International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 10 Issue: 10 | Oct - 2026 SJIF Rating: 8.659 ISSN: 2582-3930

FPGA Implementation Of 8085 Microprocessor Control Unit

Dr. Sunita Vishwakar, Sharvill Deshpande, Ayush Bijwe, Atharv Kamble
Department of Electronics and Communication Engineering, Priyadarshini College of Engineering, Nagpur, India
sunitaparihar1824@gmail.com

Abstract—This paper presents an FPGA-based educational model of the Intel 8085 microprocessor control unit. The control sequence is represented using a finite state machine (FSM) designed in Verilog HDL and implemented on a PYNQ-Z2 FPGA platform. The model demonstrates opcode-dependent T-state progression and selected bus-control signals, including Address Latch Enable (ALE), active-low Read (RD_n), and active-low Write (WR_n). The supplied Vivado behavioural-simulation waveform records clock, reset, opcode, T-state and control-signal activity for five displayed opcodes: 00H, 3EH, 06H, C3H and 32H. The work is limited to a control-sequencing demonstration and does not claim to implement a complete 8085-compatible processor. The result is discussed within the evidence available from the supplied simulation capture.

Keywords—8085 microprocessor, FPGA, Verilog HDL, finite state machine, control unit, machine cycle, T-state, Vivado.

I. INTRODUCTION

A microprocessor control unit coordinates the operations required to fetch an instruction, interpret its opcode, and initiate the corresponding data transfer or execution sequence. In the Intel 8085, these operations are organised into machine cycles, each consisting of one or more T-states. The timing of these states determines when the address is latched and when data is read from or written to the system bus.

The control signals Address Latch Enable (ALE), active-low Read ($R\bar{D}$), and active-low Write ($W\bar{R}$) are essential to the bus interface. ALE identifies the address phase, while $R\bar{D}$ and $W\bar{R}$ indicate read and write operations respectively. Correct sequencing of these signals is therefore central to understanding the 8085 control unit [1], [5].

An FPGA provides a reconfigurable platform for implementing and observing digital control logic. This project represents selected 8085-style instruction sequences using a finite state machine (FSM) described in Verilog HDL. The design is intended as an educational control-sequencing model and does not claim to implement a complete 8085-compatible processor.

A. Objectives and Scope

The objectives are to represent instruction-dependent machine-cycle progression using an FSM, encode selected control outputs in synthesizable Verilog, verify the state and signal activity through behavioural simulation, and demonstrate the design on a PYNQ-Z2 FPGA board. The project scope is limited to the control sequence and selected bus-control outputs; the complete register file, ALU, program counter, memory subsystem and interrupt system are outside the present implementation.

II. TECHNICAL BACKGROUND

An FSM is defined by a finite set of states, transition conditions and output functions. In a synchronous implementation, a state register updates on an active clock edge, while combinational next-state logic determines the following state from the current state and instruction information. Reset establishes a known starting condition and prevents the sequence from beginning in an undefined state.

For this project, the FSM state represents the current step in the control sequence. A three-bit T-state output is used to expose the progression to the testbench and hardware observation interface. The output decoder associates each state with the required control-signal values. This separation of state progression and output decoding makes the RTL easier to simulate and inspect.

III. SYSTEM ARCHITECTURE

The proposed design is organised into five functional blocks: an opcode source, instruction-dependent transition logic, an FSM state register, an output decoder, and an FPGA observation interface. The opcode source supplies selected instruction values for demonstration. The transition logic selects the next state according to the current state and opcode, while the output decoder generates the T-state code and selected bus-control signals.

The target platform is the TUL PYNQ-Z2 board, which is based on the Zynq-7000 XC7Z020 device. The board provides a 125-MHz programmable-logic clock and four user LEDs [7]. A clock divider is used in the hardware wrapper to make state changes visible at the LEDs. The LED display provides a qualitative indication of activity; detailed timing relationships are examined in behavioural simulation.

A. Control Signals

The simulation observes clk, reset, opcode[7:0], tstate[2:0], ALE, RD_n and WR_n. The three-bit T-state output identifies the current timing step. ALE marks the address-latch phase. RD_n and WR_n are active-low signals, so a low logic level indicates an asserted read or write control respectively. Signal names and polarity are retained in the RTL and in the waveform.

B. FSM Operation

Let q(t) denote the current FSM state, op the opcode, and r the reset condition. The state update can be represented as q(t+1) = F(q(t), op, r). The control-output vector y(t) = [T-state, ALE, RD_n, WR_n] is generated by an output-decoding function. In the intended synchronous sequence, reset returns the state register to its initial state, and subsequent clock edges advance the FSM through the selected instruction sequence.

The opcode must remain associated with the instruction while its control sequence is active. If the demonstration wrapper changes opcodes automatically, the transition logic must sample or retain the selected opcode at an instruction boundary so that an in-progress sequence is not unintentionally altered.

IV. RTL DESIGN AND IMPLEMENTATION

The control unit is described in Verilog HDL using a finite state machine. The RTL contains the state register, reset handling, next-state transitions and output decoding. A clear separation between sequential and combinational logic improves readability and supports synthesis. Combinational processes should assign default values to outputs and next-state variables so that unintended latches are not inferred. Reset and unsupported-opcode behaviour should also be defined explicitly.

The implementation flow begins with a state/output specification. The opcode-to-sequence mapping is then described in Verilog and compiled for syntax and elaboration checks. Behavioural simulation is performed before synthesis. The design is subsequently synthesised in Xilinx Vivado for the target FPGA, followed by implementation, placement and routing, and bitstream generation. The generated bitstream is programmed onto the PYNQ-Z2 board.

A. Baseline Instruction Coverage

The supplied baseline RTL snapshot documents five opcode cases. NOP (00H) performs instruction fetch and completion. MVI A, data (3EH) and MVI B, data (06H) include an immediate-byte read. JMP address (C3H) requires acquisition of two address bytes. STA address (32H) includes address acquisition followed by a memory-write phase. The project team has separately reported a broader working subset of approximately 10–15 instructions; the additional opcode list was not present in the supplied baseline snapshot and should be reconciled with the final RTL before reporting expanded coverage.

B. Hardware Demonstration

The FPGA wrapper provides the demonstration opcode source and slows the visible state progression for LED observation. The baseline mapping exposes the T-state code and selected control information through the four user LEDs. This interface is useful for demonstrating that the programmed logic is active, but it cannot display the complete control vector or nanosecond-scale transitions. The simulation waveform is therefore the primary evidence for examining the detailed signal sequence.

V. VERIFICATION METHODOLOGY

Behavioural verification is performed with a testbench that applies clock and reset stimuli and observes the opcode, state/T-state and control outputs. The testbench should begin from a known reset condition, release reset at a defined time, apply an opcode at an instruction boundary, and sample the outputs consistently with the clock. The resulting trace is inspected to determine whether the state sequence progresses and whether ALE, RD_n and WR_n change in the intended phases.

For each supported instruction, an expected trace should specify the sequence of states or T-states and the expected control-output values. The observed waveform can then be compared cycle by cycle with the reference sequence. Additional tests should cover reset assertion and release, repeated execution of an opcode, transitions between supported opcodes, and unsupported-opcode behaviour. An opcode appearing in an RTL case statement should not be described as verified unless its expected sequence has been checked.

VI. RESULTS AND DISCUSSION

The supplied Vivado behavioural-simulation capture covers a displayed interval of approximately 0–620 ns. It shows the clock and reset signals, the opcode bus, the three-bit T-state output, and the ALE, RD_n and WR_n control signals. The opcode bus displays the sequence 00H, 3EH, 06H, C3H and 32H during the testbench run. The T-state bus shows repeated progression through the displayed state values while the control outputs change over time.

The waveform provides direct evidence that the testbench exercised the displayed opcode values and that the FSM-related outputs were active during the simulation. The active-low polarity of RD_n and WR_n must be interpreted carefully: a low level represents an asserted read or write control. The waveform should be used to inspect the ordering and duration of these signals relative to the T-state progression.

A. Result Interpretation

The simulation result supports the operation of the implemented demonstration sequence under the applied testbench conditions. It does not, by itself, establish complete cycle-by-cycle compatibility with every 8085 instruction. Such a claim requires a reference timing trace and an expected-versus-observed comparison for each supported opcode. Likewise, LUT utilisation, flip-flop count and timing slack should be reported only from the final Vivado synthesis and implementation reports; these numerical values were not included in the supplied material.

VIII. SIMULATION RESULT

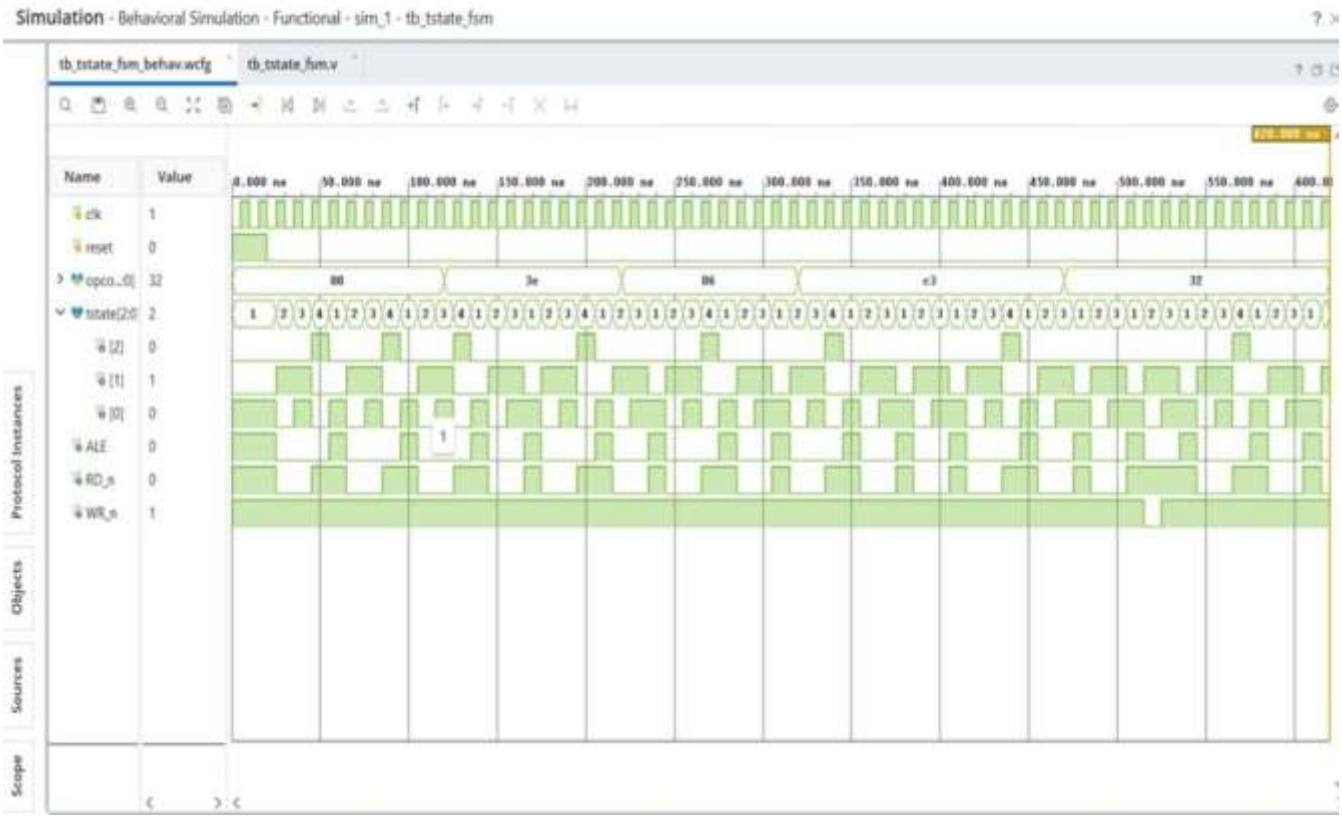


Fig. 1. Vivado behavioural-simulation waveform showing clk, reset, opcode[7:0], tstate[2:0], ALE, RD_n and WR_n over the displayed 0–620 ns interval.

VII. LIMITATIONS AND FUTURE WORK

The present work is a control-sequencing demonstration rather than a complete 8085 microprocessor. It does not implement the complete datapath, register set, ALU, program counter, memory subsystem or interrupt handling. The baseline source snapshot documents five opcode branches, while a larger working subset has been reported by the project team. The exact expanded instruction list must be checked against the final RTL revision.

The hardware wrapper uses an internally generated opcode sequence for demonstration and the LED interface exposes only a subset of the control vector. Consequently, LED activity is not sufficient evidence of detailed timing correctness. The supplied material also does not include numerical FPGA resource utilisation or timing-slack reports.

Future work should freeze and identify the final RTL revision, publish an instruction-coverage table, and prepare a reference sequence for each implemented opcode. A self-checking testbench can compare expected and observed T-state and control outputs and include reset and unsupported-opcode tests. An integrated logic analyser can improve on-board observability, while Vivado reports can provide measured resource and timing information.

VIII. CONCLUSION

This paper presents an FPGA-based educational model of 8085-style control sequencing using a Verilog finite state machine and the PYNQ-Z2 platform. The design exposes opcode-dependent state progression, a three-bit T-state code, and selected bus-control signals including ALE, RD_n and WR_n. The supplied behavioural-simulation waveform records the clock, reset, opcode and control-signal activity for the displayed testbench sequence. The project provides a practical link between microprocessor timing concepts, RTL design, simulation and FPGA implementation. Further work is required to verify every supported instruction against reference timing and to document measured implementation metrics.

ACKNOWLEDGMENT

The authors acknowledge the guidance and support of Dr. Sunita Vishwakar during the development of this FPGA-based control-unit project.

REFERENCES

[1] R. S. Gaonkar, Microprocessor Architecture, Programming, and Applications with the 8085, 6th ed. Mumbai, India: Penram International, 2013.

[2] A. P. Godse and D. A. Godse, Microprocessors and Microcontrollers, 4th rev. ed. Pune, India: Technical Publications, 2017.

[3] A. P. Godse and D. A. Godse, Digital Logic Design. Pune, India: Technical Publications, 2009.

[4] Intel Corporation, 8085AH/8085AH-2/8085AH-1 8-bit HMOS Microprocessor, 8085A Data Sheet.

[5] AMD, Vivado Design Suite User Guide: Synthesis, UG901. Available: https://docs.amd.com/r/en-US/ug901-vivado-synthesis.

[6] TUL, PYNQ-Z2 Reference Manual, ver. 1.1, Oct. 2019.

[7] E. J. McCluskey, “Minimization of Boolean functions,” Bell System Technical Journal, vol. 35, no. 6, pp. 1417–1444, Nov. 1956, doi: 10.1002/j.1538-7305.1956.tb03835.x.

MANUSCRIPT STATUS

Academic manuscript layout. Not yet published; no DOI or journal acceptance is claimed.