한국정보전자통신기술학회논문지(jkiiect)'21-10, Vol.14 No.5
논문 21-14-05-385 http://dx.doi.org/10.17661/jkiiect.2021.14.5.385

AURIX TC 275에서 멀티코어를 이용한 Electronic Stability Control의 수행시간 최적화

장홍순*, 조영환*, 정구민**

Processing Time Optimization of an Electronic Stability Control system design Using Multi-Cores for AURIX TC 275

Hong-Soon Jang*, Young-Hwan Cho*, Gu-Min Jeong**

요 약 본 논문에서는 차량 멀티코어 프로세서를 통한 ESC(Electronic Stability Control) 시스템을 위한 멀티코어 기반 제어기를 제시한다. 차량용 멀티코어 프로세서와 ESC 시스템의 아키텍처를 고려할 때 ESC 소프트웨어의 전체 수행 시간은 멀티코어에 최적화되어 있다. 일반적으로 차량용 멀티코어 시스템에서는 코어 간 동기화, 멀티코어에 대한 테스크 할당, 코어 종속 변수에 대한 메모리 할당을 고려해야 한다. 본 논문에 사용된 ESC 시스템은 초기화, SlipRatio 계산, YawRate 계산, ABS, 통신으로 구성된다. 제안된 설계 방법을 기반으로 싱글코어 프로세서는 멀티코어 프로세서로 확장된다. ESC 시스템은 기능 모듈 할당, 세마포어, 인터럽트, 코어 별 변수 할당과 같은 멀티코어 최적화 방법을 사용하여 멀티코어 제어기로 재설계된다. 실험 결과로 멀티코어 프로세서의 수행 시간이 싱글코어 프로세서에 비해 59.7% 단축되었다.

Abstract This study proposes a multi-core-based controller design for an ESC(Electronic Stability Control) system in an automotive multi-core processor. Considering the architectures of an automotive multi-core processor and an ESC system, the overall execution time has been optimized for multi-core platforms. The function module assignment, synchronization between cores, and memory assignment for core-dependent variables in automotive multi-core systems are evaluated. The ESC controller comprising five function modules is used herein. Based on the proposed design, the single-core controller is extended to multi-core controllers. Using multi-core optimization methods, such as function module assignment, semaphore, interrupt awakening, and variable assignment over cores, the ESC system is redesigned to a multi-core controller. Experimental results reveal that the execution time for the multi-core processor is reduced by 59.7% compared with that for the single-core processor.

Key Words : Multi-core processor, ESC system, Synchronization, Interrupt, Memory allocation

1. 서론

최근 자동차 제어 시스템이 더욱 복잡해지고 시스템 요구 사항이 급격하게 증가하고 있다. 멀티코어 프로세서는 복잡한 소프트웨어 아키텍처를 단순화하고 싱글코어 아키텍처의 한계를 극복할 수 있기 때문에 다양한 애플리케이션에 활용되고 있다 [3]. 멀티코어를 활용하면 수행 시간을 상당히 절감할 수 있지만 제대로 설계되지 않으면 오히려 성능이 저하될 수 있다. 이러한 단점을 완화하기 위해 자동차용 멀티코어 설계 [1]-[3]와 같은 다양한 멀티코어 기술이 제안되었다.

대부분의 AUTOSAR 호환 마이크로컨트롤러(MCU)가 3개의 코어로 구성되기 때문에 자동차 제

This Paper was supported by research Fund of Hyundai Transys in 2021.
*Department of Electronic Engineering, Kookmin University
**Corresponding Author : Department of Electronic Engineering, Kookmin University (gm1004@kookmin.ac.kr)
Received October 15, 2021 Revised October 18, 2021 Accepted October 24, 2021

어 시스템의 멀티코어 프로세서는 3개의 코어로 구성된다 [9]. 자동차용 멀티 코어 설계는 타이밍 및 기능 안전 문제, Lockstep core, 수행시간 그리고 성능 문제를 극복해야 한다 [7]. 전체적인 시스템 성능은 싱글코어 MCU의 레거시 코드로부터 기능 또는 아키텍처 할당 방법을 사용하여 개선될 수 있다. 리소스를 공유할 때 시간 동기화가 보장되지 않으면 시스템 성능이 저하될 수 있다. 동기화 문제를 극복하기 위해 세마포어, 스핀락 및 이벤트와 같은 기법 [10]뿐 아니라 분할 스케줄링 및 멀티코어 게이트웨이와 같은 다양한 스케줄링 기법이 사용된다 [4]-[7]. 게다가, 변수들은 메모리 접근 시간을 줄이기 위해 코어 독립 메모리에 할당될 수 있다. Lockstep core에서 동일한 코드를 확인하기 위해 Lockstep core를 사용하여 기능 안전 이슈를 극복할 수 있다 [8]. 싱글코어 아키텍처들은 다양한 멀티코어 기술을 사용하여 자동차 애플리케이션에서 멀티코어 아키텍처로 확장됐다 [2].

[13]에서는 싱글코어에서의 ESC 애플리케이션을 싱글코어 프로세서에서 순차처리를 통한 수행 시간과 멀티코어 프로세서에서 세마포어 및 인터럽트를 활용한 수행 시간을 비교하였다. 본 논문에서는 [13]에서 더 진보된 차량용 멀티코어 프로세서의 ESC 시스템의 수행시간 최적화 방법을 제안한다. 기능 모듈을 두 개의 코어로 분배하고 각 기능 모듈에 사용되는 변수들 또한 프로세서 공용 메모리에 저장하는 대신 각각의 코어 종속 메모리에 배치하여 변수 액세스 시간을 줄여서 총 수행 시간을 크게 감소시켰다. 멀티코어 소프트웨어는 기능 모듈 할당, 세마포어 및 인터럽트에 의한 다른 코어 실행, 코어 종속 메모리에 대한 변수 할당을 통한 자동차 MCU 아키텍처와 ESC 시스템 구조의 특성을 고려하여 설계된다. 성능 향상은 Infineon AURIX 프로세서를 통한 시뮬레이션 결과를 통해 입증된다.

2. 관련 연구

2.1 멀티코어 기반 소프트웨어 최적화

멀티코어 기반 Task 할당, 동기화 및 코어 종속 메모리에 대한 변수 할당과 같은 다양한 소프트웨어 최적화 기법이 멀티코어 프로세서에 사용된다.

2.1.1 병렬 처리

병렬 처리를 위해 각 기능 모듈은 적절한 코어에 할당된다. 기능 모듈 간의 의존성을 확인 후 병렬 구조로 만들 수 있는 기능 모듈은 다른 코어에 할당된다. 그림 1에서 볼 수 있듯이 일부 기능 모듈은 병렬로 실행되어 수행 시간을 절감할 수 있다.

2.1.2 동기화

그림 2와 같이 세마포어, 스핀락, 인터럽트, 이벤트 등과 같은 동기화 방법이 사용될 수 있다. 또한, 어떤 동기화 방법을 사용하는지에 따라 총 수행 시간이 달라질 수 있다. 상황에 맞는 적절한 동기화 방법을 선택해야 한다.

2.1.3 변수 할당

변수에 대한 메모리 액세스 시간을 최소화하기 위해 적절한 코어에 변수가 할당된다. 변수가 특정 코어 내의 메모리에 할당되면 액세스 시간을 줄일 수 있다. 따라서 사용된 변수에 적합한 코어를 식별할 수 있다면 그림 3과 같이 총 수행시간을 절감할 수 있다.

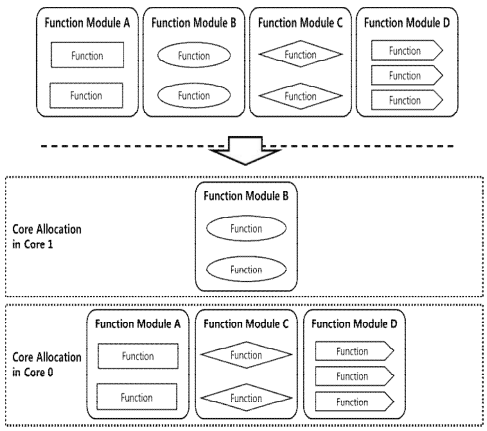

그림 1. 기능 모듈의 코어 할당
Fig. 1. Core allocation of function modules

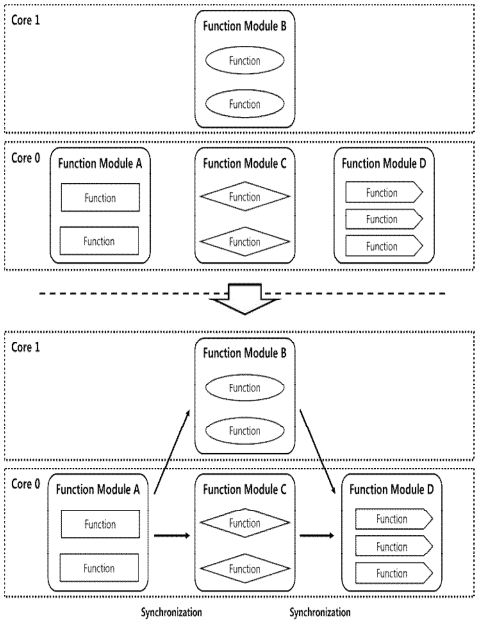

그림 2. 코어 메모리에 변수 할당
Fig. 2. Synchronization methods between cores

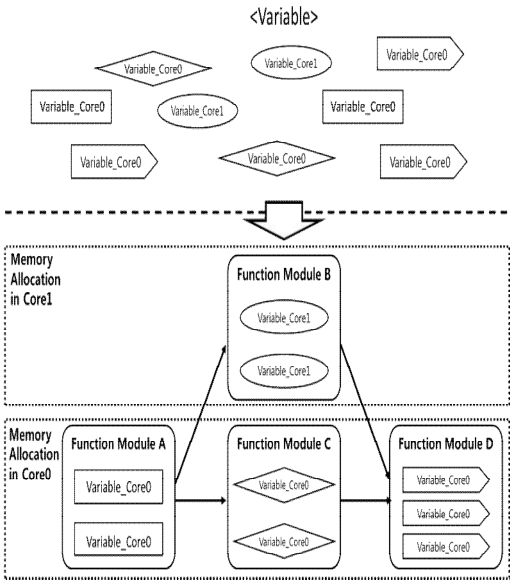

그림 3. 코어 메모리에 변수 할당
Fig. 3. Synchronization methods between cores

2.2 ESC 시스템

2.2.1 ESC 시스템 개요

ESC 시스템은 차량의 Yaw Movement를 제어하여 차량의 안정성을 향상시키는 능동적인 안전 시스템으로 뛰어난 안정성 덕분에 2010년대 출시된 모든 신차에 ESC 시스템은 의무적으로 적용되었다 [11]. 또한, 오버스티어링 및 언더스티어링 방지를 위해 차량의 각 휠은 독립적으로 제어되므로 ESC 시스템으로 안전성을 보장할 수 있다 [12].

그림 4와 5는 각각 ESC 시스템의 기본 구조와 제어 구조를 보여준다. 차량 속도, 4개 각각의 휠의 각속도, 차량의 Side slip angle, Yaw rate, Steering angle과 같은 차량 신호는 ESC 제어기에 입력으로 전송된다.

2.2.2 ESC 시스템 제어 구조

ESC 제어기는 차량에 대한 레퍼런스 모델, Yaw rate 제어기, Side slip angle 제어기로 구성된다. 레퍼런스 모델은 레퍼런스 Yaw rate를 얻기 위해 사용된다. 실제 Yaw rate와 레퍼런스의 차이에 기초하여 각 휠의 브레이크 토크는 Yaw rate 제어기에서 계산된다. 또한, 레퍼런스 Side slip angle는 4개 휠의 각속도와 실제 Yaw rate를 기반으로 Side slip angle 계산기에서 얻게 된다. 따라서 Side slip을 방지하기 위한 브레이크 토크는 레퍼런스 Side slip angle과 실제 Side slip angle의 차이를 사용하여 Side slip angle 제어기에서 계산된다. 최종 브레이크 토크는 그림 5와 같이 가변 Yaw motion을 제어한다.

2.2.3 시뮬레이션에 사용된 ESC 시스템

그림 6은 시뮬레이션에 사용된 ESC 시스템을 나타내며, [13]에서 사용한 시스템과 동일하다. 그림 4와 그림 6의 차이점은 Side slip angle을 Yaw rate 제어에 사용할 수 없다는 것이다. Yaw rate는 Slip ratio 기준으로 브레이크 토크를 조정하는 ABS(Anti-lock Braking System) 제어기를 사용함으로써 더 효과적으로 제어할 수 있다. 시뮬레

이션에는 7-DOF(Degree Of Freedom) 차량 모델과 2-DOF 레퍼런스 모델이 사용된다. 그림 7은 시뮬레이션에 대한 Yaw rate 및 Lateral positio n input/output 파형이다.

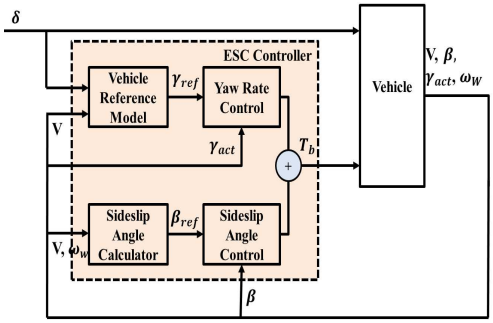

그림 4. 일반적인 ESC 시스템

Fig. 4. A typical ESC system

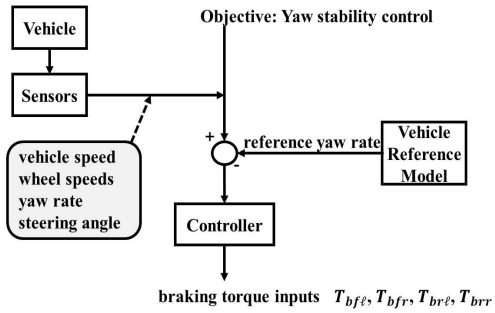

그림 5. 일반적인 ESC 시스템의 제어 구조

Fig. 5. Control architecture of a typical ESC system

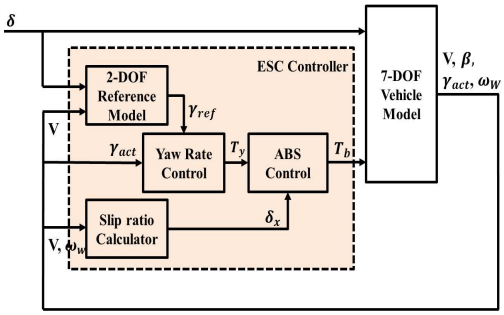

그림 6. ESC 시스템 시뮬레이터의 전체 구조

Fig. 6 Overall structure of the ESC system simulator

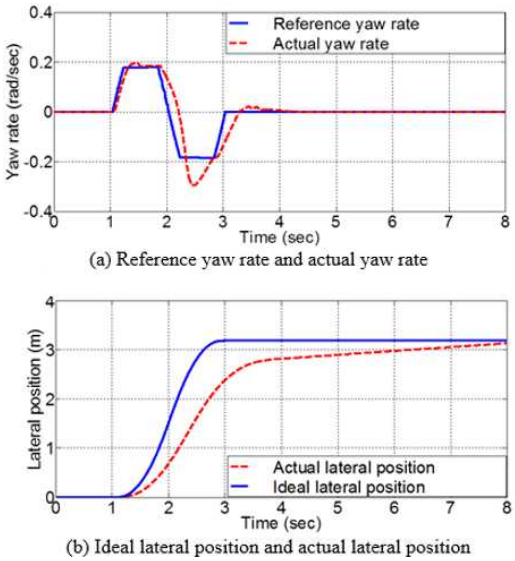

그림 7. ESC 시뮬레이터에 대한 input/output

Fig. 7. input/output for the ESC simulator

3. 차량용 멀티코어 프로세서를 위한 소프트웨어 최적화

이 절에서는 멀티코어 최적화 방법에 대해 설명한다. 그림 8과 같이 ESC 시스템의 멀티코어 설계에는 네 가지 방법이 사용된다. 첫째, 기능 모듈은 싱글코어 프로세서의 순차 실행에서 나뉜다. 각 기능 모듈에는 인터럽트 기능 또는 OS 작업에 할당된 특정 기능이 포함되어 있다. 둘째, 기능 모듈 간의 데이터 의존성을 고려하여 병렬 기능 모듈을 선택한다. 각 기능 모듈은 전체 성능 향상을 위해 적절한 코어에 할당된다. 셋째, 할당된 구조를 고려하여 세마포어 및 인터럽트와 같은 적절한 동기화 방법을 활용한다. 인터럽트는 코어 간의 동시 실행을 위해 사용된다. 세마포어는 기능 모듈 간의 일반적인 동기화를 위해 사용된다. 세마포어를 동시 실행에 사용할 수 있지만, 인터럽트보다 시간이 더 많이 소요된다. 마지막으로, 코어 간 공통 메모리 대신 코어 종속 메모리에 할당된다. 각 기능 모듈에 사용되는 변수를 결정한 후 적절한 코어 내 종속 메모리에 할당한다.

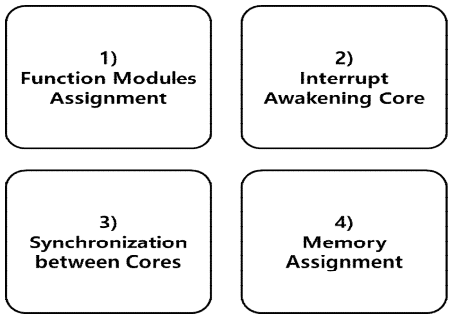

그림 8. 멀티코어 최적화 방법
Fig. 8. Multi-core optimization method

4. 멀티코어 기반 ESC 시스템 설계

이 절에서는 멀티코어 기반 ESC 시스템 설계를 소개하고 3절에서 논의한 방법을 적용한다. 싱글코어 컨트롤러에서 전반적인 성능 향상을 위해 멀티코어 컨트롤러로 재설계되었다. 그림 9는 ESC 시스템의 싱글코어 컨트롤러 구조를 보여준다. 차량용 멀티코어 프로세서를 위한 세 개의 코어 중 싱글코어 컨트롤러는 코어 0에서 구현되고 플랜트는 코어 1에서 실행된다. 제시된 멀티코어 컨트롤러는 코어 0 및 코어 2에서 설계되었다. 이후 절에서 기능 모듈 할당, 세마포어 및 인터럽트 적용, 변수 할당에 대해서 설명한다.

4.1 멀티코어 기반 ESC 시스템 기능 모듈 할당

그림 10은 ESC 시스템의 코어 0와 코어 2에 대한 기능 모듈 할당을 보여준다. 전체 컨트롤러는 초기화, Slip ratio 계산, Yaw rate 계산, ABS 제어, 통신의 5가지 기능 모듈로 나뉜다. 초기화 기능 모듈에서는 전체 시스템이 초기화된다. ABS 제어기는 Slip ratio 기능 모듈에서 얻은 브레이크 토크로부터 브레이크 토크 값을 조정하는데 사용된다. 조정된 브레이크 토크 값은 통신 기능 모듈과 함께 플랜트로 전송된다. 제시된 설계에서 Yaw rate 기능 모듈이 가장 시간이 많이 소요되므로 코어 2와 다른 기능 모듈은 코어 0에 할당된다.

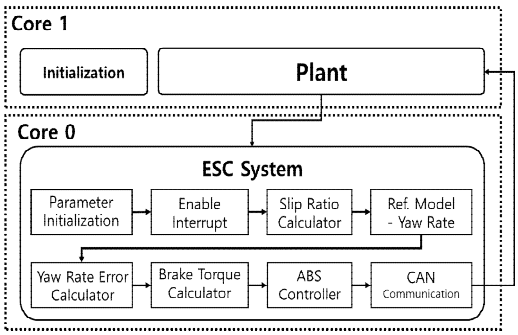

그림 9. 시뮬레이션에서의 싱글코어 ESC 시스템 구조
Fig. 9. Single-core ESC system architecture for the simulation

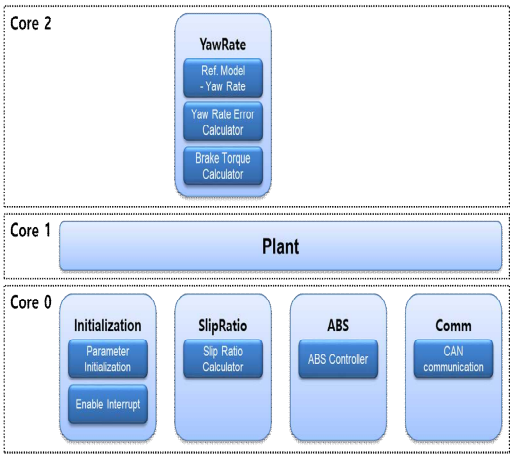

그림 10. ESC 시스템의 기능 모듈 할당
Fig. 10. Function module assignment for the ESC system

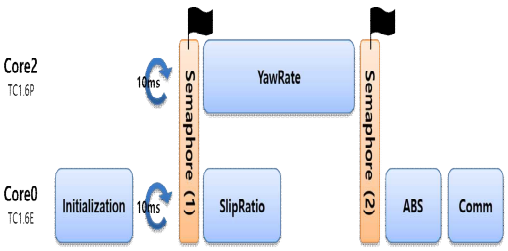

그림 11. 코어 간 동기화를 위한 세마포어
Fig. 11. Synchronization between cores using semaphore

4.2 세마포어를 이용한 코어 간 동기화

코어 간 동기화를 위해 두 개의 세마포어를 사용한다. 첫 번째 세마포어는 코어 간의 동시 실행을 위해 설계되었다. 코어 0의 SlipRatio 계산 기능 모듈은 코어 2의 YawRate 계산 기능 모듈을 실행시켜 준다. 다른 세마포어는 ABS 기능 모듈 실행에 사용된다. 제어 플로우에서의 세마포어의 위치는 그림 11과 같다.

4.3 동시 실행을 위한 인터럽트

세마포어는 인터럽트에 비해 소요 시간이 더 길다. 그림 12와 같이 첫 번째 세마포어는 인터럽트로 변경된다. SlipRatio 계산 기능 모듈 실행이 시작될 때 코어 2가 실행되고 YawRate 계산 기능 모듈이 실행된다. 따라서, 서로 다른 코어에서 두 개의 기능 모듈을 동시에 실행할 수 있다.

4.4 멀티코어 기반 변수 할당

기능 모듈을 실행하기 위해서는 각 코어가 메모리 영역에 접근해야 한다. 변수가 Infineon AURIX 프로세서의 LMU(Local Memory Unit)와 같은 공용 메모리에 할당 된 경우 수행 시간이 길어진다. 그림 13은 AURIX 프로세서의 공용 및 종속 메모리에 대한 메모리 위치를 보여준다. 특히, AURIX 프로세서의 경우 메모리 액세스 시간은 공용 메모리의 경우 5클럭, 종속 메모리의 경우 1클럭으로 차이가 크다. 표 1과 같이 코어의 기능 모듈 위치에 따라 변수는 적절한 코어에 할당된다.

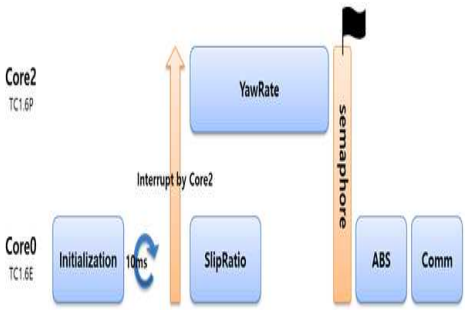

그림 12. 인터럽트를 통한 동시 실행
Fig. 12. Interrupt for simultaneous execution

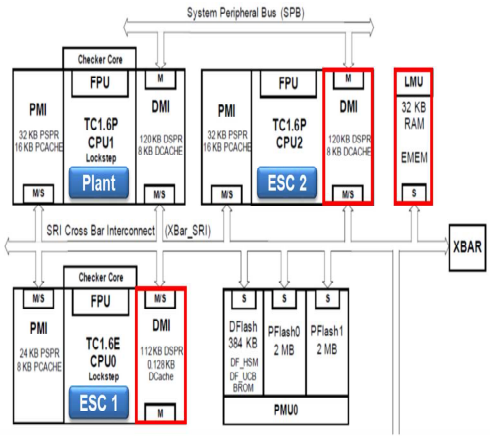

그림 13. Infineon AURIX 프로세서의 메모리 아키텍처
Fig. 13. Memory architecture of the Infineon AURIX processor

표 1. 코어 및 코어2의 변수 할당
Table 1. Variables assigned to Core 0, Core 2

Core 0.	Core 2	
ESC_wheel_speed_	ESC_brake_torque	BT_idx
ESC_wheel_speed_updated_	ESC_ref_yaw_rate	BT_idx_0
ESC_brake_torque_	ESC_speed_lat_	BT_idx_1
ESC_brake_torque_updated_	Core2_ESC_speed_lon_	BT_idx_2
Core0_ESC_speed_lon_	ESC_ref_yaw_angle_	ESC_ref_yaw_
ESC_speed_updated_		

5. 실험 결과

제안한 설계 방법은 Trace 32 디버거를 사용하여 Infineon AURIX TC275 보드를 통해 시뮬레이션 된다. 그림 14는 시뮬레이션에 사용된 장비 및 툴 환경을 보여준다. AURIX 프로세서의 코어 3개 중 코어 0과 코어 2는 멀티코어 컨트롤러 설계에 사용되고 코어 1은 플랜트에 사용된다. 표 2는 최적화 방법에 대한 수행 시간을 보여준다. 표 3은

각 기능 모듈의 수행 시간을 나타낸다. 시뮬레이션 800회 실행한 결과의 평균을 판단 지표로 사용하였다. 싱글코어 컨트롤러를 사용하는 경우 전체 실행 시간은 1,060 Ticks이다. 반면에, 본 논문에서 제안한 방법인 멀티코어에 대한 기능 모듈 할당, 세마포어 및 인터럽트를 통한 동기화, 각 코어 종속 메모리에 변수 할당을 통한 최적화를 완료한 후 수행 시간을 측정해보면 427 Ticks로 싱글코어 대비 수행 시간이 59.7%가 단축되었음을 알 수 있었다.

표 2. 최적화 방법에 대한 수행 시간

Table 2. Execution time depending on the optimization methods

Method	Execution Time (Ticks)	Reduction Rate (%)
Sequential Execution	1060.154	0
Function Modules Execution by Semaphore (2)	988.238	6.8
Function Modules Execution by Interrupt (3)	963.291	9.2
Memory Allocation in Each Core (4)	426.773	59.7

표 3. 기능 모듈에 따른 수행 시간 측정

Table 3. Execution time for each function module

Method	SlibRatio	YawRate	ABS	Sum
Execution Time (Ticks)	228.051	776.486	103.924	1108.461

(a) Infineon AURIX TC275 and Trace 32 Debugger

(b) Tasking VX-Toolset

그림 14. 개발 환경

Fig. 14. Development environment

6. 결론

이 논문에서는 시스템 성능 향상을 위해 차량 제어 시스템의 ESC 시스템을 위한 멀티코어 기반 컨트롤러가 제안된다. 그러나 멀티코어 소프트웨어가 정확하게 설계되지 않으면 성능이 저하될 수 있다. ESC 시스템의 전체 수행 시간은 차량용 멀티코어 프로세서와 ESC 시스템 구조를 분석한 후 멀티코어 최적화를 진행하였다. ESC 시스템은 기능 모듈 할당, 세마포어 및 인터럽트를 통한 동기화, 코어에 대한 변수 할당과 같은 멀티코어 최적화를 통해 싱글코어 컨트롤러에서 멀티코어 프로세서로 재설계 된다. 차량용 멀티코어 프로세서의 시뮬레이션 결과 수행 시간 측면에서 성능이 크게 향상된 것을 확인할 수 있었다. ESC 시스템은 루프 내 하드웨어 시스템과 통합되어야 한다. 향후 연구는 제안된 방법을 다른 차량 제어 시스템에 적용하는 방향으로 진행된다.

감사의 글

이 논문은 2021년도 산업통상자원부의 재원으로 한국산업기술평가관리원의 지원을 받아 수행된 한국산업기술평가관리원 사업의 연구결과입니다.(1415175290)

REFERENCES

[1] A. Biondi, M and Di Natale, "Achieving Predictable Multicore Execution of Automotive Applications Using the LET Paradigm“, 2018 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS), 2018.

[2] E. Díaz, E. Mezzetti, L. Kosmidis, J. Abella and F. Cazorla, “Modelling multicore contention on the AURIXTM TC27x”, DAC '18: Proceedings of the 55th Annual Design Automation Conference, 2018.

[3] C. Avasalcai, D. Budhrani and P. Pop, “Work-in-progress: towards industry strength mapping of AUTOSAR automotive functionality on multicore architectures,” 2017 International Conference on Compilers, Architectures and Synthesis For Embedded Systems (CASES), 2017.

[4] E. Sha, M. Xu, S. Gu,and Q. Zhuge, “Optimizing the data placement and scheduling on multi-port DWM in multi-core embedded system”, Journal of Systems Architecture, 2021.

[5] A. Abdi and H. R. Zarandi, "Improving the Reliability of Multicore Embedded Systems through an Evolutionary-based Task Scheduling Approach," 2021 29th Iranian Conference on Electrical Engineering (ICEE), 2021.

[6] J. Zhang, G. Cheng, C. Lu, T. Guo, J. Kang, X. Yan, X. Zhang amd X. Yuan., "Flow Data Task Scheduling Model of RTOS Based on Multicore Operation System," 2021 IEEE 11th International Conference on Electronics Information and Emergency Communication (ICEIEC)2021 IEEE 11th International Conference on Electronics Information and Emergency Communication (ICEIEC), 2021.

[7] M. Sayed, E. Saad, R. Aly and S. Habashy, “Energy-Efficient Task Partitioning for Real-Time Scheduling on Multi-Core Platforms”, Computers; Basel Vol. 10, Iss. 1, 2021.

[8] E. Ozer, B. Venu, X. Iturbe, S. Das, S. Lyberis, J. Biggs, P. Harrod and J. Penton, "Error Correlation Prediction in Lockstep Processors for Safety-Critical Systems," 2018 51st Annual IEEE/ACM International Symposium on Microarchitecture (MICRO), 2018.

[9] Infineon's Aurix TC27x reference manual, www.infineon.com

[10] G. Bloom, J. Sherrill, T. Hu, I. Bertolotti, “Real-Time Systems Development with RTEMS and Multicore Processors”, CRC Press, 2020.

[11] A. Lutz, B. Schick, H. Holzmann, M. Kochem, H. Tuve, O. Lange, Y. Mao and G. Tosolin, “Simulation methods supporting homologation of Electronic Stability Control in vehicle variants”, Vehicle System Dynamics, vol. 55, 2017.

[12] S. Yim, “Optimum Yaw Moment Distribution with Electronic Stability Control and Active Rear Steering,” Journal of Institute of Control, Robotics and Systems“, vol. 20, 2014.

[13] G. Jeong, T. Lee, J. Kim, J. Shin, J. Jeon. "Comparison of ESC Application Execution Time by Task Assignment Based on Multi-core ECU for Automotive“, Spring conference of the Korea Automotive Engineering Association, 2018.

저자약력

장 홍 순 (Hong-Soon, Jang) **[학생회원]**

- 2021.02: 국민대학교 전자공학과 학사
- 2021.03 ~ 현재: 국민대학교 전자공학과 석사과정

〈관심 분야〉 차량용 MCU, 차량 S/W 플랫폼 AUTOSAR

조 영 환 (Young-Hwan Cho) **[학생회원]**

- 2021.02: 국민대학교 전자공학과 학사
- 2021.03 ~ 현재: 국민대학교 전자공학과 석사과정

〈관심 분야〉 차량용 MCU, 차량 S/W 플랫폼 AUTOSAR

정 구 민 (Gu-Min Jeong) **[일반회원]**

- (2013 ~ 현재) 대학전기학회 정보 및 제어 부문, 이사
- (2017 ~ 현재) 한국모빌리티학회, 부회장
- (2017 ~ 현재) 한국정보전자통신기술학회, 부회장
- (2020 ~ 현재) 한국모빌리티학회, 부회장

〈관심 분야〉 자율주행 데이터, 자율주행 소프트웨어 플랫폼, 차세대 마이컴 기반 AUTOSAR 플랫폼