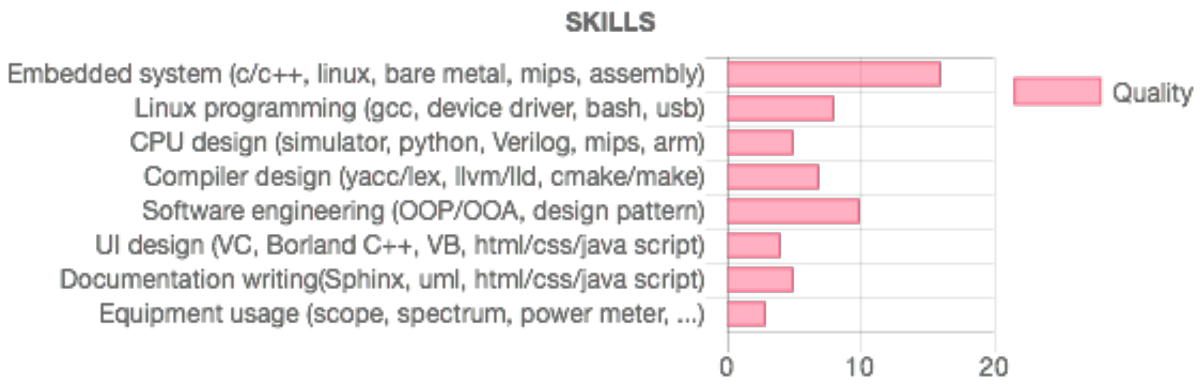
Chung-Shu Chen 陳鍾樞 (03)6681193, 0970577923 gamma_chen@yahoo.com.tw

I am an experienced programmer who used many different computer languages, tools and computer related knowledge to develop products of Set Top Box, CPU, compiler, simulator, ..., in my career. In addition, I know tools and software engineering method very well and write good document for my program to deliver readable and maintainable software.

RESUME

QUALIFICATION

Over 20 years experience in c/c++ embedded system programming, well knowledge in software development and debugging.



MY OPEN SOURCE PROJECT

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

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

EDUCATION

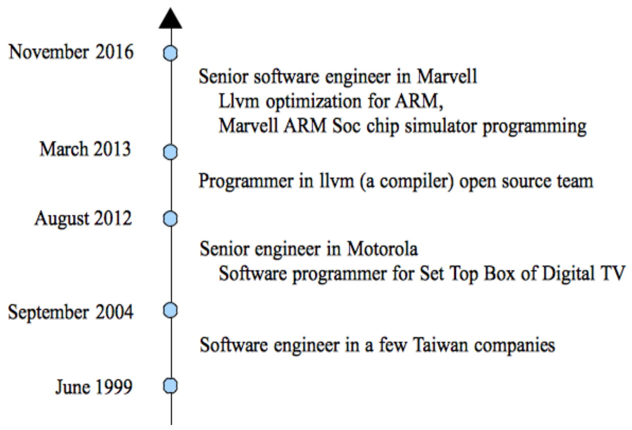
1997-1999 Master, June 1999, National Taiwan Normal University, Taipei, Major: Computer Science.

1991-1994 B.S., June 1994, National Taiwan Technology University of Science and Technology, Taipei, Major: Industry Engineer.

LICENSE

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

EXPERIENCE


November 2016
Senior software engineer in Marvell
Llvm optimization for ARM,
Marvell ARM Soc chip simulator programming
March 2013
Programmer in llvm (a compiler) open source team
August 2012
Senior engineer in Motorola
Software programmer for Set Top Box of Digital TV
September 2004
Software engineer in a few Taiwan companies
June 1999

THESIS OF MASTER DEGREE

[The Researches of Column Sort and Related Problems]()

PROPOSAL OF PHD STUDY

[The Researches of Sorting Network and Related Algorithm]

ACHIEVEMENT

Marvell

Introduced polly for Marvell llvm toolchain optimization. Polly is a software for loop optimization.

> The advantage was that it might be 500% speed up in some numerical application program, such as matrix multiplication, or other kind of matrix operation. Though Marvell's CPUs were not used in numerical applications.

I applied compiler code generation concept to propose this solution and Csim team adopted it.

I completed cmake and adjusted python programs to build Csim instead of make.

Report the better solution for cmake writting and performance compare with ninja.

Mortorola

Designed QIP7kP1 and P2 board level testing code. Over 500 millions units of QIP7k model of Set Top Box was tested with this code on production line, as well as shipping to market. Other major products DCX33, DCX34 and DCX35 (over 1000 millions of box) were porting from QIP7kP1 & QIP7kP2.

Combined mips cpu and software engineer knowledge to create a frame work for board level testing code as manager request.

> I used Rational Ross UML tool to design the frame work. In this task. I applied object oriented analysis & program (OOA, OOP, design pattern) knowledge and UML tool skill.

Implemented the I/O redirection mechanism to make the new version of code porting task easier. One of my colleague adopted this mechanism by his will.

Implemented Board Level Test with design pattern

CPU, COMPILER, AND OS RELATED EXPERIENCES

CPU architecture study

Studied books “Computer Organization and Design: The Hardware/Software Interface” and “Computer Architecture – A Quantitative Approach” and Mips CPU user manual. Verilog language/tool study.

Apply in work:

> Finding the compiler bug of Code Warrior compiler for Mips cpu. The bug came from #pack(1) statement code generation.
>
> I emailed the bug along with example code to Code Warrior and it admited that this was a bug of their compiler for mips backend.

Implemented the mini process mechanism with mips assembly language to solve the program hang problem. After this implementation, the code was running more stable and fast. It was a serious problem since when one test item hang on, the other test items scheduled after that wouldn’t be run. In this situation, operator had no idea of the test result since test result was displayed after all test items executed. I implemented a mechanism (hundred lines of mips assembly) to jump out from the test item (by restore the stack and cpu registers status).

Applied Verilog and book “Computer Architecture” knowledge to change Cpu0 verilog code to fit my llvm backend compiler design.

Compiler study:

Studied llvm front & backend design. Studied book “Compilers – Principles, Techniques, & tools 2nd Aho,...” and yacc and lex tool.

Apply in work:

> I implemented the Cpu0 llvm compiler backend started from scratch for tutorial purpose, and writing books as in my publish books. A lot of readers have been asking me questions and I answering them out of working time.

Applied the compiler syntax parsing tool, yacc & lex, in test station design (specify test items for each station in script language I defined).

OS study:

Study book “Linux Device Driver (2nd & 3rd edition)”

Apply in work:

> Implemented the mini process mechanism as mentioned in above section "CPU architecture study".