SystemC 기반 RISCV 모델링 위한 Visual Studio 개발 환경 구축 및 시뮬레이션

김건명*, 이재빈*, 임승호*, 조상영*
*한국외국어대학교 컴퓨터.전자시스템공학부
kgm2543@naver.com, gaebin1212@gmail.com, slim@hufs.ac.kr, sycho@hufs.ac.kr

Building and Simulation of Visual Studio Development Environment for SystemC-based RISCV Modeling

Geon-Myoung Kim*, Jae-Bin Lee*, Seung-Ho Lim*, Sang-Young Cho*
*Div. of Computer and Electronic Systems Engineering, Hankuk University of Foreign Studies

요 약

현재 RISCV-VP 시뮬레이션 프로그램은 Linux 환경으로만 존재하며, Windows 환경에서 동작하는 프로그램은 존재하고 있지 않다. 그래서 Linux 환경으로 개발된 RISC-VP(Virtual Platform)를 Windows 환경에서도 사용할 수 있게 Visual Studio 를 사용하여 구축하였다. 또한, Windows VS 환경의 RISCV-VP 프로그램을 통해 머신 러닝 시뮬레이션이 가능하도록 개발 환경을 구축하였다. 이 논문은 위에 설명한 각각의 개발환경 구축에 필요한 구성 방법을 제공한다.

1. 서론

RISC-V 칩은 상용 ARM 칩과 비교해서 비슷한 성능으로 칩 면적은 30%~50% 축소되고 소비전력은 60%나 감소하는 등 상당히 높은 효율과 경제성을 보여 미래에 상용화된다면 ARM 의 강력한 경쟁자가 될 수 있다는 기대를 받고 있다[1]. 현재 RISC-V 기반 개발을 위해 RISC-V 를 시뮬레이션 할 수 있는 프로그램은 다수 존재한다. 다양한 프로그램이 있지만 모두 Linux 환경으로만 존재한다.

본 논문에서는 Linux 가 아닌 Windows 에서 RISC-V 기반 개발을 하고자 하는 Windows 사용자를 위해 Linux 플랫폼에서 시뮬레이션 할 수 있는 RISCV-VP(RISCV-Virtual Platform)를 Windows 플랫폼인 Visual Studio 에서 RISC-V 를 개발 가능한 환경을 구축하였다.

또한, 이 개발 환경에서 머신 러닝 모델링을 위해 CNN 모듈을 추가하였고, 총 구현된 개발 환경에서 시뮬레이션 및 테스팅을 진행하였다.

2. Windows VS based RISC-VP 요구사항

Linux 환경으로 구성된 RISCV-VP(Virtual Platform)[2][3]를 Windows VS 환경에서 사용하기 위해서는 개발이 진행될 IDE 인 Windows Visual Studio 그리고 기존의 RISCV-VP 를 VS 로의 porting 작업이 필요하다. porting 작업에는 RISC-VP 를 동작 시키기에 필요한 필수 라이브러리 구성 및 VS 프로젝트 설정 그리고 Linux 기준으로 작성된 RISC-VP API 를 Windows 환경 기준으로의 수정이 필요하다.

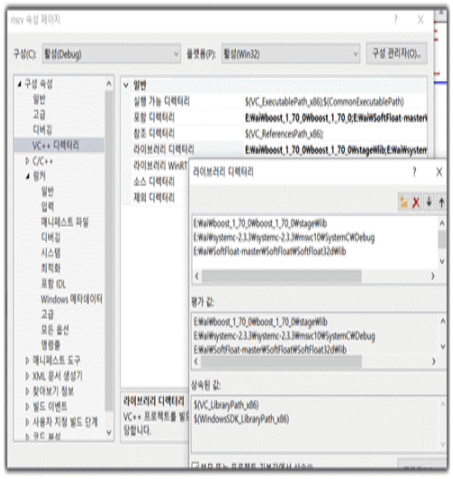
사용된 Windows VS 버전은 Windows Visual Studio 2019로 (그림1)과 같다. (그림 1)과 같이 확인하려면

Microsoft Visual Studio 도움말 탭의 정보 항목에서 확인할 수 있다.

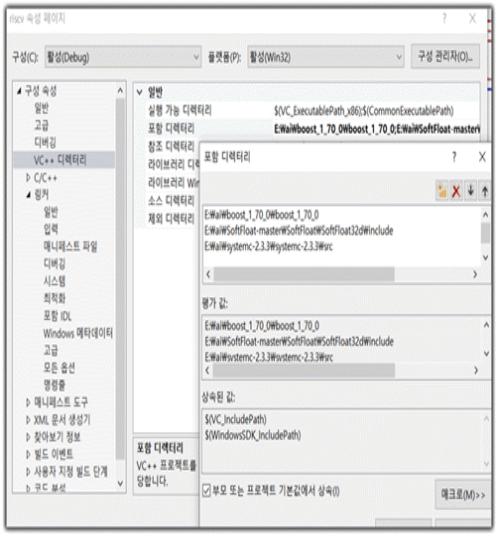
(그림 1) Windows Visual Studio 2019

RISC-VP 에서 필요로 하는 라이브러리는 3 가지로 1) SystemC[4] 2) Boost[5] 3) SoftFloat[6] 이다. 각 라이브러리를 설치하고 빌드하여 사용이 가능한 상태가 되면 우리가 RISC-VP 환경을 구현할 VS 프로젝트에 포함해주어야 한다. (그림 2), (그림 3)은 빌드가 완료되어 라이브러리 및 라이브러리 코드가 저장된 경로를 프로젝트에 추가한 것이다. 추가로 프로젝트 속성을 사용하는 라이브러리와 호환할 수 있도록 변경해주어야 한다.

프로젝트 속성이 완료되었으면 기존의 RISCV-VP 소스 코드들을 Windows VS 프로젝트에 추가하고 추가된 RISC-VP API 를 Windows 환경에서 사용할 수 있게 Linux 시스템콜을 Windows 에서 동작할 수 있도록 수정하는 작업이 필요하다.



(그림 2) SystemC, Boost, Softfloat 라이브러리추가



(그림 3) SystemC, Boost, Softfloat Include 추가

3. Windows에서 동작하는 기본 RISCV-VP 시뮬레이터에 CNN모듈 추가

Window에서 동작할 수 있도록 수정된 RISCV-VP 시뮬레이터에서 머신 러닝 개발 및 시뮬레이션을 수행하기 위해서는 SystemC 기반의 RISCV-VP에 CNN[7] 모듈을 구현하여 추가하는 과정이 필요하다. 추가한 CNN모듈은 RISCV-VP에서 딥러닝 연산이 가능하도록 SystemC로 테스트용으로 자체적으로 설계 및 구현한 하드웨어 모듈로써 기존 RISCV-VP에 포함되어 있지 않은 추가될 모듈이다.

모듈 추가는 크게 모듈 생성, 포트 추가, 바인딩 작업 순으로 진행되며 동작을 위한 각 모듈을 클래스 형태로 선언하고 정의한 뒤 (그림 4)와 같이 main 함수 내부에서 추가하는 과정으로 이루어진다.

```
CNN_MODULE cnn("CNN_MODULE", 6, mem.data, &count);
bus.ports[7] = new PortMapping(opt.cnn_loader_start_addr, opt.cnn_loader_end_addr);
bus.isocks[7].bind(cnn.router->tsock);
```

(그림 4) RISCV-VP 에 모듈을 추가하는 과정

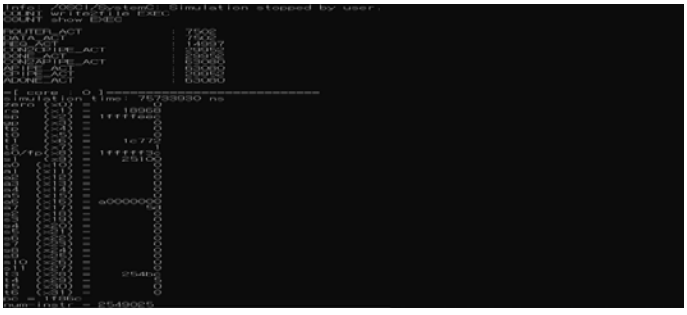
4. ELF 파일 동작 테스트 및 머신러닝 동작 테스트

구현된 RISCV-VP는 RISC-V 칩 HW를 시뮬레이션 할 수 있는 환경으로 SystemC 기반으로 구현이

되어있다. 시뮬레이션이 시작하면 선언된 모듈들은 모두 생성되며 SW 프로그램이 HW상에 올라가면 각각의 모듈들은 필요한 동작을 수행하는 구조이다. 여기에서 SW 프로그램에 해당하는 것이 ELF 파일이다. ELF 파일은 Linux 에서 RISCV-VP에서 실행하고 싶은 SW 프로그램을 컴파일하여 생성하고, Linux 상에서 생성된 파일은 Linux RISCV-VP 그리고 Windows VS RISCV-VP 환경에서 둘 다 사용할 수 있다.

(그림 5)는 빌드된 Windows VS 기반의 RISCV-VP로 예제를 실행한 결과이며, (그림 6)는 RISC-VP에 SystemC기반의 CNN 모듈을 추가해 주고 빌드 및 러닝을 진행한 결과이다.

(그림 5) 빌드된 Windows VS 기반의 RISCV-VP 로 예제를 실행한 결과



(그림 6) RISC-VP 에 CNN 모듈을 추가하여 테스트한 결과

5. 결론

Linux 에서만 동작이 가능했던 RISCV-VP 시뮬레이터를 Windows VS 에서도 동작이 가능하도록 Porting 을 완료하여 기본적인 RISCV-VP 명령어 구성으로 이루어진 ELF 파일이 동작하는 것을 확인했고, CNN 연산 즉 머신 러닝 모듈을 추가하여 적절한 Input 데이터를 통한 연산이 가능하다는 것도 확인하였다. 다양한 플랫폼에 종속 받지 않고 개발 및 동작을 할 수 있는 시뮬레이터를 구성해 봄으로써 좀 더 유연하고 다양한 개발이 가능해진다.

Acknowledgments

본 연구는 과학기술정보통신부 및 정보통신기획평가원의 SW 중심대학지원사업의 연구결과로 수행되었음(2019-0-01816)

참고문헌

[1] “RISC-V compliance task group,” https://github.com/riscv/riscv-compliance.

[2] Github, RISC-V Virtual Prototype, https://github.com/agra-uni-bremen/riscv-vp

[3] V. Herdt, D. Groβe, H. M. Le and R. Drechsler, "Extensible and Configurable RISC-V Based Virtual Prototype," 2018 Forum on Specification & Design Languages (FDL), Garching, pp. 5-16, 2018.

[4] David C.Black, Jack Donovan SystemC : From the Ground Up. 2004

[5] “boost c++ libraries”, https://www.boost.org/

[6] “Berkeley SoftFloat”, http://www.jhauser.us/arithmetic/SoftFloat.html

[7] Lin Bai, Yiming Zhao, Xinming Huang (2018). A CNN Accelerator on FPGA Using Depthwise Separable Convolution. IEEE Transactions on Circuits and Systems II: Express Briefs,65,1415 – 1419