Shou Hao (Lennox) Ho

Email:// lennoxhoe@gmail.com LinkedIn:// "Lennox" Shou Hao Ho

Personal Github:// lennoxho Work Github:// holenno1

Tel: (+1)416-788-4307

WORK EXPERIENCE

INTEL PSG (FORMERLY ALTERA) | Senior Software Developer, Grade 7

June 2016 - Current | Toronto, ON Canada

- Software developer for the Intel Quartus Prime software

- Co-owner of Quartus **common code infrastructure**. Responsibilities include:
  - Design and implementation of **internal debugging and profiling tools**
  - Design and implementation of **core in-house libraries** and integration of **third party libraries**
  - Lead **compiler upgrade** (MSVC, GCC and Clang) efforts on the (≈ 30 **million LOC**) Quartus codebase
  - Maintain the **overall stability and health** of the Intel Quartus Prime software
  - 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
  - Ensure correct and efficient **exchange of data** between Quartus modules
  - **Orchestration** of Quartus processes

ACHIEVEMENTS

- Fixed a **critical defect** blocking Quartus **support on Windows 11**. Departmental Recognition Award (DRA). More...

- Complete redesign and rewrite of Quartus in-house **memory profiler** (qmp), **runtime** reduced by **10x**, **memory** by **2x**. More...

- Clean room implementaton of **DWARF** debugging format (v2-v5) **decoder**. Tailored to qmp's runtime and memory requirements. More...

- Designed and implemented an **Electron**-based **GUI** for qmp. Power tool to analyse memory usage across a process lifecycle. More...

- Reduced **build time** of Quartus binaries by **4x**. Integration test runtime reduced by 3 hours

- Reduced Quartus **devkit setup time** on Windows by **6x** (1 hour to 10 minutes)

- Fixed performance of **TCMalloc** on Windows, yielding a **10% global reduction** in Quartus **runtime**. More...

- Complete redesign and rewrite of Quartus in-house **database schema parser/generator**, leveraging **libclang** to parse C++ code. More...

- Designed and implemented a **flowgraph-driven parallel serialisation** library to trivialise Quartus netlist database loading time. More...

- Rewrote Quartus **wildcard assignment processing** engine, achieving a **10x speedup** . More...

- Migrated Quartus netlist databases from **memory mapping to heap memory**, yielding a **2x** reduction in **virtual memory** usage

- Spearheaded **ABI and runtime compatibility** checking for Quartus as an engineering-driven approach of ensuring product stability on a wide range of platforms

- Designed a **novel quasi-succinct fixed-data hash table** to store associative data in Quartus device databases. More...

- Designed a **novel mutex methodology** using thread local storage, used to implement **semi-lock-free data structures** that are signifcantly more scalable than contemporaries. More...

PROGRAMMING LANGUAGES | SKILLS

PROFICIENT

- **C++ 20** • **C** • **Python**
- **Microsoft Windows** • **Linux**

- Concurrent & parallel programming • Lock-free programming
- Systems programming • Low-level optimisations • Synchronisation primitives
- Template Metaprogramming • Code & assembly-level debugging

FAMILIAR

- **D** • **Rust** • **Javascript** • **Java**

- x86 assembly • x86 SIMD programming

TOOLS

- MSVC • gdb • Clang • 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

LEN'S HOT TAKES

• Weekly blog series on Intel PSG's internal forums, doing deep dives into obscure topics such as the implementation of thread local storage, debugging runtime issues etc

• "-Bsymbolic and std::string" explores how a module built with -Bsymbolic and the old libstdc++ ABI (uses copy-on-write strings) causes weird crashes

• "The mystery of the __std_type_info_name crash" explores how using an 8-byte aligned allocator results in an instruction-level fault

• Since December 2020

PROJECTS

QUARTUS WINDOWS 11 COMPATIBILITY

• A critical defect was found late in the development cycle affecting 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 Windows, 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**

• Features custom variable-length encoding format and data deduplication efforts to minimise size of results (≈ 50 MBs / hour of collection)

DWARF DEBUGGING FORMAT DECODER

• Clean room implementation of DWARF debugging format decoder with no 3rd party dependency

• **Versions 2-5** (most recent as of writing) of the debugging format are supported with the exception of split and supplementary-DWARF

• Caches decoded data in memory for fast lookup from qmp while minimising memory overhead

• 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

QUARTUS WILDCARD ASSIGNMENT PROCESSING

• Rewrote Quartus' **handling of wildcard assignments**

- Achieved a **10x speedup** by leveraging the **hierarchical nature of netlists** and how that relates to the **"greedy"** nature of **simple wildcard patterns**

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**

COMPILER AND C++ STANDARD UPDATES FOR QUARTUS

- Led efforts to update Quartus to MSVC 2019 and 2022, GCC 12 and Clang 15

- Led efforts to upgrade the (≈ 30 **million LOC**) **Quartus codebase** to **C++14, 17, then 20**

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**

CLANG ONBOARDING FOR QUARTUS

- Fixed the (≈ 30 **million LOC**) **Quartus codebase** over multiple releases to be **compatible with the Clang compiler**

- Clang compliance is a **prerequisite** for a number of **Clang-based static analysis tools** such as SourceTrail and ClangTidy

- This also enables Quartus to be built with the **Clang compiler** or the **Intel ICX compiler**