

ECE 510

FINAL PROJECT REPORT

This document briefly describes the design Specifications and the verification strategy used along with the coverage and Errors determined in the DUT through the verification analysis.

Rohit Kulkarni Anuja Vaidya

Contents

[Introduction: 3](#_Toc421648731)

[Design components 4](#_Toc421648732)

[IFD Unit 4](#_Toc421648733)

[Description: 4](#_Toc421648734)

[Execution Unit 5](#_Toc421648735)

[Description: 5](#_Toc421648736)

[Memory 6](#_Toc421648737)

[Description: 6](#_Toc421648738)

[Clock Gen 6](#_Toc421648739)

[Description: 6](#_Toc421648740)

[Unit Level Testbenches 6](#_Toc421648741)

[Unit Level Testbench for IFD unit 7](#_Toc421648742)

[Stimulus: 8](#_Toc421648743)

[Checks to perform: 8](#_Toc421648744)

[Coverage Results: 8](#_Toc421648745)

[Unit Level Testbench for EXEC unit 9](#_Toc421648746)

[Stimulus: 9](#_Toc421648747)

[Checks to perform: 10](#_Toc421648748)

[Coverage Results: 11](#_Toc421648749)

[Chip level testbench 12](#_Toc421648750)

[Stimulus: 12](#_Toc421648751)

[Checks to perform: 13](#_Toc421648752)

[Coverage Result: 13](#_Toc421648753)

[Functional Coverage at the full chip: 13](#_Toc421648754)

[Design Bugs: 13](#_Toc421648755)

Figure 1: FSM decode unit ............................................................................................................................ 3

Figure 2: Fsm: Execution unit ......................................................................................................................... 4

Figure 3: Unit Level test Bench: IFD unit ....................................................................................................... 6

Figure 4: UNIT level test bench: Execution unit............................................................................................ 8

Figure 5: FULL chip test bench.................................................................................................................... 11

# Introduction:

Four principal modules in design:

* Clkgen\_driver
* Instr\_decode
* Instr\_exec
* Memory\_pdp

Goal: Build a verification environment around the design

Verification Strategy:

* Unit Level Testbenches for – IFD Unit

– EXEC Unit

* Full chip validation for entire design

# Design components

## IFD Unit

### Description:

IDLE

INST

\_

DEC

READY

SEND

\_

REQ

DATA

\_

RCVD

STALL

DONE

Reset

\_

n

!

Reset

\_

n

PC

\_

value

==

int

\_

base

\_

add

STALL

!

PC

\_

value

==

int

\_

base

\_

add

!

Reset

\_

n

Reset

\_

n

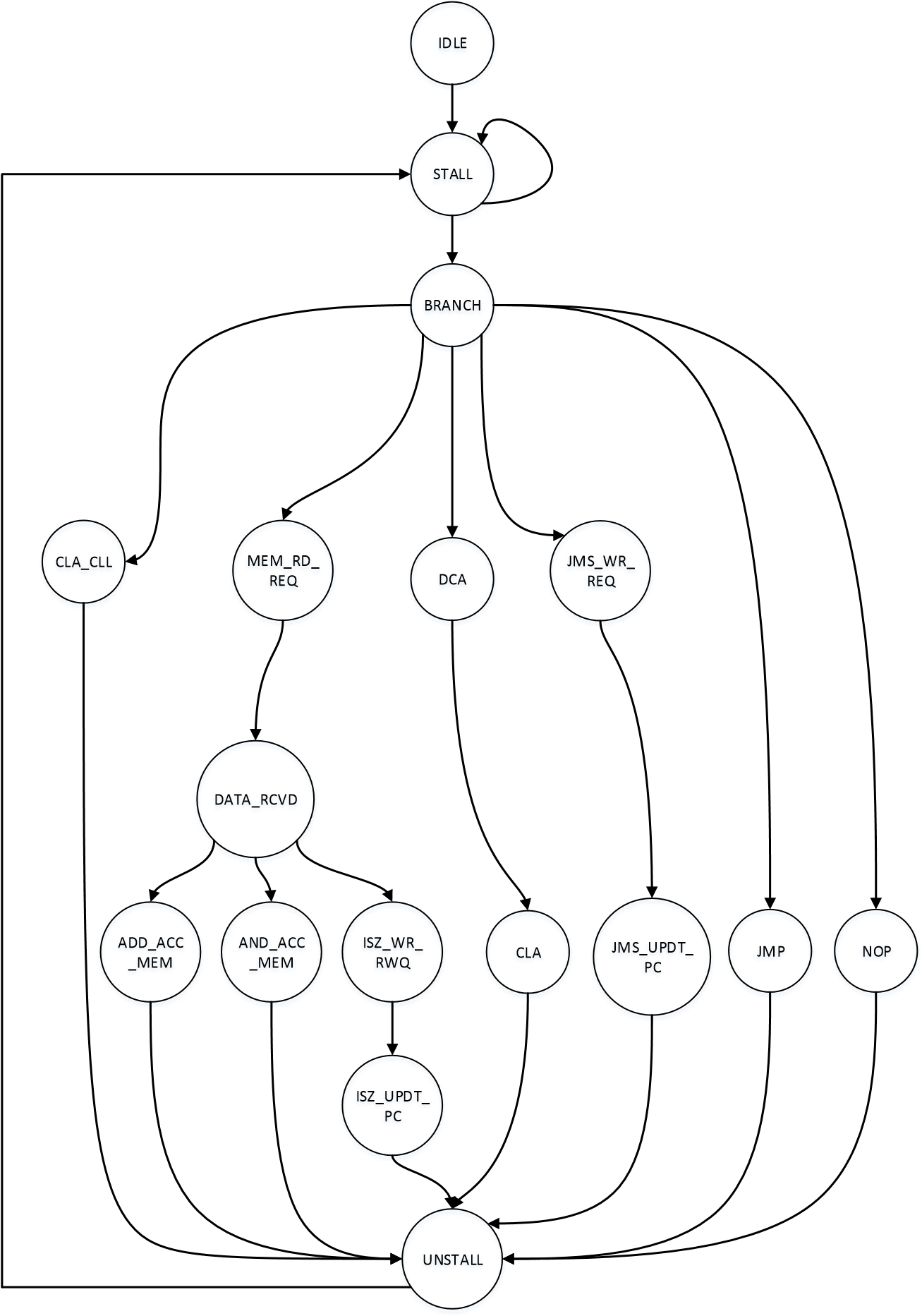
**FIGURE 1: FSM DECODE UNIT**

* Fetches instruction
* Decodes fetched instruction
* Supports decoding of
  + 6 memory reference instructions
* Opcode 0 to 5
  + 22 microinstructions
* Opcode 7 : Group 1 and Group 2

## Execution Unit

## Description:

* Executes instruction decoded by IFD unit.
* Accesses memory if instruction among opcodes 0 to 5.
* Reads operand from memory
* Writes result into memory
* Stalls IFD unit until current instruction completes execution

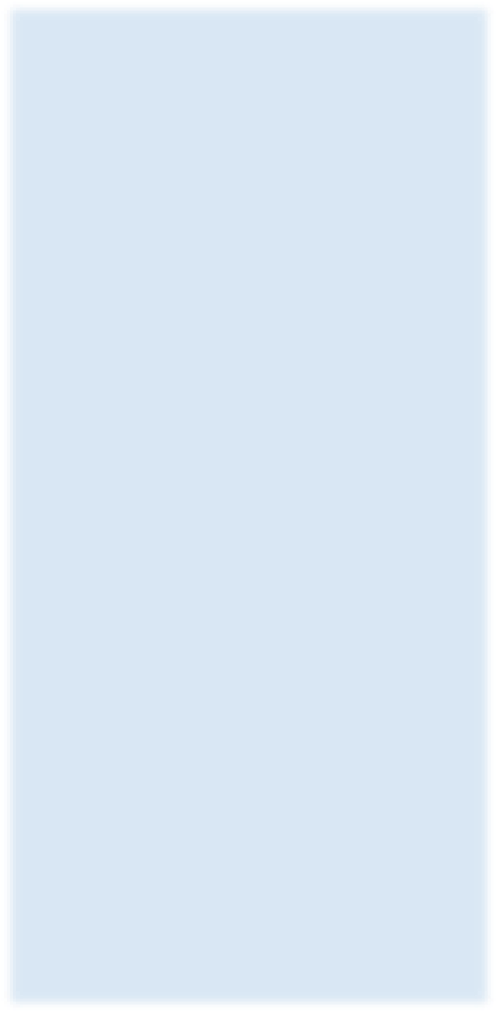


**FIGURE 2:FSM: EXECUTION UNIT**

## Memory

### Description:

• The memory unit (4K words of 12bit each) is pre-loaded with data derived from a compiled PDP8 assembly language test. IFD unit fetches instruction from memory. EXEC unit fetches operand from memory and writes back the result.



Memory

\_

pdp

clk

Ifu

\_

rd

\_

req

Ifu

\_

rd

\_

addr

Exec

\_

wr

\_

req

Exec

\_

wr

\_

addr

Exec

\_

wr

\_

data

Exec

\_

rd

\_

req

Exec

\_

rd

\_

data

Ifu

\_

rd

\_

data

Exec

\_

rd

\_

addr

## Clock Gen

### Description:

• This module provides clock and reset to all the modules in this design.



Clkgen

\_

driver

clk

Reset

\_

n

# Unit Level Testbenches

## Unit Level Testbench for IFD unit



instr

\_

decode

clk

Reset

\_

n

stall

Pc

\_

value

Ifu

\_

rd

\_

data

Ifu

\_

rd

\_

req

Ifu

\_

rd

\_

addr

Base

\_

addr

Pdp

\_

mem

\_

opcode

Pdp

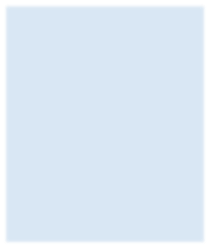
\_

op

7

\_

opcode



Memory

\_

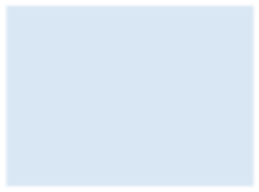
BFM



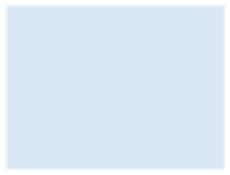
Exec

\_

BFM



Checker



Clkgen

\_

driver

clk

Reset

\_

n

clk

Reset

\_

n

**FIGURE 3: UNIT LEVEL TEST BENCH: IFD UNIT**

### Stimulus:

• Stimulus:

* IFU\_RD\_DATA from MEMORY\_BFM
* STALL from EXEC\_BFM
* PC\_VALUE from EXEC\_BFM
* RESET\_N from CLK\_GEN

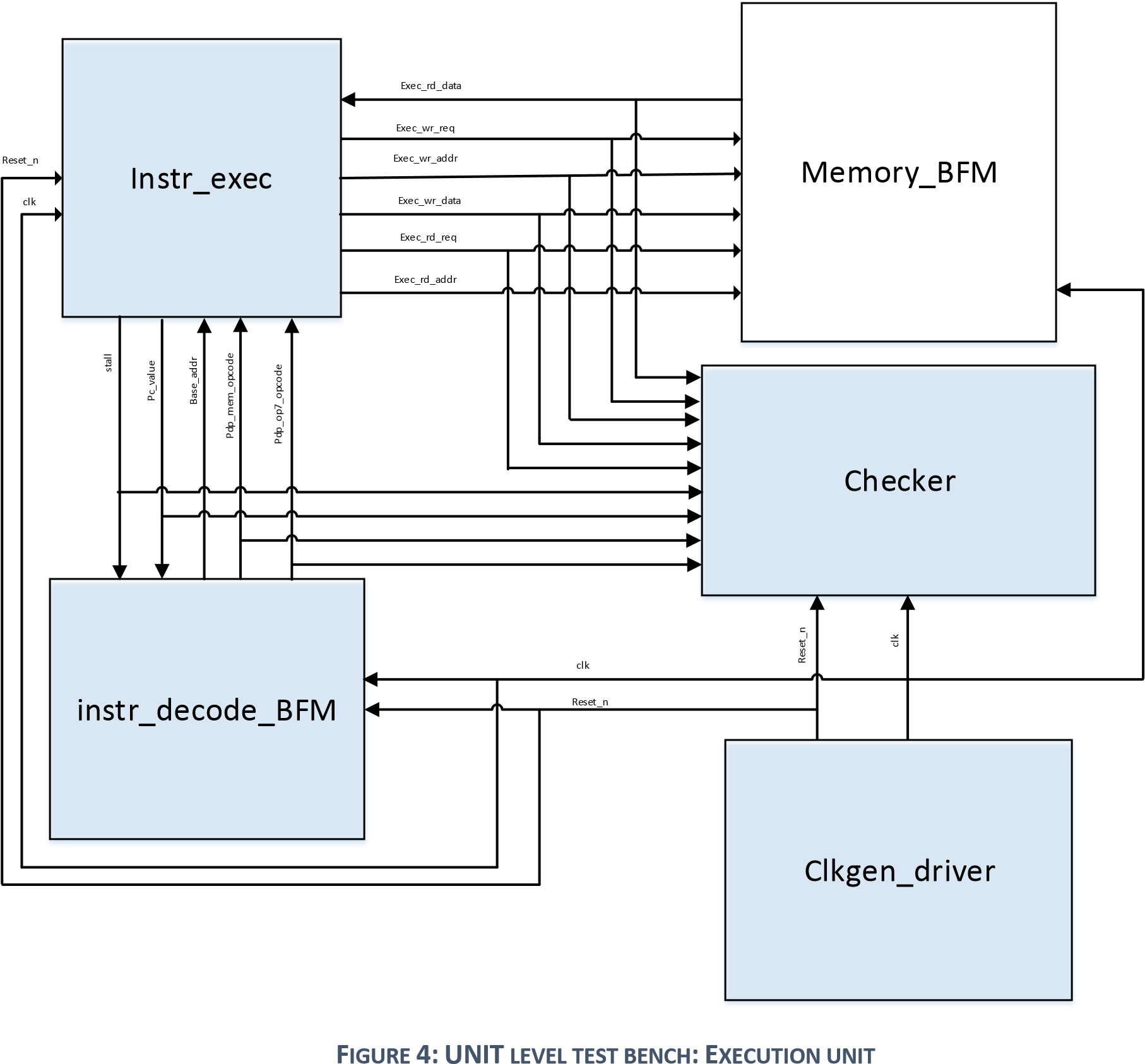
### Checks to perform:

* Checks to perform:
* Check if opcode sent to EXEC is correctly decoded
* Check if address sent to EXEC is correctly decoded
* When STALL is asserted, IFD should not fetch the next instruction and no new requests sent to EXEC unit
* If reset\_n asserted at any point, then all outputs are cleared within 1 clock cycle
* On asserting reset, next instruction should be fetched from o200
* If current state = STALL and PC = 0200, then next state = DONE
* No instructions are fetched after going into DONE state
* Check timing requirements such that the FSM stays in the particular state only for 1 cycle.
* Check whether illegal opcodes are decoded as NOPs
* If the STALL = 0 and base address is not reached then the FSM should jump into send\_req state or if STALL = 1 and base address has been reached then the FSM should jump to DONE state.
* On reset internal registers should be reset and the FSM should be in the IDLE state.

### Coverage Results:

### 

## Unit Level Testbench for EXEC unit



### Stimulus:

* PDP\_OPCODE from INSTR\_DECODE\_BFM
* EXEC\_RD\_DATA from MEMORY\_BFM
* RESET\_N from CLK\_GEN

### Checks to perform:

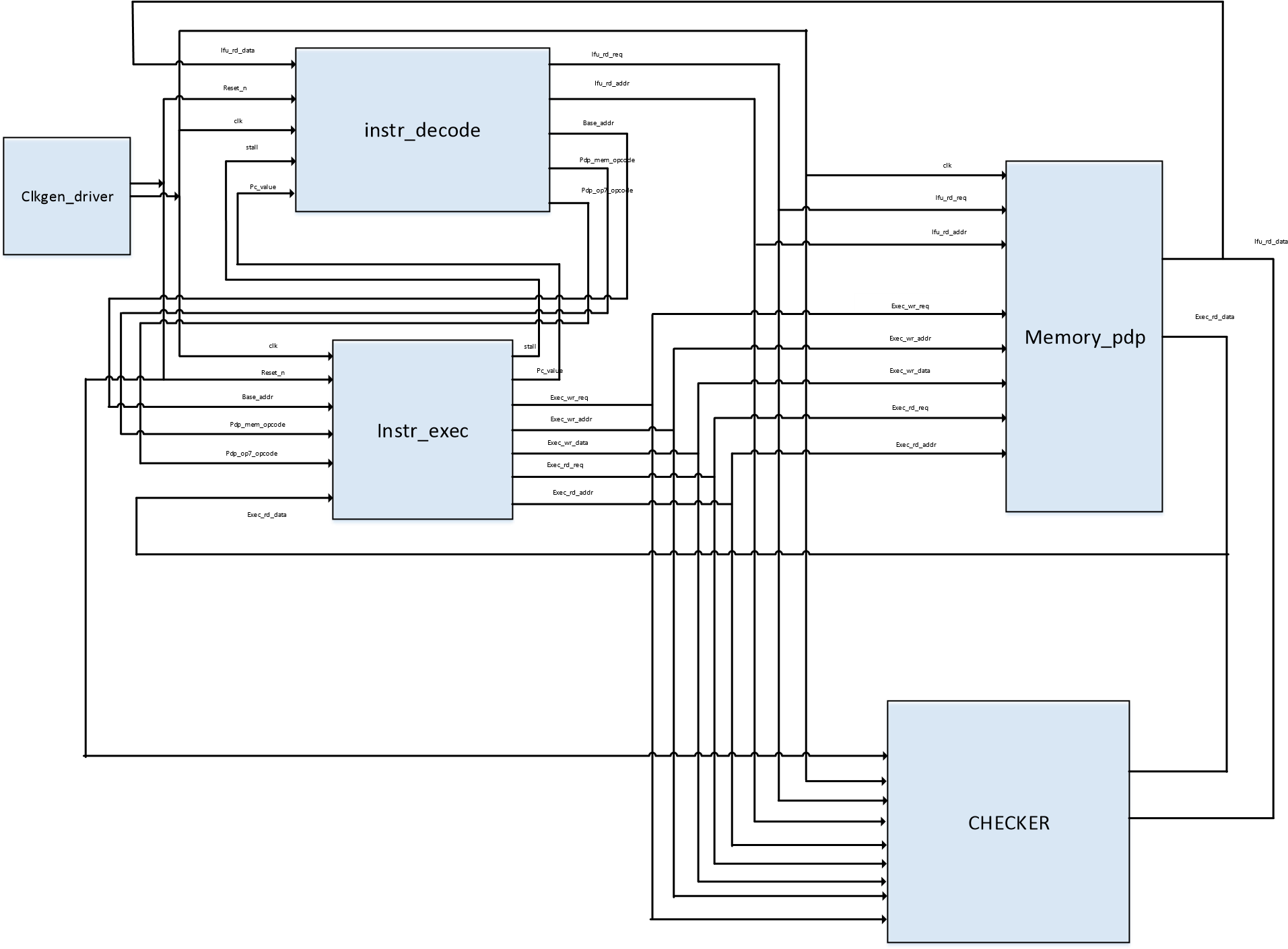
* Check if result is computed correctly
* Check if result computed is written to memory for certain instructions
* Check if LINK bit is correctly computed
* Check if STALL is asserted after beginning to execute each opcode
* Check if STALL is deasserted after completion of execution of each opcode
* Check if illegal opcodes are executed as NOP

Checks to be performed for each state transition:

|  |  |  |  |
| --- | --- | --- | --- |
| **State** | **Instruction response check** | **State transition** | **Number of cycles** |
| CLA\_CLL | Clears Acc and link register | Unstall | 1 |
| Mem\_RD\_REQ | Gets the address from the instruction and read request is sent to that ADDRESS | Data\_rcvd | 1 |
| Data\_rcvd | Data received from the memory and latched. Present  value of ACC and linker are stored in a temp reg | Jumps to different states based on the instructions/opcodes | 1 |
| ADD\_Acc\_mem | Data from the memory location from the instruction is added with the  accumulator. Carry is  obtained in the link which is  complemented. TAD is done  here | Unstall | 1 |
| AND\_Acc\_mem | Data from the memory is  ANDed with the data in the accumulator, AND operation is done here. | Unstall | 1 |
| ISZ\_Wr\_Req | Data from memory is incremented. Write add is calculated here | ISZ\_UPDT\_PC | 1 |
| ISZ\_UPDT\_PC | PC is incremented here conditionally ,ISZ is done  here | Unstall | 1 |
| DCA | Write request is asserted here and it gets the address location from the opcode.  Write data is the data in the Acc | CLA | 1 |
| CLA | ACC is cleared here | Unstall | 1 |
| JMS\_WR\_REQ | Write request is asserted here and it gets the address location from the opcode.  Write data is the PC value | JMS\_UPDT\_PC | 1 |
| JMS\_UPDT\_PC | PC is updated with the value from the PC and JMS is done  here | Unstall | 1 |
| JMP | PC is updated with the value from the PC and JMP is done  here | Unstall | 1 |
| NOP | - | Unstall | 1 |
| Unstall | Stall is cleared  Write request is cleared and  PC is incremented internally | Stall | 1 |

### Coverage Results:

# Chip level testbench



**FIGURE 5: FULL CHIP TEST BENCH**

### Stimulus:

* Stimulus: Assembly code loaded in memory

### Checks to perform:

Vertical re-use of Unit level checkers is made the full chip ,ensuring that different stimuli are provided at each level.

### Coverage Result:

# Functional Coverage at the full chip:

# Design Bugs: