Personal Github://lennoxho Tel: (+1)416-788-4307

# **WORK EXPERIENCE**

#### **EFINIX** | Software Developer, MTS

Feb 2024 - Current | Toronto, ON Canada

- Software developer for the **Efinity EDA** software
- Project lead for flows, databases and code Infrastructure (FDI). Responsibilities include:
  - Design and implementation of high level compiler control flows
  - Profiling, monitoring and pathfinding for runtime, memory and load time improvements
  - Setup and maintenance of local **SLURM compute farm**

## INTEL PSG / ALTERA | Software Developer, Grade 7

June 2016 - Dec 2023 | Toronto, ON Canada

- Software developer for the Intel Quartus Prime EDA software
- Co-owner of Quartus common code infrastructure. Responsibilities include:
  - Design and implementation of internal debugging and profiling tools and core in-house libraries
  - De facto engineering representative for infrastructure & OS support discussions
- Maintainer of Quartus database and design flows. Responsibilities include:
  - Design and implementation of custom data structures for netlists and ancilliary data

# **ACHIEVEMENTS**

- In the first year at Efinix,
  - Reduced Efinity synthesis runtime by 70% through a series of micro-optimisations, algorithm reformulation to enable parallelism
  - Reduced Efinity place & route runtime by 40% through a series of micro-optimisations, data reorganisation for better locality
  - Reduced Efinity peak memory by 3x, greater for typical usage scenarios. Dropped Efinity memory requirements from 32GB to 12GB
  - Reduced Efinity load time by 5x, from up to 20 minutes down to 3 minutes on largest designs
  - Replaced the **Efinity SDC parser** with a compliant Tcl interpreter
  - Trivialised wildcard processing time in SDC scripts by implementing an SSE-optimised wildcard matching engine
  - Set up a local **SLURM compute farm** to support more QoR experiments
- During tenure at Altera,
  - Fixed a system library defect blocking Quartus support on Windows 11. Departmental Recognition Award (DRA).
  - First-principles rewrite of Quartus in-house memory profiler (qmp). Profiling overhead reduced by 10x compared to previous effort
  - Rewrote Quartus in-house database parser/generator/serialisation library by leveraging libClang to parse C++ code
  - Clean room implementation of DWARF debugging format (v2-5) decoder. Tailored to qmp's runtime and memory requirements
  - Designed and implemented an Electron-based GUI for qmp. Power tool to analyse memory usage across a process lifecycle
  - Fixed performance of open-source TCMalloc library on Windows, yielding a 10% global reduction in Quartus runtime
  - Reduced build time of Quartus binaries by 4x (3 hours), devkit setup time on Windows by 6x (50 minutes)
  - Designed a novel quasi-succinct fixed-data hash table to store prebuilt and constant associative data in Quartus device databases
  - Designed a novel mutex methodology using thread local storage, used to implement semi-lock-free data structures

# PROGRAMMING LANGUAGES | SKILLS

#### **PROFICIENT**

- C++ 23 C Python
- Concurrent & parallel programming
  Lock-free programming
  Synchronisation primitives
- Microsoft Windows Linux Template Metaprogramming x86 SIMD programming Instruction-level optimisation

#### **FAMILIAR**

• D • Rust • Javascript • Java • x86 assembly

#### **Tools**

• MSVC • gdb • libClang • LLVM • Boost • Intel TBB • libunwind • CMake • Electron • Node-API

## **EDUCATION**

## UNIVERSITY OF TORONTO | BSc in Computer Science (Hons.)

Specialising in Scientific Computing | CGPA 3.95/4.0

# **PROJECTS**

## **QUARTUS WINDOWS 11 COMPATIBILITY**

- Critical defect found late in the development cycle affected all Quartus GUI applications on Windows 11
- Used trampoline hooking in TCMalloc to work around problematic Windows runtime behaviour
- Worked with Intel TBB maintainers to evaluate possible mitigations in **TBBMalloc**. Implemented **import address** table (IAT) hooking in TBBMalloc
- Recognised with a Departmental Recognition Award (DRA)

## TCMALLOC PERFORMANCE ON WINDOWS

- TCMalloc is a high performance allocation library developed by Google
- TBBMalloc was used on the Windows build of Quartus, as TCMalloc appeared to offer lower performance
- Identified the reason for TCMalloc slowdown on Windows and implemented a fix, improving TCMalloc performance on Windows by 10%
- Upstreamed fixes and enhancements to the TCMalloc public repository

# QUARTUS MEMORY PROFILER (qmp) V2

- Redesigned the Quartus in-house **memory profiling tool** from first principles as a completely standalone tool
- Achieved 10x reduction in runtime overhead while memory overhead is reduced by 2x
- Typical runtime overhead of memory profiling is reduced to 35% compared to no profiling
- This speedup was made possible using heavily modified copies of **libunwind** and **TCMalloc**, as well as custom **lock-free data structures** ensuring wait-free operation
- Enabled memory profiling to be successfully performed on the entire **Quartus competitive benchmarking design** suite with a **24 hour turnaround time**
- $\bullet$  Features custom variable-length encoding format and data deduplication efforts to minimise size of results ( $\approx 50$  MBs / hour of collection)

#### DWARF DEBUGGING FORMAT DECODER

- Clean room implementation of **DWARF** debugging format decoder with no  $3^{rd}$  party dependency
- Versions 2-5 (most recent as of writing) of the debugging format are supported with the exception of split and supplementary-DWARF
- Also exposed qmp-addr2line as a GNU addr2line clone

# **QMP GUI**

- Designed and implemented an **Electron**-based GUI to display **qmp** profiling results
- Custom Node-API-based backend to perform data marshalling and heavy number crunching in C++
- Shared memory used in backend to avoid duplication of profile data across processes
- Heavily customised popular Javascript data visualisation libraries for real time performance on GBs of compressed data
- Implemented a **Level-of-detail** (LOD) system for the **Dygraphs** library to exceed the recommended 10,000 data point limit
- Features a timeline (stacked) chart of memory usage broken down by module over the profiling duration
- A Flame chart representing the call stack is overlayed and synchronised with the timeline (memory) chart. This allows Quartus developers to track the lifecycle of any byte of memory
- Fully interactive, including zooming and panning support

#### FLOWGRAPH-DRIVEN PARALLEL SERIALISATION LIBRARY

- Trivialised Quartus netlist database loading time by designing and implementing a **flowgraph-based serialisation library**
- A flowgraph is used to represent the Quartus data dependency graph, which when fed to a parallel computing library can yield huge reductions in loading time

## **QUARTUS DATABASE SCHEMA PARSER V2**

- Quartus' in-house **serialisation library** ships with a tool to automatically **generate serialisation routines** (schemas) based on C++ code
- This parser/generator tool featured a handwritten C++ parser, which was fragile and frequently broke whenever a new compiler was introduced
- Redesigned and rewrote the parser/generator tool to leverage libclang as its C++ parser
- This change improved the accuracy of the generated schemas while ensuring the long term viability of the library

# QUASI-SUCCINCT FIXED-DATA HASH TABLE

- Designed and implemented a **novel quasi-succint hash table** data structure tailored for **precomputed data**
- Open-addressing & linear-probing based, where the worst case probe length was the size of the largest bucket
- In memory compression (inspired by Elias-Fano encoding) was used to store hash table indices. x86 SIMD instrinsics and jump tables were used to perform fast bucket offset lookup
- Invented as a solution to eliminate wastage when traditional associative data structures were used for fixed data

## HYBRID SHARED MUTEX & SEMI-LOCK FREE DATA STRUCTURES

- Designed to overcome the traditional limitations of conventional lock-free data structures, chiefly the need for individually-allocated nodes
- While various optimisations exist, node-based data structures generally have a high memory overhead
- Hybrid shared mutex leverages thread-local-storage to provide very cheap read locks due to its hardware cache advantages over shared atomics
- Semi-lock free data structures may use read locks to perform atomic operations on a fixed size bufer, while a write lock is used to increase the size of the buffer
- The ability to use **fixed size bufers** in semi-lock free data structures allows for more **compact implementations of high performance**, **thread-safe data structures**

# **QUARTUS TCL VIRTUAL FILESYSTEM**

- Implemented a virtual filesystem layer to provide the Quartus Tcl interpreter direct access to Quartus databases
- This greatly improves the project migration (export/import) experience by packaging all **user SDC and Tcl scripts** directly into the **Quartus databases**