Vaishnav Prasad Ravilichitty Kishor

+91 8825694490 • rkvp.in • rkvaishnavp@gmail.com • linkedin.com/in/rkvaishnavp • github.com/rkvaishnavp

Education

- **B.E. Electronics & Instrumentation - BITS Pilani, K.K.Birla Goa Campus** **October 2020 - May 2024**
  Relevant Coursework:- Computer Architecture, Analog and Digital VLSI Design, Digital Design, Operating Systems, OOPs, Network Programming, Microprocessor Programming and Interfacing

Work Experience

NXP Semiconductors (formerly Kinara AI) June 2024 – Present
System Hardware Engineer (Design Verification Engineer) Hyderabad, Telangana

- Joined Kinara AI in June 2024; continued as System Hardware Engineer (Design Verification Engineer) following NXP Semiconductors' acquisition in October 2025.
- Developed a SystemVerilog-based constrained-random verification environment for a reconfigurable processor to auto-generate C testcases derived from the instruction model, executed on the ISS to validate custom instruction functionality across all configurations and corner scenarios.
- Verified custom TIE-defined load/store/move instructions by monitoring functional coverage of opcodes, operands, internal states, and address ranges, and debugging mismatches through ISS result comparison and waveform analysis.
- Generated and analyzed instruction-level functional coverage metrics across opcode variations, operand combinations, and exception scenarios to ensure verification completeness and achieve coverage closure.
- Root-caused instruction mismatches and behavioral anomalies via detailed waveform inspection, log correlation, and bit-accurate ISS comparison in close collaboration with architects and designers.
- Maintained automated regression infrastructure with coverage and test result tracking across nightly and on-demand runs to ensure functional correctness and design stability throughout the reconfigurable processor development cycle.

Technical Skills

- **Languages**: UVM, SystemVerilog, Verilog, C, C++, Python
- **Protocols**: AXI4, UART
- **Tools**: Questasim, Xilinx Vivado, Xtensa Xplorer

Projects

RISC-V CPU Core github.com/rkvaishnavp/YARC

Verilog, Xilinx Vivado, Artix A7

- Designed a single-cycle Non-Pipelined CPU Core with RV32I ISA in Verilog HDL, then designed an RV32I-based 5-Stage Pipelined Core.
- Also, developed an AXI4-Lite Compatible Slave Memory Module & Interface to the core, which will be used as IMem and DMem for the RISC-V Core.

Hand Written Digit Recongnition in FPGA github.com/rkvaishnavp/SingleLayerMNIST

Verilog, Digital Design, Xilinx Vivado, Artix A7

- Trained a Neural Network with perceptron neurons in Python for Handwritten digit recognition (used MNIST Dataset).
- Currently working on accelerating the digit recognition neural network inference on FPGA. The weights and biases data of the neurons are sent to FPGA using UART communication, which is then stored in BlockRAM. Subsequently, image data is sent again through UART, which is then passed to the accelerator module in FPGA for parallel computing to recognize the digit. The recognized number is then sent back to the computer via UART.

Display and VGA Core github.com/rkvaishnavp/Display-and-VGA-Core

Verilog, Python, Digital Design, Xilinx Vivado, Artix A7

- Created a Display Core in Verilog which generates VGA signals (VGA Standards: red, green, blue, hsync, vsync) in 4-bit format RGB444.
- The core takes a mem file as the image source in binary uncompressed format (This was generated using the Image Module Python package, which takes a cropped image (640x480) as input).
- To test if the core is generating the signal according to the standards, a simple decode unit was written in Verilog HDL, which takes VGA signals and dumps the data to the log file. Python scripts were used to compare the input image and output data to check if the core is generating the correct signals.

Simple Adder Core using UART and Blockram github.com/rkvaishnavp/uart-core
Verilog, Digital Design, Xilinx Vivado, Artix A7

- Designed a simple core in verilog which gets two number input using UART interface in FPGA(the data to the uart is sent from the python code in the PC) and the data received store in Blockram in FPGA
- Once both the numbers are stored in Blockram, we read it from the block ram and add the two numbers and store it the blockram and then send it back to the PC using UART interface

Custom System Call Addtion to LINUX Kernel 5.4.0 github.com/rkvaishnavp/OS-Project
C, Linux Kernel Compilation, Makefile

- Completed as a part of the CS F372 Operating Systems course at BITS Goa.
- Added a custom system call to Linux Kernel 5.4.0 for adding two float numbers.
- Successfully recompiled the kernel and tested the system call, adding two float numbers in C.

Simple Device Driver github.com/rkvaishnavp/OS-Project
C

- Added a Simple Device Driver. If the laptop is charging, Screen brightness will be set to 100% else the brightness percentage will be set equal to battery percentage