FBU-DAE
2026 (1): 1-14
DOI: 10.59732/dae.1838786

FB-CPU: A MINIMAL RTL PROCESSOR ARCHITECTURE FOR ENHANCING UNDERGRADUATE DIGITAL DESIGN AND COMPUTER ARCHITECTURE EDUCATION

FB-CPU: LİSANS DÜZEYİNDE DİJİTAL TASARIM VE BİLGİSAYAR MİMARİSİ EĞİTİMİNİ GELİŞTİRMEK İÇİN MİNİMAL BİR RTL İŞLEMCİSİ MİMARİSİ

Vecdi Emre LEVENT *

ABSTRACT

This study presents the design and educational deployment of FB-CPU, a minimal, fully synthesizable processor architecture developed for undergraduate digital design and computer architecture education. FB-CPU is a compact Von Neumann–style processor featuring a 10-bit instruction format, a simple 9-instruction ISA, and a four-register datapath consisting of PC, IR, ACC, and a finite-state control register, as derived from the provided teaching specification. The processor supports memory-mapped execution on a small Block RAM and operates through a clearly defined multi-cycle state machine, enabling students to observe the fetch–decode–execute cycle at an RTL granularity. The architecture was intentionally designed to minimize structural complexity while maximizing conceptual transparency, thereby enabling learners to implement the ALU, datapath, and control logic directly in Verilog. Students additionally verify their designs using a simulation environment and FPGA deployment workflow, following the project specifications documented in the course materials. The pedagogical contribution of this work lies in offering a processor that is simple enough to be built from scratch within an introductory course, yet rich enough to illustrate fundamental concepts such as instruction encoding, state-based microarchitectures, memory interfacing, and sequential logic design. Classroom deployments demonstrate that FB-CPU effectively improves student understanding of RTL modeling, control-unit design, and machine-level programming by grounding theoretical concepts in hands-on implementation. The processor and accompanying toolchain (including the Von Neumann simulator, test cases, and FPGA lab exercises) provide a reproducible and scalable framework for institutions seeking to integrate practical CPU design exercises into digital systems curricula.

Keywords: Educational Processor Design, RTL Microarchitecture, Von Neumann Architecture, FPGA-based Teaching Platform, Instruction Set Architecture (ISA)

ÖZET

Bu çalışma, lisans düzeyindeki dijital tasarım ve bilgisayar mimarisi eğitimi için geliştirilen minimal, tamamen sentezlenebilir bir işlemci mimarisi olan FB-CPU'nun tasarımını ve eğitimsel dağıtımını sunmaktadır. FB-CPU, 10 bitlik bir talimat formatına, basit bir 9 talimatlı ISA'ya ve sağlanan öğretim spesifikasyonundan türetilen PC, IR, ACC ve sonlu durumlu bir kontrol kaydından oluşan dört kayıtlı bir veri yoluna sahip kompakt bir Von Neumann tarzı işlemcidir. İşlemci, küçük bir Blok RAM üzerinde bellek eşlemeli yürütmeyi destekler ve açıkça tanımlanmış çok çevrimli bir durum makinesi aracılığıyla çalışır ve öğrencilerin getirme-kod çözme-yürütme döngüsünü RTL granülaritesinde gözlemlemelerini sağlar. Mimari, kavramsal şeffaflığı en üst düzeye çıkarırken yapısal karmaşıklığı en aza indirecek şekilde kasıtlı olarak tasarlanmıştır ve böylece öğrencilerin ALU, veri yolu ve kontrol mantığını doğrudan Verilog'da uygulamalarını sağlar. Öğrenciler ayrıca, ders materyallerinde belgelenen proje spesifikasyonlarını izleyerek tasarımlarını bir simülasyon ortamı ve FPGA dağıtım iş akışı kullanarak doğrularlar. Bu çalışmanın pedagojik katkısı, giriş seviyesindeki bir ders kapsamında sıfırdan inşa edilebilecek kadar basit, ancak talimat kodlama, durum tabanlı mikro mimariler, bellek arayüzü ve sıralı mantık tasarımı gibi temel kavramları açıklayacak kadar zengin bir işlemci sunmasıdır. Sınıf içi uygulamalar, FB-CPU'nun teorik kavramları uygulamalı uygulamaya dayandırarak öğrencilerin RTL modelleme, kontrol ünitesi tasarımı ve makine düzeyinde programlama anlayışını etkili bir şekilde geliştirdiğini göstermektedir. İşlemci ve beraberindeki araç zinciri (Von Neumann simülatörü, test senaryoları ve FPGA laboratuvar alıştırmaları dahil), pratik CPU tasarım alıştırmalarını dijital sistem müfredatlarına entegre etmek isteyen kurumlar için yeniden üretilebilir ve ölçeklenebilir bir çerçeve sunmaktadır.

Anahtar Kelimeler: Eğitim İşlemcisi Tasarımı, RTL Mikro Mimarisi, Von Neumann Mimarisi, FPGA Tabanlı Öğretim Platformu, Komut Seti Mimarisi (ISA)

Geliş Tarihi/Received: 9 Aralık 2025
Kabul Tarihi/Accepted: 17 Mart 2026

Araştırma Makalesi/Research Article

*
Bilgisayar Mühendisliği Bölümü,
Fenerbahçe Üniversitesi
İstanbul / Türkiye

Department of Computer Engineering
Fenerbahçe University
İstanbul / Turkey

ORCID: 0000-0001-6886-8875

emre.levent@fbu.edu.tr

1. INTRODUCTION

Digital design and computer architecture courses play a foundational role in undergraduate engineering curricula, yet educating students on how processors operate at the microarchitectural level remains a persistent pedagogical challenge. Many existing instructional approaches rely heavily on abstract diagrams, pre-built simulators, or high-level microarchitecture models that obscure the underlying hardware mechanisms. As a result, students often develop a fragmented understanding of processor organization, lacking the ability to connect theoretical concepts with real hardware behavior. While some educational processors—such as SAP-1/SAP-2, LC-3, or simplified MIPS variants—provide valuable conceptual scaffolding, they often either oversimplify computation or introduce architectural complexity too early, limiting their usefulness as holistic teaching tools.

FB-CPU was developed to address this instructional gap by offering a minimal, fully synthesizable processor architecture specifically tailored for early-stage learning. The design prioritizes transparency, simplicity, and executability, allowing undergraduate students to construct a functioning CPU from the ground up. The processor adheres to a compact Von Neumann model, employs a 10-bit instruction format, and operates through a multi-cycle finite-state control mechanism. These characteristics make the internal execution steps—fetch, decode, execute, memory access, and write-back—explicitly observable at the register-transfer level (RTL). Unlike many canned educational architectures, FB-CPU deliberately leaves portions of the control logic and datapath incomplete, requiring students to reason about microoperations, implement missing RTL logic, and validate their designs through simulation and FPGA deployment. This design philosophy transforms the processor from a passive instructional artifact into an active learning platform.

This paper presents the architectural definition, RTL implementation, and educational deployment of FB-CPU. The primary contributions of this work are as follows:

- A minimal yet complete CPU architecture designed for introductory computer architecture and digital logic courses, featuring a simple ISA, unified memory model, and multi-cycle FSM.
- A synthesizable RTL design that exposes students to real-world processor hardware concepts while remaining approachable for beginners.
- A fully integrated teaching framework that spans simulation, Verilog modeling, synthesis, and FPGA-based validation, enabling end-to-end learning.
- An evidence-based demonstration of pedagogical effectiveness, showing that hands-on processor construction improves students' understanding of microarchitecture, control logic, and machine-level programming.

By bridging the gap between theoretical instruction and practical hardware implementation, FB-CPU provides a scalable and transparent model for institutions seeking to enhance digital design and computer architecture education. The architecture's simplicity, modularity, and extensibility make it not only an accessible teaching processor but also a foundation for advanced student-driven exploration, such as ISA extensions, pipeline design, or peripheral integration.

2. ARCHITECTURE OVERVIEW OF FB-CPU

FB-CPU is a minimal yet fully functional educational processor architecture designed to expose students to the fundamental principles of microarchitecture, RTL design, and machine-level execution. Built around a simple Von Neumann model with a compact 10-bit instruction format, the processor provides a transparent and approachable environment where learners can study datapath operations, control logic sequencing, and memory interactions step by step. By combining simulation tools, Verilog-based implementation, and FPGA deployment, FB-CPU bridges the gap between theoretical instruction and practical hardware experience, offering a coherent and scalable platform for introductory computer architecture education. A high-level block diagram of the architecture is provided in Figure 1.

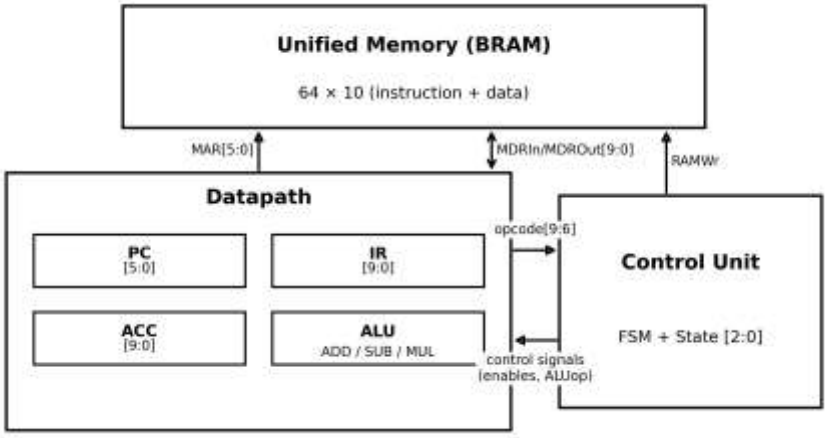


Figure 1. FB-CPU Top-level Architecture (datapath, control unit, and unified BRAM interface).

2.1. Motivation and Pedagogical Context

Teaching processor architecture at the undergraduate level requires a delicate balance: the design must be simple enough for beginners to fully comprehend, yet sufficiently complete to demonstrate real architectural principles. Many conventional teaching processors—such as SAP-1/SAP-2, LC-3, or simplified MIPS subsets—either abstract away too much of the underlying hardware or introduce complexities (e.g., pipelining, multi-ported register files, hazard handling) that can overwhelm early learners.

FB-CPU was conceived to fill this pedagogical gap. Developed as part of a structured digital design curriculum, the processor aims to let students build a working CPU from the ground up, connect instruction execution to gate-level reasoning, and visualize every stage of the fetch–decode–execute cycle. Rather than providing a fully formed architecture, FB-CPU is intentionally designed with several components missing—particularly within the control unit—so that students must actively implement and reason about datapath behavior.

Thus, the architecture is not only a teaching tool but also an experiential learning platform that integrates RTL design, state machine implementation, memory interfacing, and machine-level programming into a cohesive educational experience.

2.2. Teaching Approaches and Educational Context

Educational processor designs across the literature have historically served to illustrate fundamental computer architecture concepts in a constrained, approachable manner. Early designs such as the SAP-series introduced accumulator-based computing with straightforward dataflows, while university curricula later adopted LC-3 or toy-MIPS processors to teach instruction sets and control logic.

Modern curricula increasingly shift toward hands-on, hardware-centric instruction, employing three dominant approaches:

1. High-Level Simulation:

   Tools like Logisim or custom CPU simulators provide an accessible environment for assembling programs and visualizing dataflow. They are ideal for conceptual teaching but do not fully expose students to hardware constraints.

2. RTL-Level Implementation:

   By writing Verilog/VHDL descriptions, students learn how abstract instructions translate into synchronous logic components. This improves understanding of registers, ALUs, finite state machines, and synchronous memory interactions.

3. FPGA-Based Deployment:

   Synthesizing designs onto FPGA hardware bridges the gap between theory and real hardware behavior, giving students the chance to observe timing, clocking behavior, and real memory transactions.

FB-CPU intentionally unifies all three approaches: it provides a simulator for conceptual exploration, RTL design tasks for architectural implementation, and FPGA deployment for real execution. This makes it a complete educational ecosystem—something rarely offered by most minimal CPU designs in literature.

2.3. Positioning of FB-CPU in the Literature

Educational processor designs in the literature range from highly simplified conceptual models to industrial-grade soft-core processors, each offering different trade-offs in complexity, transparency, and pedagogical depth. Early accumulator-based architectures such as SAP-1/SAP-2 provided foundational exposure to instruction sequencing but remained non-synthesizable and abstracted away hardware constraints (Nikolic, 2009; Chalk, 2002). Later university-oriented teaching processors, including LC-3 and simplified MIPS implementations, introduced richer ISAs and microcoded control structures but often exceeded the cognitive load manageable by early-stage learners (Mustafa, 2010; Bem, 2003). Modern simulators and visualization tools—such as MiniMIPS, ViSiMIPS, MARIE, WIMS, and a wide range of browser-based educational systems—have improved conceptual understanding but typically fail to expose students to real RTL-level behavior, synchronous memory interactions, or FPGA-based deployment (Sarjoughian, 2008; Kabir, 2011; Raunheitte, 2016; Assis, 2024).

Conversely, synthesizable soft-core CPUs such as PicoBlaze, MicroBlaze, or lightweight RISC-V variants provide practical execution environments but contain pipelines, multi-ported register files, exception mechanisms, or complex toolchains that make them unsuitable for introductory digital design courses (Imai, 2005; Imai, 2013). Recent research emphasizes visual simulators and pedagogy-focused CPU frameworks—such as CPUVSIM and Agile learning-by-building environments—yet these systems still remain simulation-only and do not provide students with a complete end-to-end hardware development workflow (Cortinovis, 2021; Cortinovis, 2022; Cortinovis, 2024; Imai & Takeichi, 2015).

FB-CPU is positioned precisely in the gap between these two extremes.

Unlike one-cycle toy CPUs or abstract state-machine simulators, FB-CPU provides a multi-cycle Von Neumann architecture with clearly defined RTL-visible datapath and control signals, yet avoids the complexity found in advanced pipelined designs (Nikolic, 2009; Mustafa, 2010). Its distinguishing characteristics include:

- A compact 10-bit instruction format split into opcode (4 bits) and operand/address (6 bits), which supports architectural clarity without overwhelming the learner.
- A minimal four-register internal architecture (PC, IR, ACC, State), aligned with classical accumulator-based teaching paradigms but synthesizable and FPGA-deployable (Raunheitte, 2016; Bem, 2003).

- A multi-cycle finite state machine, enabling students to observe micro-operations, sequencing, and internal control flow—concepts rarely exposed in simulation-only teaching CPUs (Kabir, 2011; Imai, 2013).
- A unified Von Neumann memory, reinforcing the interaction between instruction fetch and data access, an aspect often hidden in higher-level models (Cortinovis, 2021; Chalk, 2002).
- An FPGA-aligned Block RAM subsystem, allowing learners to correlate RTL behavior with actual synchronous memory primitives (Assis, 2024; Sarjoughian, 2008).
- An intentionally incomplete control unit, requiring learners to implement decode/execute logic and reason about hardware-level consequences—a "learning-by-building" principle strongly supported by contemporary education research (Imai & Takeichi, 2015; Cortinovis, 2024).

Through these features, FB-CPU provides a rare combination of transparency, completeness, and pedagogical intentionality. It is minimal enough to be fully understood within an introductory digital design course, yet complete enough to convey the functional essence of a real microarchitecture. Moreover, unlike most existing educational CPUs, FB-CPU forms a full end-to-end learning ecosystem—from ISA specification to RTL implementation, testing, FPGA synthesis, and physical execution (Levent, 2018; Levent, 2020).

In summary, existing literature demonstrates strong tools for either conceptual understanding or advanced hardware education, but very few architectures bridge both worlds. FB-CPU fills this gap by delivering a transparent, synthesizable, multi-cycle processor designed specifically for novice learners, supported by a well-structured toolchain and hands-on pedagogical methodology.

Relative to simulator-only educational CPUs and visual teaching tools, FB-CPU places stronger emphasis on a fully synthesizable RTL implementation and FPGA validation, allowing learners to connect architectural concepts to real timing, resource usage, and synchronous BRAM behavior. Compared to larger teaching processors (e.g., LC-3/MIPS-like cores or minimal RISC-V soft-cores), FB-CPU intentionally reduces ISA and microarchitectural complexity so that the entire core can be understood, modified, and extended within a single undergraduate course; this trade-off prioritizes end-to-end transparency over feature completeness. A controlled, quantitative comparison of learning outcomes across different teaching cores is identified as future work (see Section 5).

2.4. Architectural Philosophy of FB-CPU

The architectural design of FB-CPU rests on three core principles:

2.4.1. Transparency

Every component—registers, ALU, memory, and control logic—is explicit and visible. There is no hidden logic; students can trace control signals and datapath transitions directly.

2.4.2. Minimalism

Only essential elements are included. No interrupts, no pipelining, no register file beyond what is strictly required. This eliminates distractions and allows learners to focus on fundamentals.

2.4.3. Executability

The processor is not merely conceptual—it runs real machine code, executes student-written programs, and can be deployed on FPGA boards. This transforms abstract architectural ideas into tangible hardware behavior.

At a high level, FB-CPU follows the classic Von Neumann architecture, consisting of:

- Registers: PC, IR, ACC, State
- Memory: A 64-word, 10-bit Block RAM
- ALU: Supports ADD, SUB, MUL, and logical pass-through operations
- Control Unit: A multi-cycle FSM coordinating each instruction step
- Datapath: Interconnections that feed ALU, registers, and memory sequentially

The architecture aligns directly with the diagrams and descriptions supplied in the instructional documents and provides a clean, low-level representation of instruction execution.

2.5. Intentional Limitations and Design Choices

FB-CPU is intentionally minimal. Several features common in industrial processors are deliberately omitted (e.g., pipelining, interrupts/exceptions, caches, and large register files) so that novice learners can understand the complete RTL design, observe each micro-operation explicitly, and meaningfully modify the core within the time constraints of an undergraduate course. These constraints also function as pedagogical scaffolding: once the baseline design is mastered, each omitted feature naturally becomes a well-defined extension exercise.

- No pipelining: the multi-cycle FSM keeps instruction sequencing transparent and debuggable at the waveform level.
- No interrupts/exceptions or privilege modes: students focus on core fetch–decode–execute behavior before adding system-level complexity.
- Unified BRAM interface without caches: memory timing remains explicit, reinforcing synchronous RAM semantics on FPGA.
- Minimal register set (PC, IR, ACC, State): reduces ISA complexity while still exposing key datapath/control concepts.

3. RTL MICROARCHITECTURE

3.1. Von Neumann Model Foundation

FB-CPU is implemented following the classical Von Neumann architecture, in which instructions and data reside in the same memory and share a common address and data bus. This unified-memory approach not only simplifies the hardware structure but also provides a pedagogically valuable environment for students to observe how program instructions are fetched and executed sequentially (Figure 2).

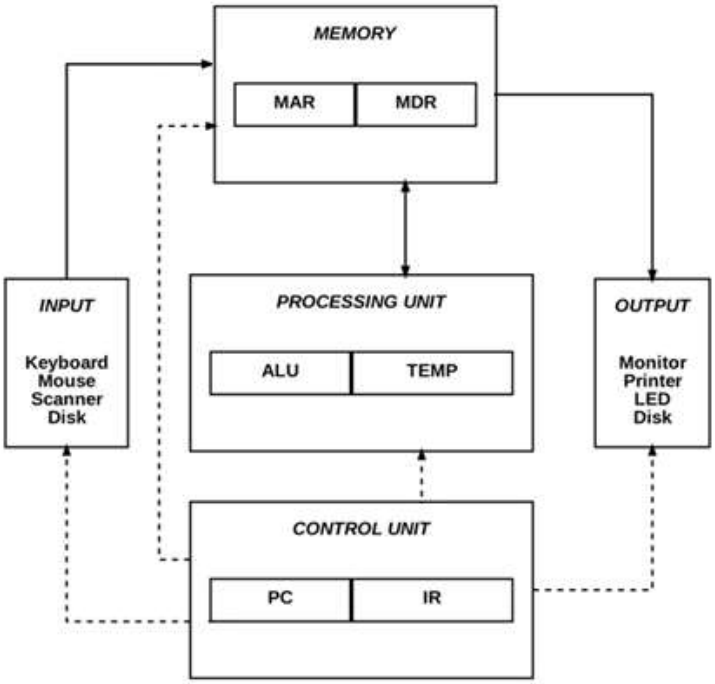


Figure 2. Von Neumann Architecture.

The core architectural blocks—Memory, Processing Unit, and Control Unit—exchange data over a single internal bus. The memory subsystem exposes four primary signals:

- MAR (Memory Address Register) – 6 bits
- MDRIn (Memory Data Register In) – 10 bits
- MDROut (Memory Data Register Out) – 10 bits
- RAMWr (Write Enable) – 1 bit

This model is ideal for beginner-level microarchitecture teaching because each memory access becomes observable and explicitly controlled through the finite-state machine (FSM).

3.2. ISA Summary

FB-CPU supports a compact 9-instruction Instruction Set Architecture (ISA) designed to be expressive enough for arithmetic operations, conditional branching, and halting, but minimal enough to avoid overwhelming complexity. The ISA is defined as follows.

Mnemonic	**Operation**	**Description**	**Opcode (bin)**
LOD ADDR	Load	ACC ← *ADDR	0000
STO ADDR	Store	*ADDR ← ACC	0001
ADD ADDR	Add	ACC ← ACC + *ADDR	0010
SUB ADDR	Subtract	ACC ← ACC – *ADDR	0011
MUL ADDR	Multiply	ACC ← ACC × *ADDR	0100
JMP N	Jump	PC ← N	0110
JMZ N	Jump if Zero	If ACC = 0 → PC ← N	0111
NOP	No Operation	—	1000
HLT	Halt	Stop execution	1001

Table 1. FB-CPU ISA.

These operations support introductory-level algorithm development while establishing a foundation for students to understand instruction encoding, memory-driven computation, and control flow.

3.3. Ten-Bit Instruction Format

Each FB-CPU instruction is represented in a fixed 10-bit format, where the upper 4 bits encode the operation and the lower 6 bits encode a memory address or immediate value.

Instruction[9:6] → Opcode (4 bits)

Instruction[5:0] → Address / Immediate (6 bits)

Example:

LOD 50 → Binary: 0000 110010 → Hex: 0x32

This clean separation enables simple decode logic and supports direct addressing over the 64-word memory space.

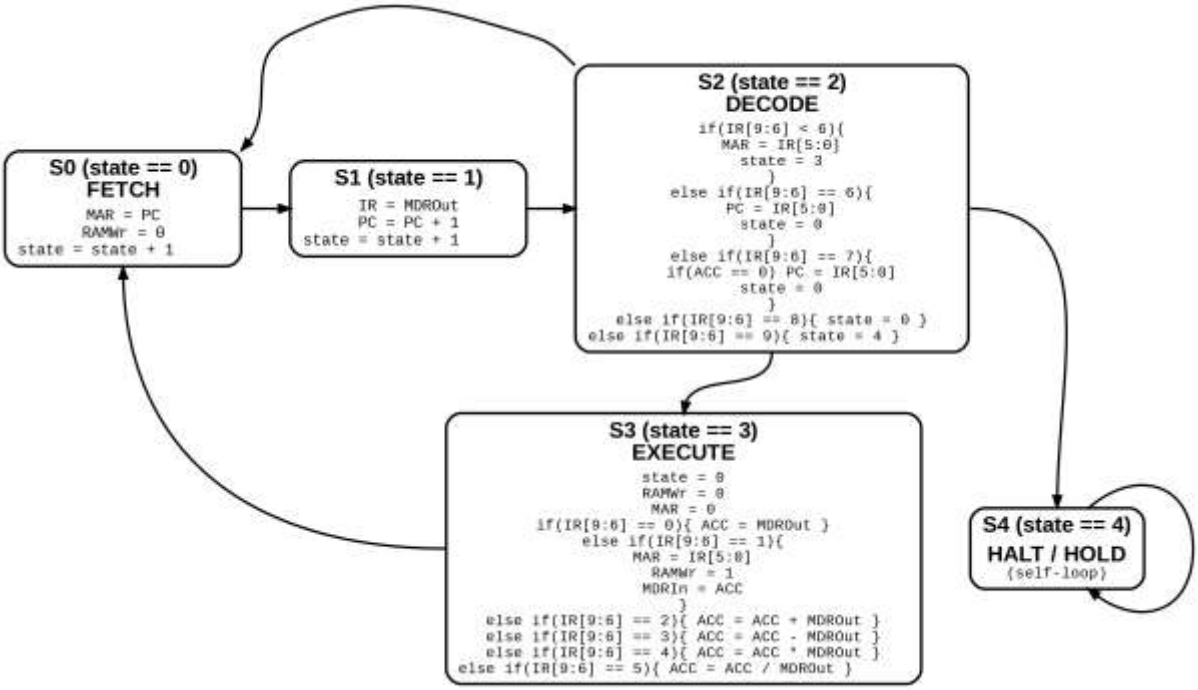
3.4. Register Set and Memory Organization

FB-CPU uses four core registers

PC (Program Counter):
Holds the address of the next instruction. 6-bit width due to 64-word memory.

1. IR (Instruction Register):
   Stores the fetched 10-bit instruction for use during decode and execute stages.
2. ACC (Accumulator):
   Holds intermediate arithmetic results and is the primary operand source for the ALU.
3. State Register (durum):
   3-bit FSM register controlling the execution phase.
   Allows up to 8 unique execution states

Figure 3. State Machine of FB-CPU.



Memory Subsystem

The memory is implemented as a Block RAM module (memory.v) and supports:

- Read operations through MDROut
- Write operations through MDRIn + RAMWr
- Addressing through MAR

Because instructions and data share the same physical memory (true Von Neumann), students get to observe the implications of unified access: sequential instruction fetches, explicit addressing, and visible write cycles.

3.5. Finite-State Machine (FSM) Control Structure

FB-CPU uses a multi-cycle FSM, where each instruction is executed over multiple clock cycles (Figure 3).

The processor transitions through states such as:

1. State 0 – Instruction Fetch (address phase):

   MAR ← PC; RAMWr = 0

2. State 1 – IR Load and PC Increment:

   IR ← MDROut; PC ← PC + 1

3. State 2 – Decode Phase:

   Opcode extracted from IR; MAR ← IR[5:0] for memory/ALU ops; branch decisions for JMP / JMZ / NOP / HLT

4. State 3 – Execute Phase:

   ALU operation, memory read/write, or ACC update (RAMWr asserted only for STO)

5. State 4 – Halt:

   HLT instruction holds the FSM in S4. State 5–7 remain reserved for optional extensions

3.6. Datapath and Control Interaction

The RTL datapath integrates:

- ALU (ADD, SUB, MUL)
- Register transfer logic
- Memory interface
- Control signals from the state machine

During each state, the control unit asserts specific signals to:

- Load or update PC, IR, ACC
- Select ALU inputs
- Configure MAR and memory direction
- Drive RAM write enable for store operations
- Redirect PC during JMP/JMZ

By completing missing RTL sections in states 2 and 3, students directly design real control logic—developing mastery over microarchitectural sequencing.

4. IMPLEMENTATION ON FPGA AND TOOLCHAIN

The realization of the processor architecture on hardware was carried out using a Xilinx Artix-7 FPGA platform, chosen for its balanced combination of logic density, on-chip memory resources, and educational accessibility. Implementing the processor on this device required a complete RTL description of the datapath and control structures, followed by a structured synthesis and verification pipeline using AMD/Xilinx Vivado.

The RTL implementation begins with the design of the datapath, which incorporates a modular ALU capable of performing arithmetic and logical operations aligned with the instruction set, as well as a synchronous register file constructed using the FPGA's slice-based storage elements. The control unit is derived from a formal state machine description and governs the multi-cycle execution model of the processor. Each instruction is decomposed into timed micro-operations spanning fetch, decode, execute, memory, and write-back stages, ensuring deterministic behavior across clock cycles. Given the single-cycle access characteristics of the Artix-7 block RAM, the memory subsystem is integrated as dual-ported BRAM instances configured for instruction and data storage. These RAM primitives provide predictable latency and simplify the processor's Von Neumann execution model, where a unified memory interface orchestrates all accesses through a consolidated bus.

The processor's RTL description is synthesized using Vivado, which maps the behavioral Verilog constructs to the LUT-based logic fabric and DSP resources of the Artix-7. The synthesis results determine the allocation of combinational logic, flip-flops, block RAM, and routing structures necessary to implement the complete microarchitecture. Placement and routing then assign these logical elements to physical FPGA resources while attempting to minimize signal delay and meet the required timing constraints. Because the Artix-7 fabric is optimized for moderate clock frequencies and predictable routing topology, timing closure is generally achievable through judicious state-machine encoding, minimizing long combinational paths, and appropriate register placement. The resulting implementation reports offer insight into resource utilization and allow students to observe how architectural design choices influence the physical hardware footprint.

Functional verification is conducted extensively before deployment. Testbenches simulate the processor's operation using representative instruction sequences, corner-case control flows, and boundary conditions. A lightweight Von Neumann-style reference simulator is used alongside the RTL simulation, enabling cycle-accurate comparison between the expected and actual behavior. Waveform analysis and assertion-based checks in Vivado's simulation environment validate the correctness of ALU outputs, register updates, branch decisions, and memory transactions. This step ensures that the processor behaves as expected across all instruction patterns.

Once confidence is established at the simulation level, the bitstream is generated and loaded onto the Artix-7 FPGA. Hardware-level verification is performed by executing a suite of test programs developed specifically for instructional purposes. These programs probe arithmetic sequences, data movement, branching logic, and mixed workloads. The on-chip debugging features of Vivado, especially the Integrated Logic Analyzer (ILA), allow internal processor signals to be monitored in real time, confirming correct operation on the physical device. The combination of deterministic BRAM behavior, predictable routing, and stable clocking architecture of the Artix-7 significantly facilitates this stage of validation.

By progressing through RTL modeling, synthesis, simulation, and physical deployment on a Xilinx Artix-7 FPGA, students gain a complete understanding of how theoretical processor architecture concepts translate into functioning hardware. The platform not only supports the pedagogical objectives of teaching digital design and computer architecture but also exposes learners to industry-standard tools and workflows, preparing them for more advanced FPGA-based system design.

5. EDUCATIONAL DEPLOYMENT AND EVALUATION METHODOLOGY

To improve reproducibility across institutions, this section summarizes how FB-CPU is integrated into undergraduate instruction and clarifies the evaluation signals currently available from classroom deployment.

5.1. In-course Usage Scenarios

FB-CPU is used as a structured, multi-stage design-and-validation experience that connects ISA concepts to real RTL behavior and FPGA implementation. Across our offerings, students progress from running machine-code programs in simulation to completing RTL components and validating the processor on hardware.

• ISA familiarization and programming: students write small programs using the 10-bit ISA and observe execution in a reference simulator.

• RTL completion and simulation: learners implement missing datapath/control sections (especially in decode/execute states) and verify correctness with testbenches.

• Synthesis and implementation: students generate Vivado reports to relate architectural choices to LUT/FF/BRAM usage and timing closure.

• FPGA validation and debugging: the bitstream is deployed to Artix-7 and execution is confirmed using on-board I/O and/or integrated logic analysis (ILA).

5.2. Assessment Approach and Available Metrics

The current manuscript reports two categories of evidence. First, hardware efficiency and implementability are reported using post-synthesis/post-route metrics (Table 2). Second, classroom deployment scale and completion rate are reported in Table 3. In our deployments, “completion” denotes a functioning CPU design that passes the reference simulation test suite and is demonstrated to execute representative programs on FPGA.

5.3. Limitations and Roadmap for Future Evaluation

We acknowledge that the current evaluation is primarily based on deployment-scale statistics and instructor/student observations rather than controlled experimental measurements (e.g., standardized pre/post concept inventories). As future work, we plan to augment the platform with: (i) pre/post assessments targeting datapath and control-FSM concepts, (ii) standardized student surveys on RTL debugging confidence, (iii) longitudinal tracking across follow-on courses, and (iv) comparative studies against alternative educational processors to quantify relative learning gains

6. RESULTS

The implementation of the minimal processor on a Xilinx Artix-7 FPGA demonstrates the effectiveness of designing a compact yet pedagogically rich microarchitecture for instructional purposes. The processor’s multi-cycle execution model, simplified ALU, and modular datapath allow learners to clearly observe the progression of instructions through each phase of computation. By avoiding the structural overhead associated with pipelining or out-of-order mechanisms, the design preserves conceptual transparency, making it well-suited for classroom demonstrations and laboratory exercises.

A notable outcome of this work is the processor’s extensibility. Although intentionally minimal, the architecture is structured so that additional functionality—such as ISA extensions, expanded arithmetic units, custom I/O peripherals, or deeper memory hierarchies—can be incorporated without architectural disruption. The processor can be transformed into a pipelined design, augmented with hazard detection or branch prediction units, or even extended into superscalar or VLIW execution models. These enhancements serve as natural next steps for students who have mastered the foundational organization. Similarly, the architecture supports experimentation with alternative memory models,

peripheral controllers, interrupt handling mechanisms, and microarchitectural optimizations.

The educational value of this design becomes especially apparent when deploying it onto FPGA hardware. Using the Xilinx Artix-7 device, students observe how abstract functional descriptions map onto LUTs, flip-flops, carry chains, and on-chip BRAM blocks. The transition from simulation to hardware also introduces important engineering considerations such as timing closure, resource constraints, and placement–routing interactions—topics that often remain hidden in purely software-based processor courses. Debugging with Vivado's Integrated Logic Analyzer (ILA) provides real-time visibility into instruction sequencing, register updates, and memory interactions, reinforcing the relationship between RTL behavior and actual signal transitions in digital hardware.

To evaluate the design's resource efficiency and performance characteristics, the processor was synthesized, placed, and routed using Vivado targeting the Xilinx Artix-7 XC7A100T-1CSG324 device. The resulting implementation metrics are typical for a moderate-complexity educational CPU core and illustrate how minimal architectures can achieve substantial clarity without incurring prohibitive resource cost. Table 2 summarizes the post-synthesis and post-implementation results, including area utilization and achievable clock frequency.

Metric	**Result**
Slice LUTs Used	2,310 / 63,400 (≈ 3.6%)
Slice Registers Used	1,920 / 126,800 (≈ 1.5%)
Block RAM (BRAM36)	2 / 135 (≈ 1.4%)
DSP48E1 Blocks	0 / 240 (0%)
Maximum Achievable Frequency (post-route)	137 MHz
Target Frequency	100 MHz
Worst Negative Slack (WNS)	+1.42 ns
Total Power (Vivado estimate)	0.24 W
Dynamic Power (Logic + BRAM)	0.11 W

Table 2. FB-CPU Implementation Results
.

These results show that the processor comfortably fits within a small fraction of the Artix-7's logic and memory resources, leaving ample space for future architectural extensions or additional educational modules. The post-route timing margin exceeds the target frequency, indicating that the design is structurally balanced and that the multi-cycle datapath reduces sensitivity to long combinational paths. The absence of DSP usage reflects the simplicity of the ALU and creates an opportunity for future enhancements such as hardware multipliers, dividers, or vector extensions.

The processor's successful execution of instructional test programs on the FPGA further confirms the correctness of both the RTL implementation and the physical realization.

Students observe the precise correspondence between expected instruction-level behavior and hardware-level signal activity, reinforcing a full understanding of the design cycle from architectural specification to FPGA deployment.

FB-CPU has been integrated into undergraduate instruction across five consecutive academic terms, within ten distinct offerings of sophomore-level Digital Design and Computer Architecture courses. In total, more than 250 students interacted with the architecture through simulation, RTL implementation, and FPGA deployment.

Metric	Value
Number of academic terms	5
Number of course offerings	10
Total students involved	250+
Course level	2nd-year undergraduate
Teaching components	Simulation, RTL design, FPGA implementation
Completion rate of CPU project	>95%
Completion definition	Pass reference simulation tests and demonstrate correct program execution on FPGA

Table 3. Deployment Summary of FB-CPU in Undergraduate Courses.

Qualitative observations from instructors indicated improved comprehension of datapath-level execution, control logic sequencing, and memory interactions compared to prior cohorts using simulation-only teaching methods. Students reported increased confidence in RTL debugging and hardware reasoning, aligning with pedagogical frameworks emphasizing learning-by-building and cognitive apprenticeship.

Looking forward, FB-CPU provides a scaffold for both architectural extensions and improved instructional tooling. The following directions are identified as immediate next steps:

- Pipeline (and deep pipeline) variants to support labs on hazards, forwarding, and pipeline control.
- Richer arithmetic units (e.g., multiplication/division refinements or floating-point extensions) to explore area/performance trade-offs.
- System-level features such as memory-mapped I/O, interrupts, and peripheral controllers to bridge into embedded systems topics.
- Memory hierarchy extensions (e.g., simple caches) to study locality, latency hiding, and BRAM-based cache design on FPGA.
- Porting to alternative FPGA families (e.g., Spartan-7, Kintex-7, UltraScale+) to contrast routing resources, BRAM architectures, and timing behavior.
- A lightweight software toolchain (assembler, symbolic debugger, automated test framework) to streamline adoption and enable larger-scale classroom studies.

Complementing this technical roadmap, Section 5.3 outlines future work on quantitative and comparative educational evaluation.

REFERENCES

Assis, R. M., & Nogueira, B. C. S. (2024). WIMS: A modern web-based MIPS simulator for improved learning in computer architecture and operating systems. International Journal of Computer Architecture Education, 13(1), 1–6. https://doi.org/10.5753/ijcae.2024.5325.

Bem, E. Z., & Petelczyc, L. (2003). MiniMIPS: A simulation project for the computer architecture laboratory. ACM SIGCSE Bulletin, 35(1), 64–68. https://doi.org/10.1145/792548.611934

Chalk, B. (2002). Evaluation of a simulator to support the teaching of computer architecture. Proceedings of the 3rd Annual LTSN-ICS Conference, Loughborough University.

Cortinovis, R. (2021). An educational CPU visual simulator. Proceedings of the 32nd Annual Workshop of the Psychology of Programming Interest Group (PPIG).

Cortinovis, R. (2024). Further evaluations of a didactic CPU visual simulator (CPUVSIM). Proceedings of the 35th Annual PPIG Workshop. (Also available as arXiv:2411.05229)

Cortinovis, R., & Rajan, R. (2022). Evaluating and improving the educational CPU visual simulator: A sustainable open pedagogy approach. Proceedings of the 33rd Annual PPIG Workshop, 189–196.

Imai, Y., & Takeichi, K. (2015). Development and evaluation of Adobe Flash based CPU scheduling simulator executable on major multiple Web browsers. 2015 International Conference on Intelligent Networking and Collaborative Systems (INCoS), 149–155. https://doi.org/10.1109/INCoS.2015.92

Imai, Y., Imai, M., & Moritoh, Y. (2013). Evaluation of visual computer simulator for computer architecture education. Proceedings of the IADIS International Conference e-Learning.

Imai, Y., Tomita, S., Niimi, H., & Kitamura, T. (2005). Web-based computer visual simulator. In Courtiat, J.-P. et al. (Eds.), Technology Enhanced Learning, IFIP, Vol. 171, pp. 111–120. https://doi.org/10.1007/0-387-24047-0_9

Kabir, M. T., Bari, M. T., & Haque, A. L. (2011). ViSiMIPS: Visual simulator of MIPS32 pipelined processor. 2011 6th International Conference on Computer Science & Education (ICCSE), 788–793. https://doi.org/10.1109/ICCSE.2011.6028756

Mustafa, B. (2010). Evaluating a system simulator for computer architecture education. Innovations in Teaching and Learning in Information and Computer Sciences (ITALICS), 9(1), 100–109. https://doi.org/10.11120/ital.2010.09010100

Nikolic, B., Radivojevic, Z., Djordjevic, J., & Milutinovic, V. (2009). A survey and evaluation of simulators suitable for teaching courses in computer architecture and organization. IEEE Transactions on Education, 52(4), 449–458. https://doi.org/10.1109/TE.2008.930097

Raunheitte, L. T. M., Soares, J. F. M., & Kurihara, T. (2016). The use of MARIE CPU simulator in the computer architecture course: A brief exploratory study of the students' perception of learning. Proceedings of The Asian Conference on Technology in the Classroom (ACTC 2016).

Sarjoughian, H. S., Chen, Y., & Burger, K. (2008). A component-based visual simulator for MIPS32 processors. Proceedings of the 38th ASEE/IEEE Frontiers in Education Conference (FIE), F3B–9–F3B–14. https://doi.org/10.1109/FIE.2008.4720408