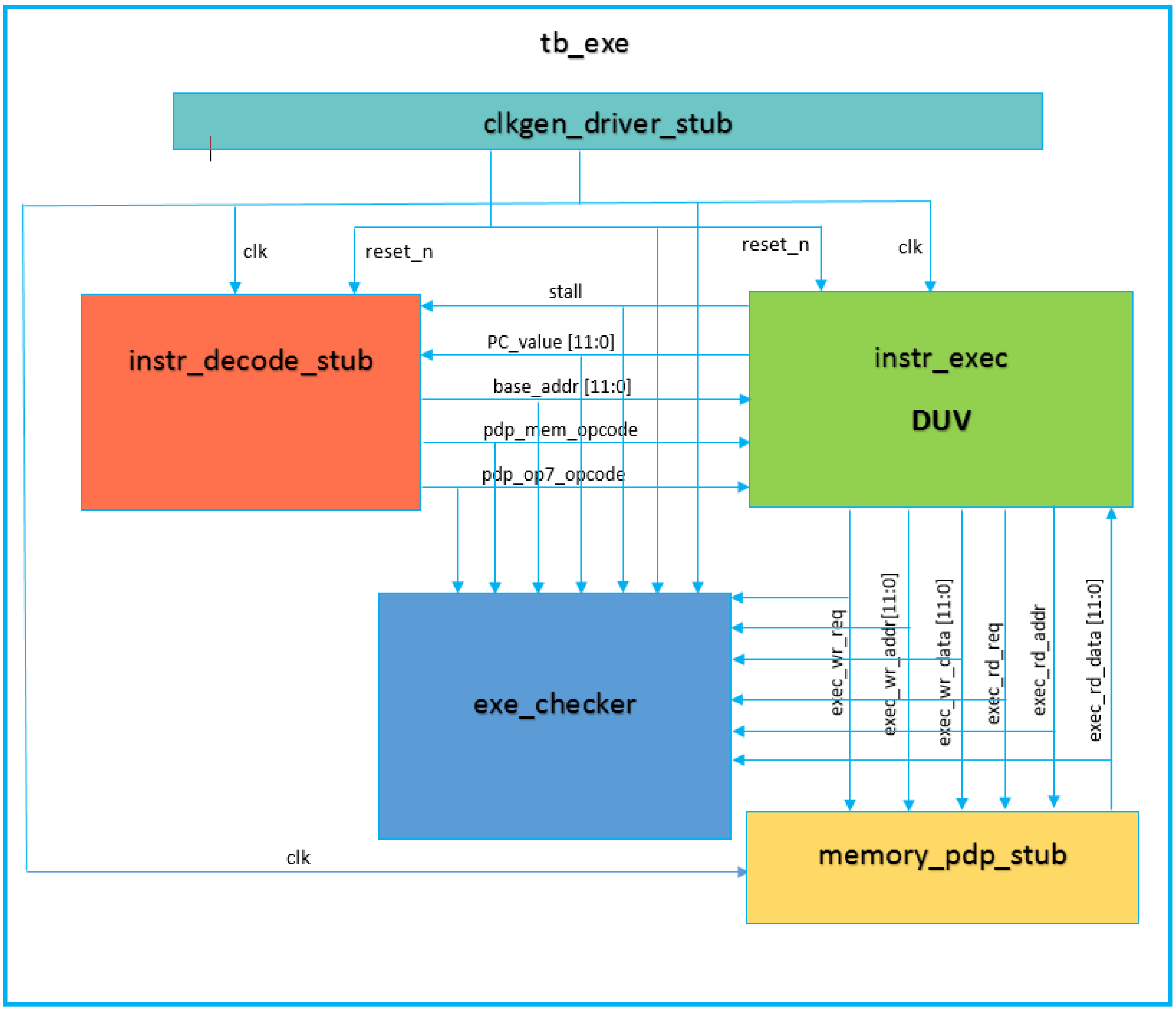
Full Chip Level block Diagram:



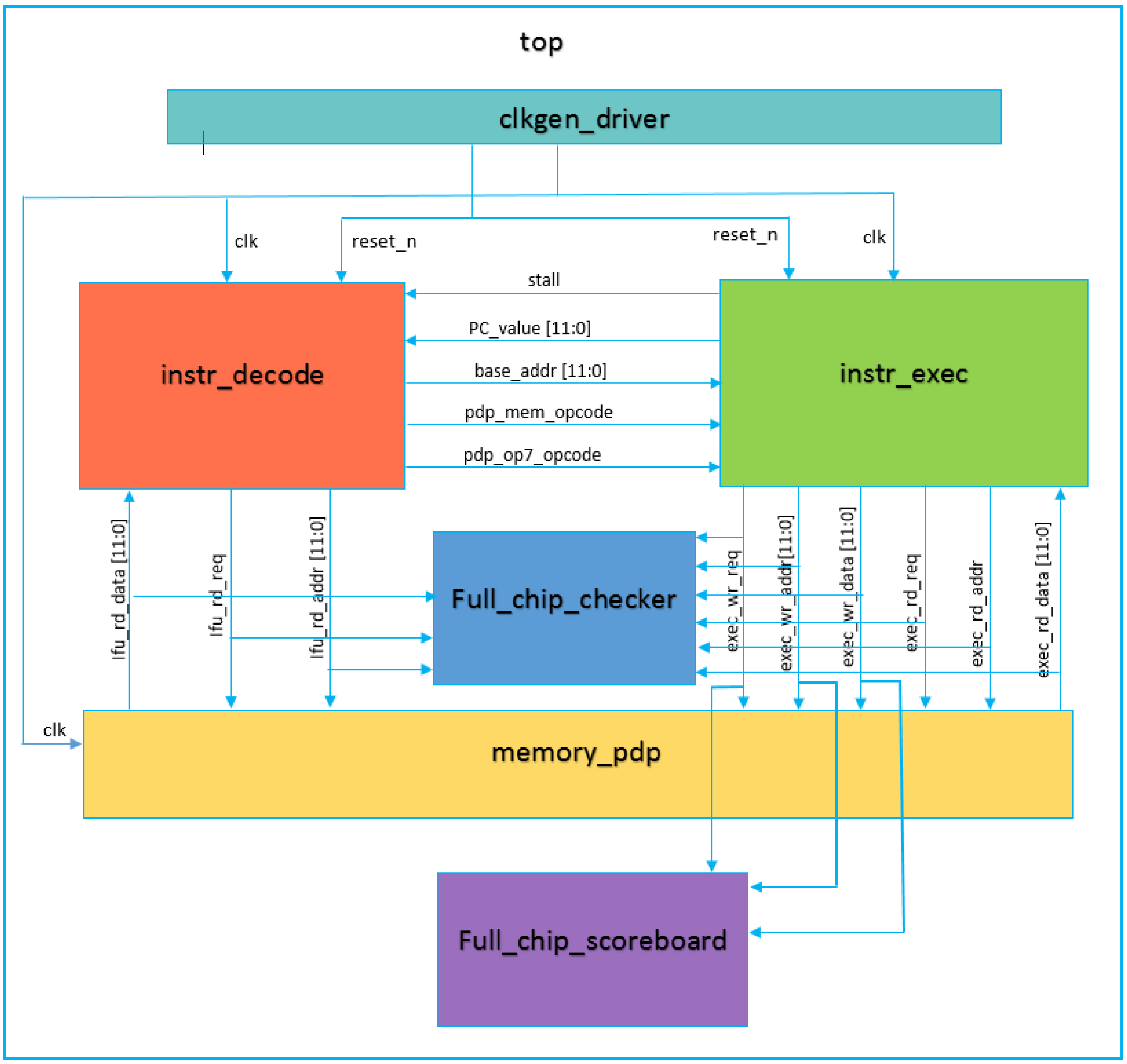
Unit- Level Block Diagram of Instruction Decode.



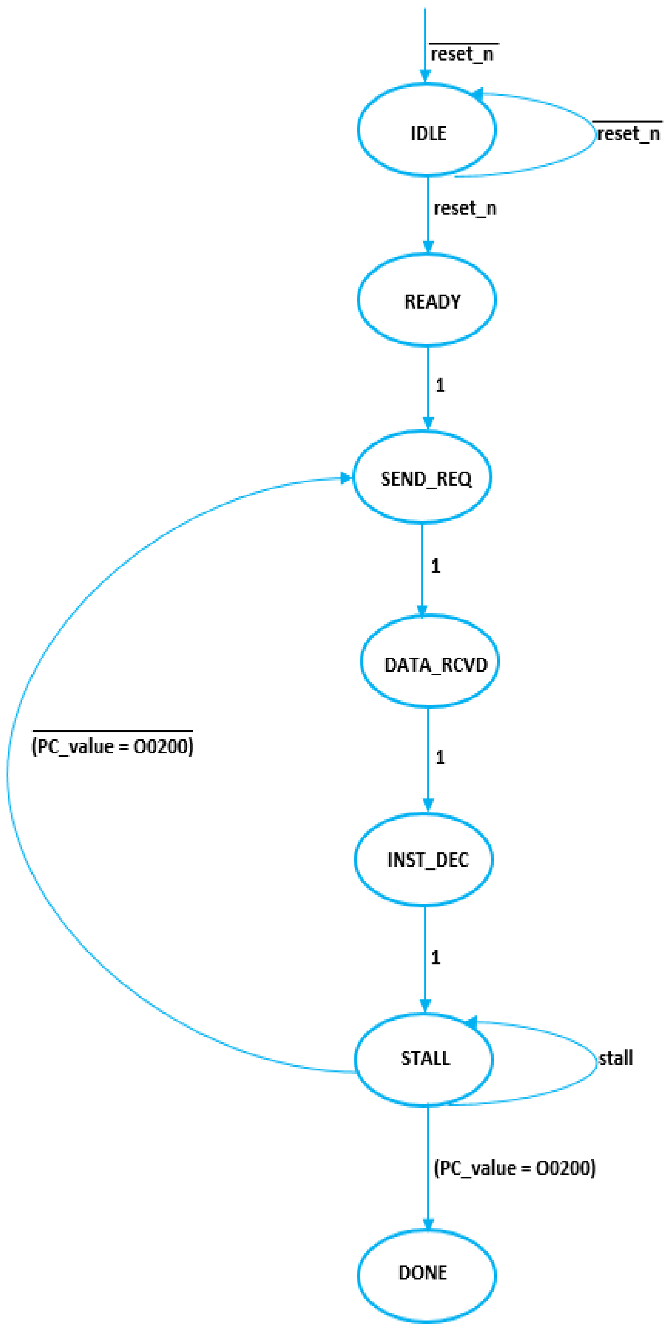
Unit- Level Block Diagram of Instruction Execution.



Chip - Level Block Diagram.



Finite State Machine for Instruction Decoder unit



$\overline{\text{reset\_n}}$
IDLE
$\overline{\text{reset\_n}}$
reset_n
READY
1
SEND_REQ
1
DATA_RCVD
$\overline{\text{(PC\_value = 00200)}}$
1
INST_DEC
1
STALL
stall
(PC_value = 00200)
DONE

Finite State Machine for Instruction Execution unit:

A= new_mem_opcode || new_op7_opcode

B= pdp_op7_opcode.CLA_CLL

C= pdp_mem_opcode.TAD || pdp_mem_opcode.AND || pdp_mem_opcode.ISZ

D= pdp_mem_opcode.DCA

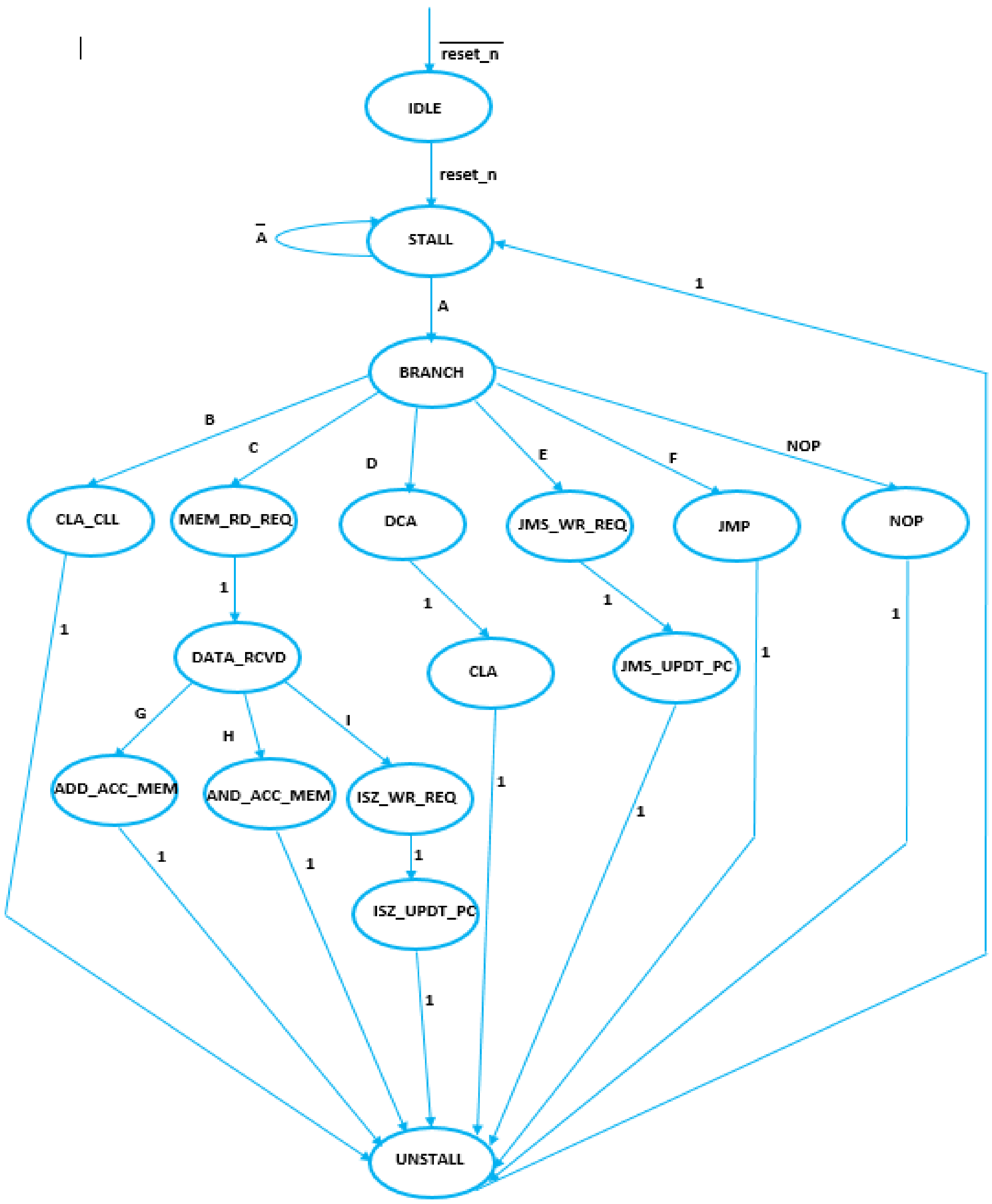
E= pdp_mem_opcode.JMS

F= pdp_mem_opcode.JMP

G= pdp_mem_opcode.TAD

H= pdp_mem_opcode.AND

I= pdp_mem_opcode.ISZ


$\overline{\text{reset\_n}}$
IDLE
reset_n
STALL
$\bar{A}$
A
1
BRANCH
B
C
D
E
F
NOP
CLA_CLL
MEM_RD_REQ
DCA
JMS_WR_REQ
JMP
NOP
1
1
1
1
1
1
DATA_RCVD
CLA
JMS_UPDT_PC
G
H
I
ADD_ACC_MEM
AND_ACC_MEM
ISZ_WR_REQ
1
1
1
1
1
ISZ_UPDT_PC
1
UNSTALL

Verification Plan:

Technical Aspect:

1. Description of Verification levels

We have chosen to verify in two levels, they are

a. Unit Level
   Unit level consists of verification of Instruction fetch and decode unit, Instruction execution unit and Memory unit.
b. Chip Level
   Chip level consists of verification of all the three units together as a single unit. I.e. the Full functionality.

2. Functions to be verified

IFD:

1. When reset_n is asserted all the outputs and internal registers should be cleared.
2. When stall is asserted high, IFD should not fetch new instruction from the memory unit.
3. Check whether the instruction fetched are decoded properly.
4. When PC_value is equal to base address after program starts, IFD should stop fetching further instructions.

EXE:

1. Check whether intended functionalities are executed properly (instruction like AND, JMP, CLA_CLL, TAD, ISZ).
2. Check whether PC_value is calculated properly (during Jump & non-jump instructions).

3. Specific tests and methods & Validation Strategy

IFD: Black box testing. Random verification.

a. Initiators - clkgen_driver_stub
b. Responders- instr_exec_stub, memory_pdp8_stub
c. Checkers- Ifd_checker

EXE: Black box testing. Random verification.

a. Initiators - clkgen_driver_stub
b. Responders- instr_decode_stub, memory_pdp8_stub
c. Checkers- exe_checker

CHIP LEVEL: Grey box testing. Formal verification.

a. Checkers- full_chip_checker
b. Scoreboard- full_chip_scoreboard

4. Coverage requirements

1. The environment has exercised all types of commands and transactions.
2. The environment has driven varying degrees of legal concurrent stimulus.
3. The initiator and responder components have driven errors in to the DUV.

5. Corner case Scenarios:

To be prepared

Project management aspect:

1. Required tools

- Questa sim simulator.
- Version: Questa Sim -64 10.4c
- Verification Language - System Verilog

2. Risks and dependencies

Risks: There are no potential risks observed. Since all the requirements are cleared and development code was freeze.

Dependencies: Schedule for the validation activity has been prepared. Resource and tool availability has no problem with executing the plan.

3. Resources requirements

Team members:

- Bharath Reddy Godi: Unit level verification of Execution unit.
- Surendra Maddula: Unit level verification of Decode and Memory unit.

Chip level verification, coverage and report written by both of us.

4. Schedule details

S.NO	DATE	TASK
1	05/08/2016	Understanding of Project.
2	05/11/2016	More Clarity about the project. Proposal preparations.
3	05/18/2016	Proposal demo.
4	05/23/2016	Planned completion of IFD unit.
5	05/26/2016	Planned completion of EXE unit.
6	05/29/2016	Planned completion of memory unit.
7	06/05/2016	Full chip verification
8	06/05/2016	Documentation
9	06/09/2016	Final Demo.

References:

1. **http://www4.wittenberg.edu/academics/mathcomp/shelburne/comp255/notes/PDP8Overview.htm**
2. **http://homepage.cs.uiowa.edu/~jones/pdp8/man/micro.html#group1**
3. **https://en.wikipedia.org/wiki/PDP-8**
4. **Lecture slides**