CPU on FPGA – Nand2Tetris Implementation

The *Nand2Tetris project*, formally known as *The Elements of Computing Systems* by Noam Nisan and Shimon Schocken, is a foundational course in computer systems design that introduces learners to building a complete computer from the ground up. It uniquely integrates concepts from logic design, computer architecture, and systems programming using a structured bottom-up approach, beginning with the simplest logic gates and culminating in the development of a functioning central processing unit (CPU) and software stack.

This project extends the *Nand2Tetris* philosophy by implementing its hardware architecture using a Field Programmable Gate Array (FPGA). By taking the conceptual designs from the course and translating them into synthesizable Verilog code, this project not only reinforces theoretical understanding but also develops practical skills in hardware description languages, FPGA toolchains, and real-world hardware constraints. This serves as an important step toward deploying Hack Assembly programs and eventually interactive applications, such as a simplified Tetris game, on physical hardware.

Objectives

To ensure a structured and measurable approach, the project sets out clear objectives that guide its progression from simple logic design to system-level implementation. These objectives outline not only the functional milestones—such as developing the Hack CPU and preparing for program execution—but also the broader goal of gaining practical proficiency with FPGA workflows and Verilog-based hardware design. They serve as a roadmap, ensuring that each phase of development builds meaningfully toward creating a deployable computing system aligned with the Nand2Tetris specification.

- **To implement all hardware modules defined in Part I of the Nand2Tetris course using Verilog**, following a bottom-up approach that builds from simple logic components to complex subsystems.
- **To deploy the designed hardware architecture on an FPGA platform**, thereby demonstrating a working implementation of the Hack computer system in real hardware.
- **To gain experience in writing synthesizable Verilog code** and working with FPGA development workflows in Vivado.
- **To prepare for the execution of Hack Assembly programs** on the FPGA and lay the groundwork for running an interactive application as a final demonstration.

Tools and Technologies

- **FPGA Platform:** Basys 3 Artix-7 FPGA Trainer Board, selected for its robust feature set and compatibility with academic development workflows.

- **Hardware Description Language:** Verilog, used for its synthesizability and industry relevance in digital design.
- **Design and Simulation Environment:** Vivado 2024.2, incorporating XSIM for module-level simulation and debugging.

Project Scope and Methodology

This project closely follows the structure of *Nand2Tetris* Part I, implementing each hardware module in Verilog and testing it using custom testbenches. The work is organized as follows:

Project 1: Foundational Logic Design
This project focused on building the essential building blocks for the Hack computer architecture. Instead of relying solely on Verilog operators, care was taken to design these circuits in a modular way that mimics hardware construction principles.

- **Core Logic Gates:**
  - Implemented basic single-bit gates such as NAND, AND, OR, XOR, and NOT to establish a foundation for all higher-level components.
  - Ensured each gate was written in a form compatible with FPGA synthesis.
- **Multi-Bit Gate Extensions:**
  - Designed 16-bit versions of common gates (e.g., And16, Or16, Not16), enabling operations on word-sized inputs across the architecture.
- **Data Selection Components:**
  - Developed **multiplexers** (1-bit, 16-bit, 4-way, and 8-way) to route data conditionally.
  - Designed **demultiplexers** to distribute signals efficiently, forming the basis for memory addressing.

This project established the primitives necessary for both data manipulation and routing, forming the conceptual and structural foundation for subsequent stages.

Project 2: Arithmetic and Processing Unit
This project focused on developing the arithmetic and processing capabilities of the Hack computer using a **hierarchical design approach**—building complex modules by integrating previously designed simpler components.

- **Adders:**
  - **Half-Adder & Full-Adder:** Implemented for single-bit binary addition.
  - **16-Bit Adder:** Constructed using a cascade of full-adders, enabling multi-bit arithmetic.
- **Incrementer:**
  - Designed to add one to a 16-bit value, used in program counter operations and address progression.
- **Arithmetic Logic Unit (ALU):**
  - Developed a 16-bit ALU capable of performing arithmetic (addition, incrementation) and logical (AND, OR, NOT, negation) operations.

    - Incorporated control signals to dynamically alter computation behavior (e.g., zeroing, negating inputs).

  By combining these modules hierarchically, this stage delivered the processing core of the Hack computer, aligning with the modular philosophy of the Nand2Tetris design.

Project 3: Sequential Components and Memory
This project implemented all sequential building blocks required for state retention, memory management, and program execution in the Hack architecture.

- **Single-Bit Storage:**
    - **Bit:** Basic 1-bit storage element using clocked logic, serving as the foundation for all multi-bit registers.
- **Registers:**
    - **Register:** 16-bit storage for holding data or intermediate results across clock cycles.
- **Hierarchical Memory Modules:**
    - **RAM8, RAM64, RAM512:** Built using registers and multiplexers to provide scalable memory with hierarchical addressing.
    - **RAM4K, RAM16K:** Expanded capacity using modular design principles, combining smaller RAM units into larger blocks.
- **Program Counter (PC):**
    - Developed a 16-bit counter with increment, load, and reset functionalities to support sequential instruction execution and branching.

  This project gives the Hack computer the ability to store and retrieve instructions/data reliably while enabling sequential control flow, forming a critical part of the system's execution pipeline.

Project 4: Instruction Set & Program Execution
This project focuses on validating the Hack architecture through program execution and planning for instruction set support in the Verilog-based CPU.

- **Program Execution in Hack:**
    - **fill.asm:** Continuously fills the screen with black when a key is pressed, or white otherwise.
    - **mult.asm:** Multiplies two numbers using repeated addition.
    - These programs were **implemented and successfully tested on the Hack platform**, verifying the correctness of instruction execution.
- **Planned Instruction Set Implementation:**
    - While these specific programs will not be directly ported, the Verilog-based CPU is being designed to **support the full Hack instruction set architecture (ISA)**, enabling similar assembly-level programs to run on FPGA hardware in the future.

  This stage bridges the gap between conceptual program execution in the Hack simulator and preparing for real-world execution on an FPGA-based CPU.

Project 5: The Computer Architecture (CPU)
Currently in progress: constructing the Hack CPU by integrating the ALU, instruction decoding logic, control unit, and program counter. Includes instruction execution and memory interface.

Project 6: Machine Language Programs
Once the CPU is complete, programs written in Hack Assembly will be used to test real execution flow. This phase sets the foundation for running a Tetris game or similar application on FPGA hardware.

Challenges Encountered

Learning Verilog and Adapting to Hardware-Oriented Thinking

As a first-time Verilog user, transitioning from software-based problem-solving to hardware description presented a steep learning curve. Unlike sequential software execution, hardware design requires understanding concurrency, timing, and resource constraints. Developing synthesizable code demanded adopting a hardware-centric mindset and refining coding practices to meet synthesis requirements.

Simulation and Testbench Development

Constructing effective testbenches to verify module correctness was a significant learning experience. Simulation in Vivado provided critical feedback, helping to identify logic errors early in the design process. These experiences emphasized the importance of structured testing and the need for iterative refinement before hardware synthesis.

Exploring CPU Integration

While full CPU integration has not yet been achieved, initial work on the design of control logic and program counter integration has provided valuable insights. This exploration has revealed potential challenges related to coordinating multiple subsystems and ensuring precise signal timing across the datapath, which will inform the next stages of development.

Key Learnings

1. Hardware-Oriented Thinking:
   Transitioning from software programming to hardware design required developing a new mindset — understanding concurrency, propagation delays, and clock-driven behavior instead of sequential program flow.
2. Hierarchical Design Principles:
   Building complex modules like the ALU and memory blocks from smaller components reinforced the importance of modularity, reusability, and layered abstraction in hardware design.
3. Verilog Proficiency:
   Gained hands-on experience with synthesizable Verilog, including structuring combinational vs sequential logic, managing control signals, and writing code suited for FPGA deployment.
4. Instruction Set Awareness:
   Working with Hack Assembly programs (e.g., fill.asm and mult.asm) improved understanding of instruction formats, execution flow, and the relationship between

hardware architecture and program behavior.

5. FPGA Workflow Familiarity:
   Exposure to Vivado's design, simulation, and synthesis workflows helped in understanding practical considerations for moving from theoretical designs to real-world hardware implementations.
6. Debugging & Iterative Development:
   Identifying and resolving design issues at each stage of the project highlighted the need for systematic verification and iterative refinement in digital system development.

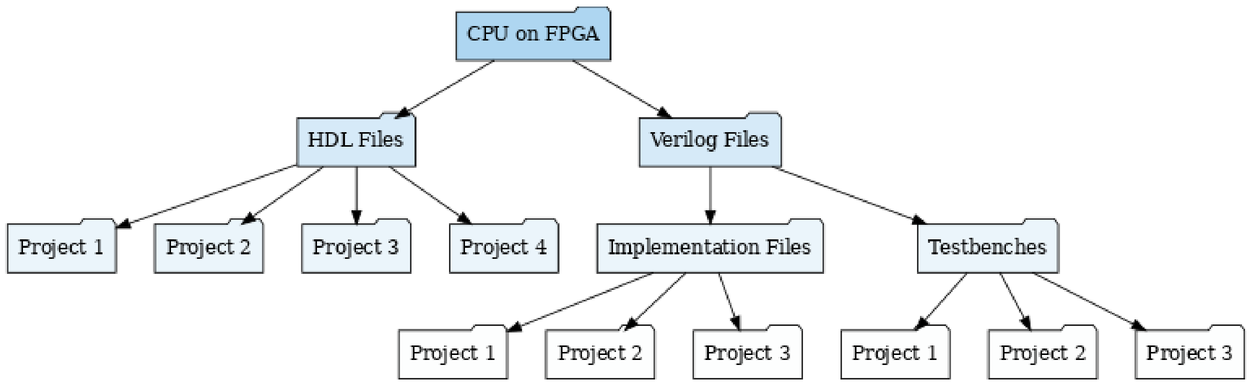
Future Work

- **Complete CPU Development:** Integrate the ALU, control logic, and program counter into a functional CPU.
- **Performance and Resource Optimization:** Explore ways to reduce logic utilization and optimize timing for FPGA deployment.
- **Interactive Demonstration:** Develop and deploy a simplified Tetris game as a capstone demonstration.

GitHub Repository

https://github.com/rishtirkulkarni/CPU-on-FPGA/tree/main

This repository serves as a central workspace for documenting and tracking the progress of our CPU on FPGA (Nand2Tetris) project. It includes Verilog implementations, Hack platform programs, and supporting documentation.



This repository is used to store our Verilog implementations, Hack programs, and progress documentation for the CPU on FPGA (Nand2Tetris) project.

Acknowledgments

I would like to extend my sincere gratitude to **Adrian P Isaac**, our alumni student coordinator, for their continuous support the course of this project.
A special thanks to **Anish Krishnakumar**, our mentor, whose expertise, constructive feedback, and encouragement were instrumental in helping us approach this project with clarity and purpose.

I am also deeply appreciative of my teammates **A V Sohan Aiyappa** and **Virajit G P** for their collaboration, shared enthusiasm, and collective problem-solving, which made this endeavor both productive and enjoyable.
Additionally, I acknowledge the support of MARVEL UVCE – https://hub.uvcemarvel.in – for providing us with the resources and an enabling environment for hands-on learning and innovation.
Finally, I would like to express my gratitude to Noam Nisan and Shimon Schocken, the creators of the Nand2Tetris course, for designing a framework that inspired this project and made the journey of building a computer from first principles truly engaging.

References

1. Nisan, N., & Schocken, S. (2005). *The Elements of Computing Systems: Building a Modern Computer from First Principles*. MIT Press.
2. Xilinx. (2024). *Vivado Design Suite User Guide*. Retrieved from https://www.xilinx.com/support.html.
3. Basys 3 FPGA Trainer Board Reference Manual. Digilent Inc.
4. Samir Palnitkar – Verilog HDL : A Guide To Digital Design And Synthesis