FPGA 기반 고성능 자율주행 자동차 시스템 개발

조민진[1], 김민지[1], 김유진[1], 정다연[1], 손명주[1], 백기영[2]
[1]이화여자대학교 전자전기공학전공 학부생
[2]KT AI Robot/교통플랫폼개발팀 SW개발본부 DX플랫폼개발담당
minjin9384@ewhain.net, alswl0109@ewhain.net, realyj2000@ewhain.net,
dyeon1324@ewhain.net, black8744@naver.com, giyoung.baek@kt.com

Development of FPGA-based high performance autonomous vehicle system

Minjin-Cho[1], Minji-Kim[1],Yu-Jin Kim[1], Da-Yeon Chung[1], Myeong-ju Son[1],
Gi-Young Baek[2]
[1]Dept. of Electrical Electronic Engineering, Ewha Womans University
[2]Dept. of AI Robot/Traffic Platform Development, KT

요 약

최근 자율주행 기술에 대한 관심과 함께 고가의 하드웨어와 소프트웨어의 복잡성이 자율주행 자동차의 상용화를 가로막고 있다. 이에 본 연구에서는 해당 문제점들을 보완하기 위해 FPGA의 유연성과 고성능이라는 이점으로 자율주행 자동차 시스템을 최적화하였다.

1. 서론

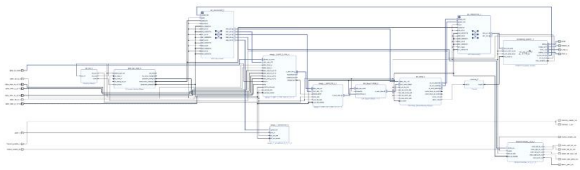
21세기 들어서 자율주행 기술은 급속도로 발전하고 있다. 많은 기업들이 자율주행 자동차를 개발해 시험 운행 중이지만, 하드웨어 경제성과 소프트웨어 복잡성 문제로 상용화가 지연되고 있다. 이에 정교하고 정확한 자율주행 기술이 필요한 상황이다.

본 논문에서는 FPGA를 활용하여 자율주행 기능을 고성능으로 구현한다. PCAM과 YOLO 기술을 사용해 도로 상황을 인지하고 판단하며, FPGA를 통해 모터를 제어하고 하드웨어 가속기를 설계했다. 최종 성능 확인을 위해 제작 도로에서 자율주행 운행을 진행한다.

2. 하드웨어적 구현

i) 모터 제어를 위한 하드웨어 IP 설계

PWM Generator, Motion Controller, Sonar Driver 등 모터의 속도와 방향을 제어하는 IP들을 구현하였다. AXI 인터페이스를 통해 IP 간의 효율적인 데이터 전송을 가능하게 한다. 차량제어는 OpenCV 모듈을 통해 각 프레임에서 이미지를 처리하며 이미지 처리 결과에 따라 플래그와 변수를 설정한다. 차량 제어 모듈은 이러한 플래그와 변수에 따라 모터의 속도와 방향을 제어하고 GPIO와 같은 주변장치를 통하여 외부와의 상호작용을 관리한다. 이를 바탕으로 작성한 모터 제어 설계도와 핵심 IP에 대한 설명은 그림1과 표1과 같다.



[그림1:모터 제어 설계도]

AXI (Advanced eXtensible Interface) Interconnect	AXI 프로토콜을 사용하여 마스터 장치(예: 프로세서)와 슬레이브 장치(예: 메모리, 주변 장치) 간의 통신을 관리하며 여러 마스터와 슬레이브 장치 간의 병렬 데이터 전송을 가능하게 하여 시스템 성능을 극대화함
AXI Timer	타이머/카운터 기능을 제공, 시스템 내에서 특정 시간 간격을 측정하거나, 카운터로 동작하여 이벤트를 모니터링하는 데 사용됨
AXI DMA (Direct Memory Access)	프로세서의 개입 없이 메모리 간의 데이터 전송을 가능하게 하며 고속 데이터 전송을 필요로 하는 응용 프로그램에서 성능을 크게 향상시킴
PWM(Pulse Width Modulation) Generator	PWM 신호를 생성하여 모터의 속도와 방향을 제어하며 PWM 신호의 듀티 사이클을 조절하여 모터의 동작을 정밀하게 조정함
Clocking Wizard	다양한 주파수의 클럭을 생성하고 분배하며 시스템 내의 다양한 부분에서 필요로 하는 클럭 신호를 생성해 동기화된 동작을 보장함
Reset Generator	시스템의 리셋 신호를 관리하여 전원 인가 시 또는 특정 조건이 발생했을 때 시스템을 안정적으로 초기화하는 데 사용됨
BRAM (Block RAM)	FPGA 내부에서 사용할 수 있는 고속 메모리 블록을 제공하며 주로 데이터 버퍼링, 임시 저장소, 또는 상태 저장에 사용됨
GPIO (General-Purpose Input/Output)	FPGA의 핀을 제어할 수 있는 일반적인 I/O 포트를 제공해 외부 장치와의 신호 송수신을 담당하며, 입력 신호를 수집하거나 제어 신호를 출력시킴

[표1:모터 제어 설계도 하드웨어 IP 설명]

ii) 하드웨어 가속기 설계

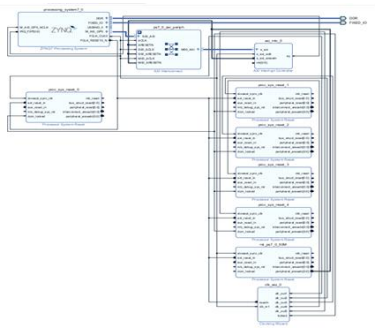
PetaLinux는 Linux 내의 이미지를 사용자 지정, 빌드 및 배포할 수 있는 tool이다. 해당 tool은 설계 생산성을 가속화하도록 맞춤화되어있으며, Linux 시스템 개발을 용이하게 한다.[1] 본 연구에서는 PetaLinux를 이용하여 하드웨어 가속기를 하고자

한다. 순서는 다음 표3과 같다.

1	하드웨어 플랫폼을 생성하기 위해 Vivado에서 HDF 파일을 출력한다. HDF 파일은 하드웨어 아키텍처를 설명하며, 보드의 전원이 켜졌을 때 이미지를 올바르게 구성하는 데 필수적이다.
2	HDF를 사용해 보드에 대한 타겟 프로젝트를 빌드하기 위inux 프로젝트를 생성한다풀력된 HDF를 사용해 보드에 대한 타겟 프로젝트를 빌드하기 위해 PetaLinux 프로젝트를 생성한다. PetaLinux는 Linux OS 환경에서 실행되므로 가상 머신에서 Ubuntu를 사용해 환경을 구성한다.
3	SD 카드에 FAT32 형식의 파티션을 나눈 후 포맷하고, 부팅 이미지를 넣는다. SD 카드를 보드에 장착하고 전원을 인가하면 PetaLinux가 부팅된다.
4	Zybo Z7-20 보드로 Vitis 내에 애플리케이션 프로젝트를 생성하고, 이 과정에서 Linux 도메인에 petalinux-build로 생성된 roofts.cpio, linux.bif, qemu_args.txt 등을 활용한다. 새로운 벡터 덧셈 파일을 생성해 Vitis를 개발하여, SW emulation, HW emulation, HW 부문에서 configuration을 실행한다.
5	SD 카드의 FAT32 파티션에 binary_container_1.xclbin, BOOT.BIN, boot.scr, image.ub, vector_addition 실행 파일들을 옮긴 후, 이를 부팅하여 확인하기 위해 GTKTerm에서 "./vector_addition binary_container_1.xclbin" 을 실행하면, 부팅이 성공할 경우 "TEST PASSED" 메시지를 확인할 수 있다.

[표3:PetaLinux를 이용한 하드웨어 가속기 설계 순서]

Vivado로 작성한 설계도 모습과 핵심 IP 블럭에 대한 설명은 다음 그림3과 표4과 같다.



[그림3:Vivado 가속기 설계도]

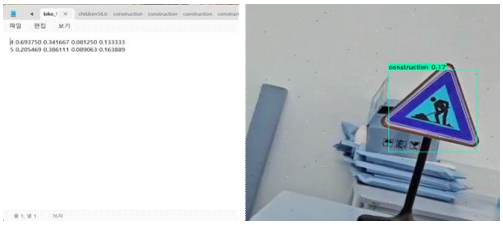
AXI4-Stream 인터페이스	입력 데이터와 출력 데이터를 고속으로 전송할 수 있는 표준 인터페이스를 제공. 이를 통해 데이터 병목 현상 최소화 가능
Input FIFO	입력 데이터를 일시적으로 저장하여 Input Processor 블록 안정적으로 공급. 입력 데이터 처리의 연속성 보장
Input Processor	병렬 연산 능력을 활용하여 입력 데이터를 고속으로 처리. 다양한 연산 유닛들이 포함되어 있어, 여러 작업 동시 수행 가능 (pipelining)
Output FIFO	처리된 출력 데이터를 일시적으로 저장하여 AXI4-Stream 인터페이스로 안정적으로 전송. 출력 데이터의 원활한 전송 보장.

[표4:Vivado 설계도 내 핵심 IP 설명]

3. 소프트웨어적 구현

i) YOLOv3 tiny 모델을 이용한 학습

YOLOv3(You Only Look Once version3)는 이미지와 비디오에서 객체를 빠르고 정확하게 탐지하는 알고리즘으로, 그리드 셀을 이용해 bounding box를 예측한다.[2] YOLOv3 tiny는 경량화된 버전으로, 임베디드 시스템과 모바일 장치에서 실시간 처리를 위해 설계되었다. Vitis AI와 호환되는 YOLOv3 tiny를 사용하여 총 9개의 클래스를 설정하고, 직접 촬영한 영상으로 데이터셋을 확보한 후 data augmentation과 labelling을 진행하였다. 최종 weight 파일을 기반으로 tensorflow를 이용해 파일 변환 후 Vitis AI에서 사용하였다. 실행 모습은 그림 4와 같다.



[그림4:YOLOv3 실행 모습]

ii) canny edge detection을 이용한 차선 인식

canny edge detection은 컴퓨터 비전에서 이미지의 경계(edge)를 검출하는 데 사용되는 알고리즘이다. 해당 알고리즘은 노이즈에 강하고 정확한 경계선을 검출할 수 있도록 설계되어있다. 차선 인식을 보다 정확하게 하기 위해, canny edge method로 차선의 edge를 추출하고, 이미지에 mask를 씌워서 차선에서 필요한 포인트들만 반환한다.

4. 결론

본 논문에서는 FPGA 기반 고성능 자율주행 시스템에 대해 연구하였다. 이는 PCAM에서 수집한 영상데이터를 YOLOv3를 이용한 객체 인식 알고리즘으로 실시간으로 처리하여 도로 상황을 인식하고, usecase에 따라 모터를 제어했다. 이를 통해 자율주행 자동차의 실시간 데이터 처리와 높은 정확도를 확인했다. 또한 FPGA를 통해 객체 알고리즘을 가속화함으로써 시스템의 효율성을 향상시켰다.

성능 확인을 위해 본 연구에서는 자율주행 시나리오를 작성하였다. 예를 들어 사거리 교차로 직진 주행, 사거리 좌회전, 정적장애물 회피 주행 등 총 8가지의 usecase를 설정하여 자율주행 RC카가 제대로 구동되는 것을 확인할 수 있었다. 결과적으로 약 6가지의 시나리오가 성공하였음을 확인하였다.

※ 본 논문은 과학기술정보통신부 대학디지털교육역량강화 사업의 지원을 통해 수행한 ICT멘토링 프로젝트 결과물입니다.

참고문헌

[1] M. Hosseinabady and J. L. Nunez-Yanez, "A systematic approach to design and optimise streaming applications on FPGA using high-level synthesis," 2017 27th International Conference on Field Programmable Logic and Applications (FPL), Belgium, 2017, pp. 1-4

[2] J. Redmon, et al, "You Only Look Once: Unified, Real-Time Object Detection," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), USA, 2016, pp. 779-788