Syed Noorul Saad

saadpiece.com · Waterloo, ON · sn3syed@uwaterloo.ca · (226) 989 5840 · saads312

Education

University of Waterloo — Waterloo, ON
B.A.Sc Computer Engineering — Sep 2022 - May 2027

- Relevant courses: Reconfigurable Computing (Master's Level), Real Time Operating Systems, Digital Hardware Systems, Computer Architecture, Compilers, Embedded Microprocessor Systems

Technical Skills

Languages: SystemVerilog, Verilog, C/C++, Python, Assembly (ARMv7, RISC-V)
Protocols: AXI4/AXI4-Lite, SPI, UART
Tools: Vivado (synthesis, STA, P&R), Quartus, CocoTB, GTKWave, Git, Linux
Concepts: RTL design, timing closure, clock domain crossing (CDC), pipelining/retiming, FSMs, high performance design, design for verification, formal verification (Lean4)

Experience

FPGA Research Assistant — Waterloo, ON
University of Waterloo - Under Prof. Mina Arashloo — Jan 2026 - *Present*

- Contributing to a protocol-agnostic FPGA **transport-layer acceleration** architecture on **AMD Alveo U250**, enabling hardware offload of multiple network protocols (TCP, RoCEv2)
- Implementing **PCIe/QDMA-based** host ↔ FPGA communication using AXI-Stream within the OpenNIC 250 MHz user logic, allowing applications to drive FPGA transport logic
- Building buffering, backpressure, and request-framing logic to reliably inject host application requests into a high-throughput FPGA networking pipeline

Digital Design Team Lead – Ethernet Packet Parser — Waterloo, ON
UW ASIC Design Team — Aug 2025 – *Present*

- Leading a team of **15 members** to architect a high-throughput **Ethernet packet parser** from scratch, owning system architecture, RTL implementation, and block-level verification across the full design lifecycle
- Designed **multi-layer packet parsing pipeline** in SystemVerilog, supporting Ethernet/IP/TCP header extraction with configurable match-action rules and line-rate throughput targeting
- Led RTL implementation and **floorplanning** of a cryptography accelerator, optimizing PPA to achieve **200 MHz** operation at **65% FPGA resource utilization**
- Architected an **ACK-based bus arbitration protocol** using open-drain signaling and encoded request lines, reducing average arbitration latency by **4 cycles** under multi-client contention
- Designed and verified an **SPI peripheral** with robust **CDC synchronization** using flip-flop chains; developed CocoTB constrained-random testbenches achieving **>90% functional coverage**

Technical Projects

BERT Encoder on AMD Versal Hard-NoC (FPGA) *SystemVerilog, Tcl, AMD Versal, NoC, AXI*

- Architected and integrated BERT-style encoder datapath on AMD Versal achieving **1.5 GB/s aggregate NoC bandwidth** for DDR-PL communication, emphasizing scalable data movement over compute
- Implemented and verified **multi-head self-attention (4 heads)** using AXI-Stream + DMA pipelines, reducing memory bandwidth requirements by **4x** through INT8 quantization of activations and weights
- Developed parameterized system operating at **300 MHz** supporting configurable tokens, embedding size, and head count, with enforced constraints for systolic tiling, AXI burst alignment, and memory efficiency

High-Frequency Adders on Agilex-5 FPGA *SystemVerilog, Quartus Prime Pro, TimeQuest, Bash*

- Optimized 2048-bit adder architecture achieving **41% frequency improvement** (600→848 MHz) and **25% area reduction** (10.9K→8.2K ALMs) through design-space sweep selecting M=16 Kogge-Stone configuration
- Eliminated all memory LAB usage (**100% reduction** from 700 blocks to 0), improving LAB utilization by **31%** (1,758→1,211 LABs) through optimized ALM packing
- Automated design-space exploration sweeping **11 configurations** (M=2 to M=2048, powers of 2) using Bash scripts to identify optimal frequency-area tradeoff

UVM Design Verification Environment (Data Aligner) *SystemVerilog*

- Built a comprehensive **UVM testbench** for a data aligner module in SystemVerilog, implementing a full agent class with **driver, monitor, and scoreboard** components for repeatable and thorough verification
- Achieved **100% code coverage** and **95% functional coverage** through constrained-random stimulus generation