Chung-Shu Chen 陳鍾樞 (03)6681193, 0970577923 gamma_chen@yahoo.com.tw

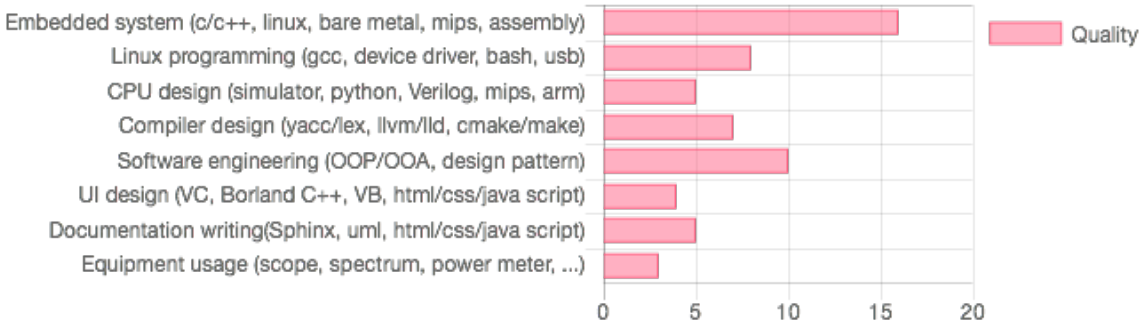
I am a programmer who used different computer languages, tools and computer related knowledge to develop different software and hardware products, such as Set Top Box, CPU, compiler, simulator, ..., in my career. I always have full passion in programming, knowing tools and software engineering method well, loving programming and debugging, and feel happy about writing good document to deliver more readable and maintainable software.

CV (DETAIL RESUME)

QUALIFICATION

Over 20 years experience in c/c++ embedded system programming, 5 years compiler toolchain related experience and research in parallel processing for master degree.

SKILLS



MY OPEN SOURCE PROJECT

I am proud of my work is accepted by LLVM documentation, appears at http://llvm.org/docs/tutorial/#external-tutorials

Tutorial: Create an LLVM Backend compiler http://jonathan2251.github.io/lbd/index.html

Tutorial: Create an LLVM Backend Toolchain http://jonathan2251.github.io/lbt/index.html

EDUCATION

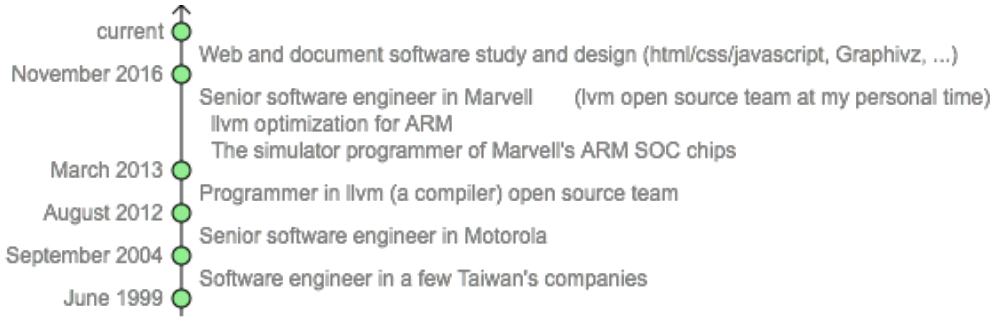
1997-1999 Master, June 1999, National Taiwan Normal University (國立台灣師範大學), Taipei, Major: Information Science.

1991-1994 B.S., June 1994, National Taiwan Technology University of Science and Technology (國立台灣科技大學), Taipei, Major: Industry Engineer.

LICENSE

Taiwan National Computer Engineer license, 1995 高考資訊技師及格.

EXPERIENCE



THESIS OF MASTER DEGREE

The Researches of Column Sort and Related Problems

PROPOSAL OF PHD STUDY

The Researches of Sorting Network and Related Algorithm

OTHER WORK

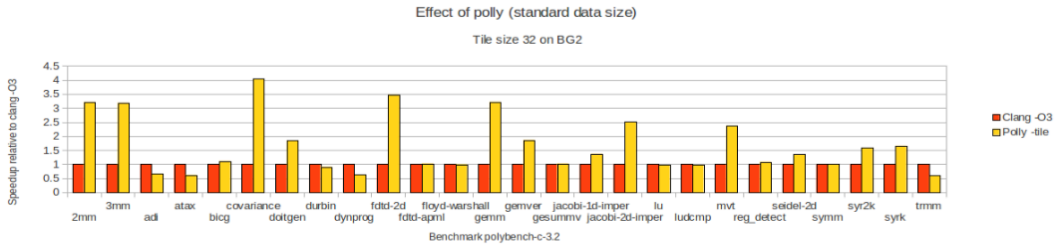
Take course "Image processing" and program: Jpeg decoder

Web and javascript: As my resume and my personal web site

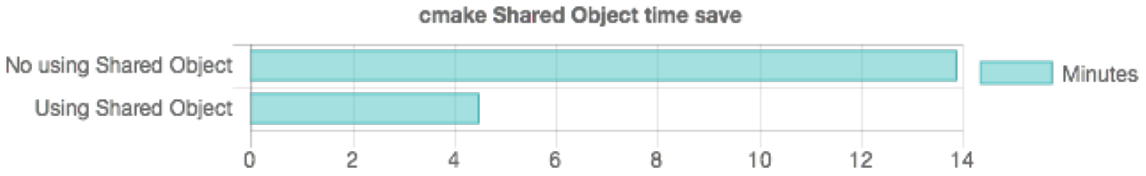
ACHIEVEMENT

Marvell

- Implement semi-auto software system of running benchmark and generating report for gcc toolchain.
  - Implemented bash script and run it automatically to generate Excel's format comparison report when the source code of Marvell's gcc toolchain has updated. The purpose is to make sure the new gcc has no side effect (the gcc programmer may update code which brings bad effect on some benchmark).
  - Implemented bash script which can compare any two version of gcc toolchain and generate report.
- Introduce polly for Marvell llvm toolchain optimization. Polly is a software for loop optimization.
  - The advantage is that it might be 500% speed up in some numerical application program, such as matrix multiplication, or other kind of matrix operation. Sadly, Marvell's CPUs are not used in numerical applications. But maybe it will have for 64 bits ARM in future.



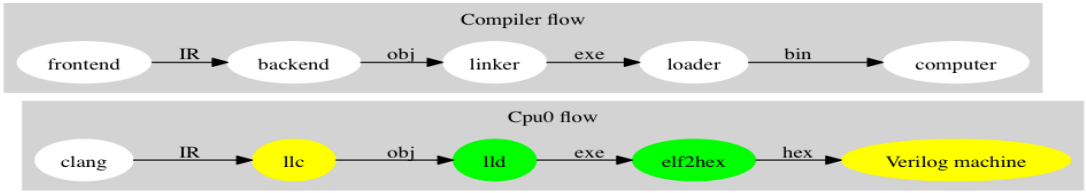
- Implement co-simulator(our name:Csim) for a few Marvell's ARM based cpu.
- Apply compiler code generation idea to propose a solution and Csim team adopted it. reference
- Complete cmake and adjust python programs to build Csim with cmake. It saves effort in maintaining both make and windows project file. In addition, cmake also support iMac user. reference
  - Present the better option for cmake writting.
    - The version of cmake in Csim server is 2.6.x, however, the cmake 2.8.8 provides shared object to save the unnecessary compilation.



  - Report performance comparing between cmake and ninja for other manager's concern. The result is "no difference"

LLVM open source project

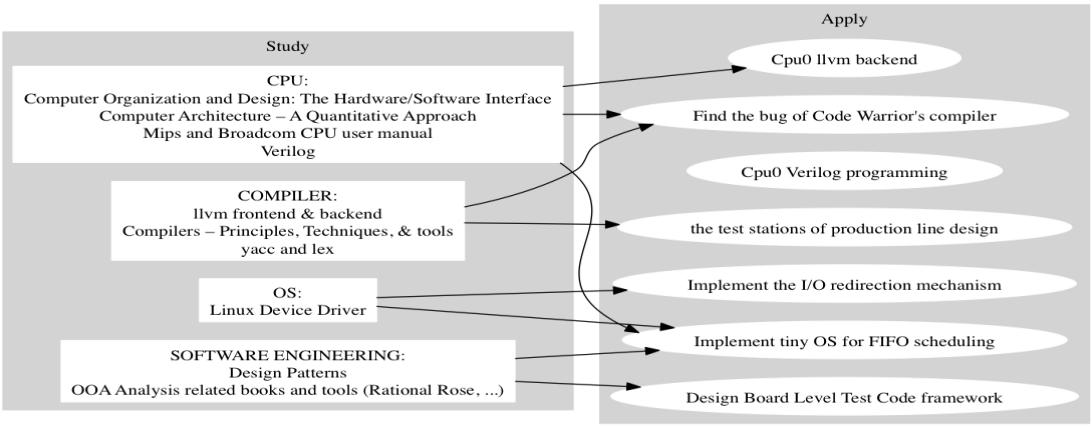
- Use knowledge from compiler books, llvm document/source code, CPU architecture books and Verilog, implement the llvm backend and Verilog CPU, Cpu0, to create a backend and write book as tutorial.
- Study the llvm linker open source code, lld, and llvm run time library open source code, compiler-rt, and implement elf linker and software floating point library for Cpu0.
- The upper half of the following figure is the work flow and software package of a computer program be generated and executed. IR stands for Intermediate Representation. The lower half is the llvm's work flow. The yellow and green parts are implemented in my books (listed as my open source project above), Tutorial: Create an LLVM Backend compiler and Tutorial: Create an LLVM Backend toolchain respectively.



Mortorola

- Designe QIP7kP1 and P2 board level production line testing code. Over 500 millions units of QIP7k model of Set Top Box are tested with this code on production line and shipping to market. Other major products DCX33, DCX34 and DCX35 (over 1000 millions of box) are ported from QIP7kP1 & QIP7kP2.
- Combine mips cpu and software engineer knowledge, lead and cowork with other four members to create a frame work for board level testing code as manager request.

CPU, COMPILER, OS, AND SOFTWARE ENGINEERING RELATED LEANING & APPLY



CPU architecture study

- Study books “Computer Organization and Design: The Hardware/Software Interface”, “Computer Architecture – A Quantitative Approach”, Mips and Broadcom CPU user manual and Verilog.
- Apply in work:
  - Find the bug of Code Warrior's compiler in supporting Mips cpu. The bug come from code generation for #pack(1) statement.
    - Email Code Warrior the bug along with the example code I write for identifing this bug, and they admit that is a bug from their compiler for mips backend support.
  - Implement the "tiny OS" mechanism with mips assembly language to solve the halting problem in some program.
    - After this implementation, the code has run more stable and fast. It is a serious problem since when one test item "halting", the other test items scheduled after that one won’t be run. In this situation, operator has no idea of the test result since test result is displayed after all test items finish execution. I implement a "time out" mechanism to jump out from the test item (by restore the stack and cpu registers status). My USA's coworker call it "tiny OS". It's c++ plus few hundred lines of mips assembly with "observe pattern" of "design pattern".
  - Apply knowledge from Verilog and book “Computer Architecture” to change the Verilog code of Cpu0 to become a cpu design for my llvm backend compiler support.

Compiler study:

- Study llvm frontend & backend design. Study book “Compilers – Principles, Techniques, & tools 2nd Aho,...”, yacc and lex tool.
- Apply in work:
  - Implement the Cpu0 llvm compiler backend from scratch for tutorial purpose, and writing books as in my publish books.
  - Apply the compiler syntax parsing tool, yacc & lex, in test station design.

OS study:

- Study book “Linux Device Driver (2nd & 3rd edition)”
- Apply in work:
  - Implement the "tiny OS" mechanism as mentioned in above section "CPU architecture study".
  - Implement the I/O redirection mechanism to make the new version of code porting task easier in Motorola.

Software Engineering study:

- Study book “Design Patterns” and "OOA Analysis related books and tools (Rational Rose, ...)"
- Apply in work:
  - During working the Board Level Test Code framework project in Motorola as mentioned above:
    - Utilize five patterns in "design pattern" and Rational Ross tool to create framework, document and generate code in Motorola.

References

[My former manager's recommendation letter: https://jonathan2251.github.io/ws/en/RL_Marvell.pdf](https://jonathan2251.github.io/ws/en/RL_Marvell.pdf)