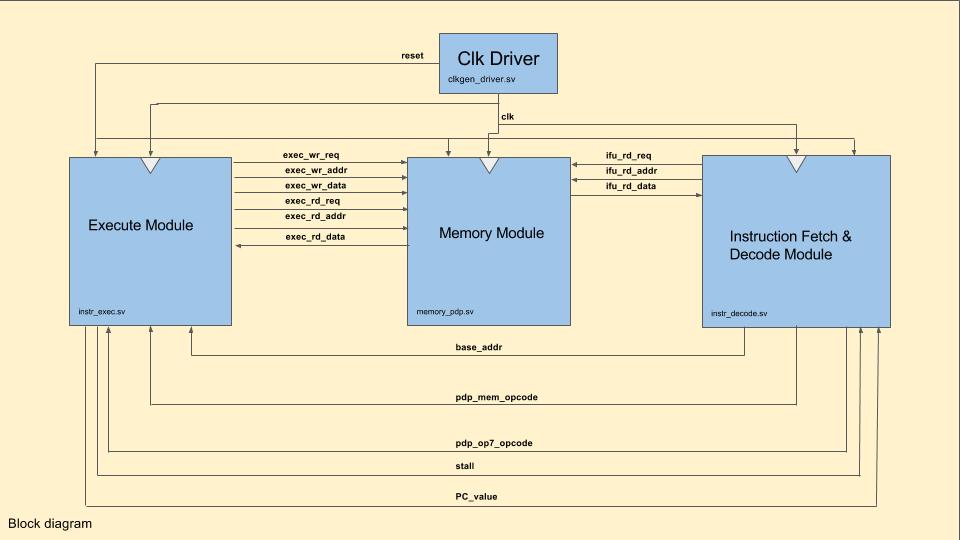
**ECE 593 / Final Project – Proposal: Team ‘P’**

**Design: Hardware implementation of PDP8 Instruction Set Architecture (ISA) level simulator**

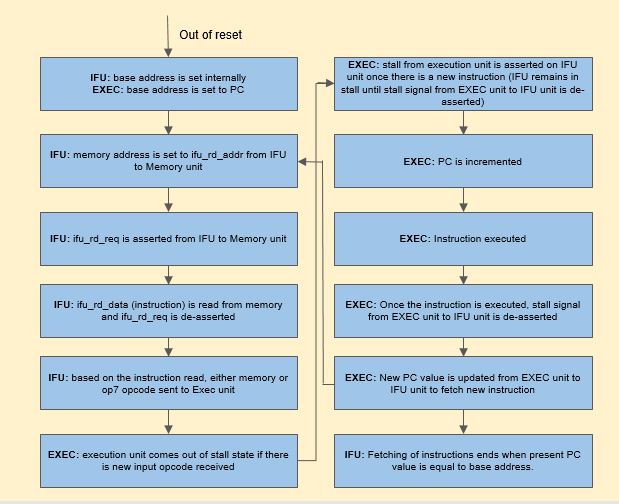
1. **Top Level Block Diagram**



The design is a set of 4 modules.

* Clock generator
* Instruction Fetch/Decode (IFD) unit
* Execution unit
* The Memory unit.

1. **Working of the Design**

****

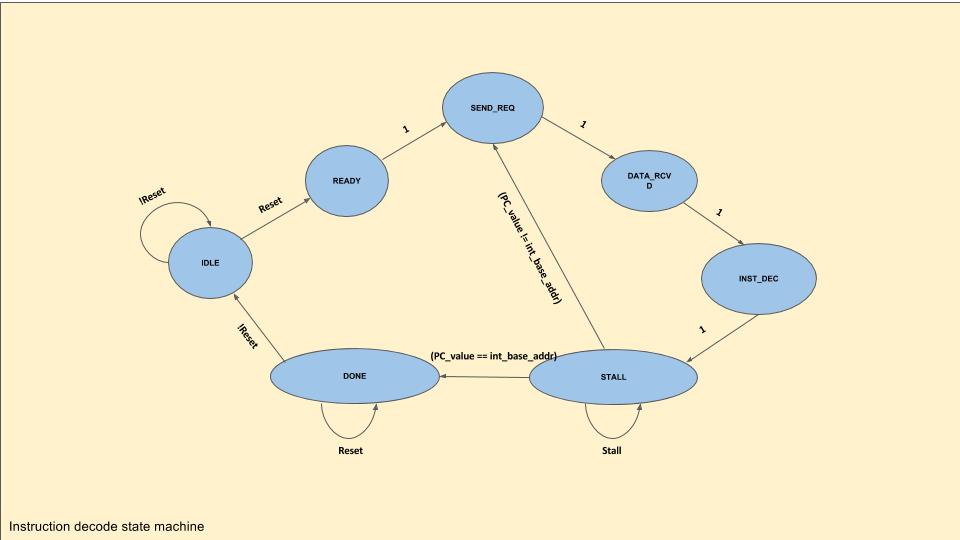
1. **Clk Driver Block**

* Clk Driver block drives continuous clock and initial reset to the remaining blocks in the design. Reset is de-asserted after predefined reset duration.
* The simulation stops after predefined runtime duration defined in the Clk Driver block.

1. **Instruction Fetch and Decode Block**

* Instruction Fetch and Decode block fetches the first instruction from a pre-determined memory location (200o).
* Once the current instruction is completed, Instruction Fetch and Decode block fetches the next instruction from a new location (based on program counter).
* The whole process is repeated until the program counter is loaded with the address of the very first instruction (i.e. 200o).

**FSM implementation of Instruction Fetch and Decode Block:**



**IDLE:**

If out of reset:

* Start address(12'o0200) is set to base address which is input to execute block.
* PDP memory opcodes (part of pdp\_mem\_opcode), PDP op7 opcodes are cleared (NOP) which are input to execute block.
* Memory instruction address (part of pdp\_mem\_opcode) is tri-stated which is input to execute block.
* Read request, Read address, Read data to/from memory block are cleared.
* In next cycle, FSM moves to READY state.

If in reset:

* Stays in IDLE state

**READY:**

* Base address is set to Read address which is input to memory block.
* In next cycle, FSM moves to SEND\_REQ state.

**SEND\_REQ:**

* PDP memory opcodes (part of pdp\_mem\_opcode), PDP op7 opcodes are cleared (NOP) which are input to execute block.
* Memory instruction address (part of pdp\_mem\_opcode) is tri-stated which is input to execute block.
* Read request is asserted which is input to memory block.
* In next cycle, FSM moves to DATA\_RCVD state.

**DATA\_RCVD:**

* Data read (Instruction) from memory to Instruction Fetch and Decode block is latched.
* Read request is cleared which is input to memory block.
* In next cycle, FSM moves to INST\_DEC state.

**INST\_DEC:**

* Decodes the instruction(12bit) read from memory block.
* If the opcode(3bit) is in range [0-5], respective PDP memory opcode is asserted with left over 9bit read data send to execute block.
* If the opcode(3bit) is 7, respective PDP op7 opcode is asserted which is input to execute block.
* If the opcode is illegal, PDP memory opcodes (part of pdp\_mem\_opcode), PDP op7 opcodes are cleared (NOP) and Memory instruction address (part of pdp\_mem\_opcode) is tri-stated which are input to execute block.
* In next cycle, FSM moves to STALL state.

**STALL:**

* The FSM will stay in this state as long the stall is asserted.
* If the stall is de-asserted, PC\_value which is input to Instruction Fetch and Decode block is set to Read address which is input to memory block from Instruction Fetch and Decode block.
  + If the PC\_value is equal to the base address which is input to execute block, FSM moves to DONE state in next cycle.
  + If the PC\_value is not equal to the base address which is input to execute block, FSM moves to SEND\_REQ state in next cycle.

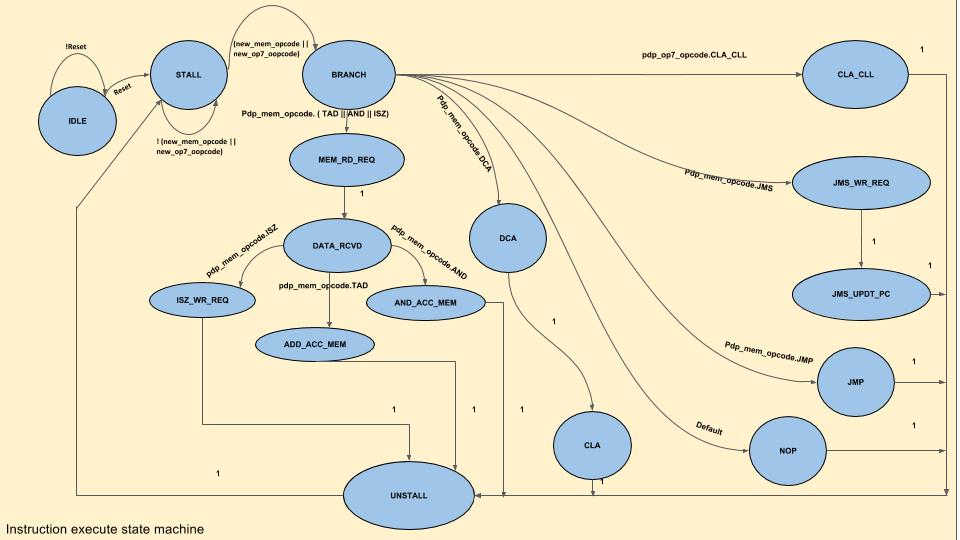
**DONE:**

* PDP memory opcodes (part of pdp\_mem\_opcode), PDP op7 opcodes are cleared (NOP) and Memory instruction address (part of pdp\_mem\_opcode) is tri-stated which are input to execute block.
* Stays in DONE state until and unless reset is asserted.
* FSM moves to IDLE state if reset is asserted.

1. **Execute Block**

* Instruction Fetch and Decode block sends the fetched instruction to execute block which processes the instruction and reads/writes from/to memory (if needed based on the instruction).

**FSM implementation of Execute Block:**



**IDLE:**

If out of reset:

* Base address which is input to Execute block is set to PC which is input to Instruction Fetch and Decode block.
* In next cycle, FSM moves to STALL state.

If in reset:

* Stays in IDLE state.

**STALL:**

If any of the memory opcodes or op7 opcodes asserted:

* Stall is asserted which is input to Instruction Fetch and Decode block.
* In the next cycle, FSM moves to BRANCH state.

If none of the memory opcodes or op7 opcodes asserted:

* Stall remains de-asserted which is input to Instruction Fetch and Decode block, read request and read address which are input to memory block from execute block are cleared to 0.
* FSM remains in the same state i.e. STALL state.

**BRANCH:**

* PC is incremented by 1.
* In the next cycle, FSM moves to respective states (CLA\_CLL or MEM\_RD\_REQ or DCA or JMS\_WR\_REQ or JMP or NOP) based on the input memory opcodes or input op7 opcodes from Instruction Fetch and Decode block.
  + If input op7 code is CLA\_CLL, FSM moves to CLA\_CLL state.
  + If input memory opcode is either TAD or AND or ISZ, FSM moves to MEM\_RD\_REQ state.
  + If input memory opcode is DCA, FSM moves to DCA state.
  + If input memory opcode is JMS, FSM moves to JMS\_WR\_REQ state.
  + If input memory opcode is JMP, FSM moves to JMP state.
  + If none of the above input memory or op7 codes, FSM moves to NOP state.

**CLA:**

* Clears the Accumulator register to 0. Link bit is unaffected. Link bit toggles if there is an overflow bit in Accumulator register.
* In the next cycle, FSM moves to UNSTALL state.

**CLA\_CLL:**

* Clears the Accumulator register and the Link bit to 0. Link bit toggles if there is an overflow bit in Accumulator register.
* In the next cycle, FSM moves to UNSTALL state.

**MEM\_RD\_REQ:**

* Read request from Execution block to memory block is asserted.
* Memory instruction address in PDP memory opcode from Instruction Fetch and Decode block, is set to Read address which is output from Execution block to memory block.

**DATA\_RCVD:**

* Read request from Execution block to memory block is de-asserted.
* Read data from memory is latched in to execution block.
* Current value of Accumulator register and Link bit are stored in temporary locations.
* Based on the input PDP memory opcodes (TAD or AND or ISZ), FSM moves to respective state for the next clock cycle.
  + If input memory opcode is TAD, FSM moves to ADD\_ACC\_MEM state.
  + If input memory opcode is AND, FSM moves to AND\_ACC\_MEM state.
  + If input memory opcode is ISZ, FSM moves to ISZ\_WR\_REQ state.

**ADD\_ACC\_MEM:**

* Data read from memory is added to Accumulator register value.
* Link bit is complimented if there is overflow bit in resultant Accumulator value.
* In the next cycle, FSM moves to UNSTALL state.

**AND\_ACC\_MEM:**

* Data read from memory is AND'ed with Accumulator register value.
* In the next cycle, FSM moves to UNSTALL state.

**ISZ\_WR\_REQ:**

* Write request asserted from execute block to memory block.
* Memory instruction address in PDP memory opcode from Instruction Fetch and Decode block, is set to Write address which is input to memory block.
* Data read from memory is incremented by 1 and set to Write data which is input to memory block from execute block.
* PC value is stored in temporary location.
* In the next cycle, FSM moves to ISZ\_UPDT\_PC state.

**ISZ\_UPDT\_PC:**

* If data read from memory is 0, PC value is incremented by 1.
* In the next cycle, FSM moves to UNSTALL state.

**DCA:**

* Write request asserted from execute block to memory block.
* Memory instruction address in PDP memory opcode from Instruction Fetch and Decode block, is set to Write address which is input to memory block.
* Accumulator register value is set to Write data which is input to the memory block from the execute block.
* In the next cycle, FSM moves to CLA state.

**JMS\_WR\_REQ:**

* Write request asserted from execute block to memory block.
* Memory instruction address in PDP memory opcode from Instruction Fetch and Decode block, is set to Write address which is input to memory block.
* PC register value is set to Write data which is input to the memory block from the execute block.
* In the next cycle, FSM moves to JMS\_UPDT\_PC state.

**JMS\_UPDT\_PC:**

* Memory instruction address in PDP memory opcode from Instruction Fetch and Decode block, is incremented by 1 and set to PC value which is input to Instruction Fetch and Decode block.
* In the next cycle, FSM moves to UNSTALL state.

**JMP:**

* Memory instruction address in PDP memory opcode from Instruction Fetch and Decode block, is set to PC value which is input to Instruction Fetch and Decode block.
* In the next cycle, FSM moves to UNSTALL state.

**NOP:**

* In the next cycle, FSM moves to UNSTALL state.

**UNSTALL:**

* Stall is cleared to 0, which is input to Instruction Fetch and Decode block.
* Write request from execute block to memory block is cleared to 0.
* PC value is updated internally.
* In the next cycle, FSM moves to UNSTALL state.

1. **Memory Block and Top Level Implementation**

* Memory block is initialized to known consecutive values.
* Memory is filled with the values read from an input ‘data file’ consisting of the input instructions compiled from an assembly test.
* Assembly test is written in separate file which is compiled using an assembler.
* Instruction Fetch and Decode block sends Read request to memory block to fetch instructions one after other.
* Instruction Fetch and Decode block sends the fetched instruction to execute block which processes the instruction and reads/writes from/to memory (if needed based on the instruction).
* Finally, memory is modified with the final computed values from execution block and written to an output file.