ISSN : 1226-7244 (Print)
ISSN : 2288-243X (Online)
논문번호 20-02-16

j.inst.Korean.electr.electron.eng.Vol.24,No.2,486~491,june 2020
http://dx.doi.org/10.7471/ikeee.2020.24.2.486

ARM Cortex-M0 DesignStart를 활용한 커스텀 시스템 설계 및 검증

Custom system design and verification using ARM Cortex-M0 DesignStart

이 성 룡*, 유 호 영*★

Sungryoung Lee*, Hoyoung Yoo*★

Abstract

ARM Cortex-M0 DesignStart provided by ARM is cost-free design development suit targeting for designing and prototyping SoC with Cortex-M0 core. In this paper, we presents a method how to implement a custom system design using ARM Cortex-M0 DesignStart. First, hardware elements for ARM Cortex-M0 DesginStart is analyzed focusing on bus and memory map, and next software toolchain is explained to clarify the translating process from high level language to binary machine language. As an example of the custom system, UART system operated with Cortex-M0 is designed and simulated.

요 약

본 논문은 ARM Cortex-M0 DesignStart를 기반으로 한 커스텀 시스템을 설계하는 방식을 정리하였다. ARM사에서 무료로 공개 배포하고 있는 ARM Cortex-M0 DesignStart를 활용하여 사용자가 원하는 ARM Cortex-M0 기반의 커스텀 시스템을 구현할 수 있다. 구현에 앞서 ARM Cortex-M0 기반의 하드웨어 구조와 특징을 살펴보고, 제공된 소프트웨어에 대하여 기술한다. 그리고 ARM Cortex-M0 DesignStart의 CMSDK(Cortex-M System Design Kit)을 활용하여 ARM Cortex-M0 기반의 UART 시스템 구현 및 테스트코드를 사용한 검증을 통해 ARM Cortex-M0의 커스텀 시스템 설계할 수 있음을 보인다.

Key words : ARM Cotex-M0, DesignStart, Custom Design, Tool Chain, UART

* Dept. of Electronics Engineering, Chungnam National University
★ Corresponding author
E-mail : hyyoo@cnu.ac.kr, Tel : +82-42-821-6585
※ Acknowledgment
This work was supported by the Brain Korea 21 Plus, by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (NRF-2019M3F3A1A01074449), and EDA tools were supported by IDEC, Korea.
Manuscript received Jun. 1, 2020; revised Jun. 14, 2020; accepted Jun. 17, 2020.

Ⅰ. 서론

4차 산업시대가 도래함에 따라 사물인터넷(IoT), 커넥티드 카(Connected Car), 인공지능(AI) 등 새로운 기술들이 대거 등장하고 있다[1]. 이러한 새로운 기술들에는 시스템 반도체가 필수적으로 요구되며 따라서 다양한 기업에서 시스템 반도체 시장을 장악하기 위해 끊임없는 변화를 시도하고 있다[2, 3].

기존의 시스템 반도체 시장에서는 ASIC 기반의 개발방식을 사용하였다. 본 방식을 통하여 목적에 최적화된 시스템 설계는 가능하였으나, 시간과 개발비용이 만만치 않아 즉각적인 수요 충족에 있어

한계가 존재하였다. 이러한 문제를 해결하기 위해 ARM, AMD와 같은 주요 시스템 반도체 회사에서는 사용자가 직접 목적에 따라 시스템을 구성할 수 있는 커스텀 시스템 설계용 IP(Intellectual Property) Core를 출시했다[4]. 이 방식은 기존 방식 대비 개발 시간을 눈에 띄게 단축시킴으로써 빠른 발전과 변화가 요구되는 여러 산업 군에서 필수적인 요소로 자리 잡았다. 이처럼 Core IP 기반으로 한 커스텀 시스템 설계의 중요성은 날이 갈수록 커지고 있지만, 그에 비해 개발자의 수는 턱없이 부족하며 엄격한 기술보호로 인해 개발 환경에 쉽게 접근 할 수조차 없다. 앞선 문제점을 파악하여 여러 국가에서는 시스템 반도체 개발자 육성을 위한 교육 프로그램을 지원하고 있으며 ARM사에서는 Core 커스텀 시스템 설계에 접근을 용이하도록 ARM Cortex-M0 [5], M3의 IP Core를 무료로 제공하는 대안을 제시하였다.

본 논문에서는 ARM 사가 제공하는 IP Core 중에서도 Cortex-M0 DesignStart [6]를 활용한 IP Core 기반의 커스텀 시스템 설계 과정을 단계별로 자세히 정리하여 여러 개발자들의 접근성을 높이는 것을 목표로 한다. 이를 위해 우선적으로 ARM Cortex-M0 DesignStart[6]의 하드웨어, 소프트웨어 구성을 설명하고 ARM Cortex-M0[4]의 특징을 설명한다. 그리고 이어서 ARM Cortex-M0 DesignStart[6]를 활용한 ARM Cortex-M0[5] 기반의 UART (Universal Asynchronous Receiver/ Transmitter) [7] 커스텀 시스템 구현 및 검증과정을 기술한다.

II. ARM Cortex-M0 Design Start

ARM Cortex-M0 DesignStart[6] 에디션은 ARM에서 제공하는 CMSDK(Cortex-M System Design Kit) [8]로 그림 1은 본 에디션의 하드웨어와 소프트웨어 구성을 나타내고 있다.

먼저 하드웨어의 경우, 크게 MCU(Micro Controller Unit)와 FPGA(Field Programmable Gate Array)로 구분되고 각각은 RTL(Resister Transistor Logic) 코드로 기술되어있다. MCU는 다시 암호화된 ARM Cortex-M0[5]와 GPIO(General-Purpose Input/Output)로 할당되는 내부 주변기기로 구성되어 있고 FPGA도 이와 유사하게 암호화된 ARM Cortex-M0 [5]와 ARM의 MPS2+보드의 외부 구성인 주변기기로

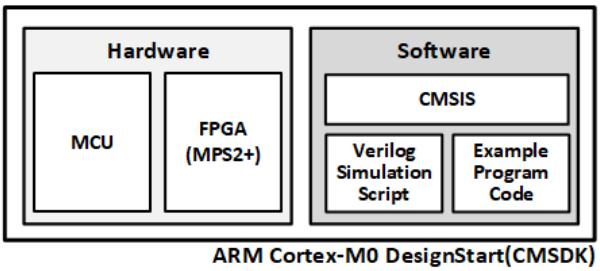

Fig. 1. ARM Cortex-M0 DesignStart Configuration.
그림 1. ARM Cortex-M0 DesignStart 구성

분리되어 있다. 이어서 소프트웨어는 CMSIS(Cortex Micro-controller Software Interface Standard)와 베릴로그 시뮬레이션 스크립트, 예제 프로그램 코드로 구성되어 있고 이를 활용하여 개발한 커스텀 시스템의 검증을 진행할 수 있다.

1. ARM Cortex-M0의 하드웨어 구조

그림 2를 통해 보다 자세한 ARM Cortex-M0 [5] 하드웨어 구조를 나타내었다. ARM Cortex-M0 [5] 하드웨어는 ARM에서 제공하는 버스 시스템인 AMBA (Advanced Microcontroller Bus Architecture)와 메모리 맵 시스템 (Memory mapped system) [9]로 특징지어진다.

가. AMBA

AMBA는 ARM에서 제공하는 버스시스템으로 일종의 규격 인터페이스를 가지고 있다.

이를 이용하여 커스텀-IP를 연결하면 원하는 시스템 구성이 가능하다. AMBA의 대표적인 버스로는 APB(Advanced Peripheral Bus)[10], AHB (Advanced High-Performance Bus)[11]가 있다.

APB [10]는 저속 버스이며 하나의 마스터에서 여러 개의 슬레이브(Slave)를 컨트롤하는 형태로 구성된다. APB [10]는 간단한 인터페이스와 저 전력인 특징을 가지고 있어 주로 I/O 및 주변 장치를 연결하는데 쓰인다.

반면 AHB [11]는 고속버스로 다수의 마스터와 다수의 슬레이브를 연결하며 중간에 디코더 (Decoder)와 아비터 (Arbiter)가 중재하는 형태로 구성된다. AHB[11]는 버스트 트랜스퍼(Burtst Transfer)[12]와 파이프라인(Pipeline)[13] 동작할 수 있는 특징을 가지며 주로 메모리와 프로세서를 연결하는데 쓰인다.

ARM Cortex-M0[5] APB[10]와 AHB[11]는 브

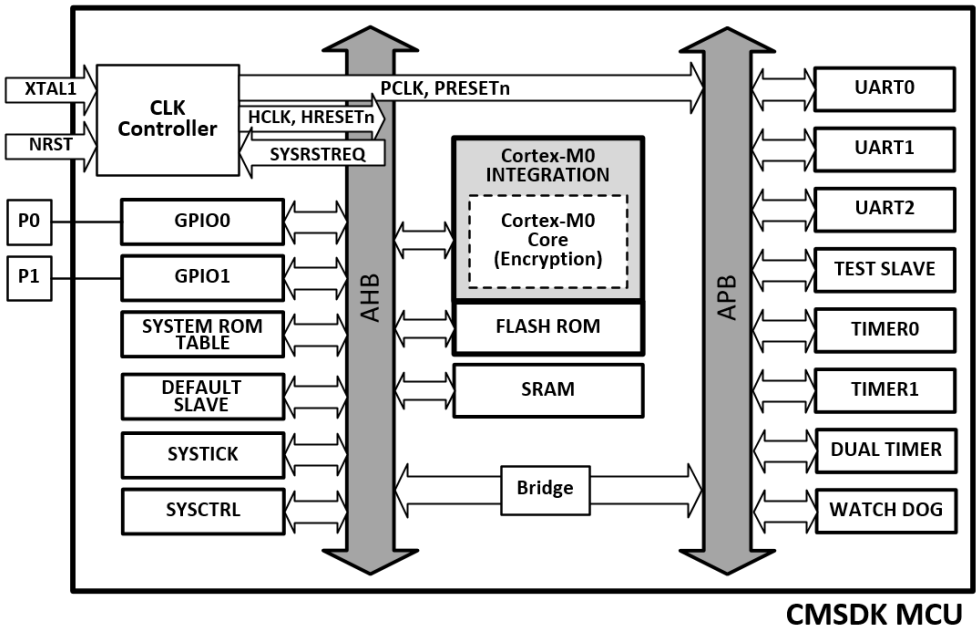

Fig. 2. CMSDK MCU Configuration.
그림 2. CMSDK MCU 구성

릿지(Bridge)로 연결되어 프로토콜 호환이 가능하도록 구성된다. 사용자는 시스템 설계 목적에 따라 APB[10], AHB[11]에 주변기기(Peripheral)를 연결하여 원하는 시스템을 구현할 수 있다.

나. 메모리 맵 시스템

ARM Cortex-M0 [5]는 그림 3과 같이 입출력장치를 액세스 할 때, 입출력과 메모리의 주소 공간을 분리하지 않고 하나의 메모리 공간인 메모리 맵 시스템으로 구성된다. 각 주변기기들은 주소에 할당되며 사용자는 커스텀 설계의 목적에 따라 주소 범위를 정하고 해당 주소에 주변기기를 연결하여 커스텀 시스템을 구현하게 된다.

AHB Address	Peripheral
0x00000000 – 0x0000FFFF	64K FLASH / Boot Firmware
0x01000000 – 0x0100FFFF	Boot Firmware (only 4k is used)
0x20000000 – 0x2000FFFF	SRAM
0x40000000 – 0x4000FFFF	APB
0x40010000 – 0x4001FFFF	AHB Peripherals
0xF0000000 – 0xF0000FFF	System ROM Table
APB	
0x40000000 – 0x40000FFF	TIMER 0
0x40001000 – 0x40001FFF	TIMER 1
0x40002000 – 0x40002FFF	DUALTIMER0
0x40004000 – 0x40004FFF	UART0
0x40005000 – 0x40005FFF	UART1
0x40006000 – 0x40006FFF	UART2
0x40008000 – 0x40008FFF	Watchdog
0x4000B000 – 0x4000BFFF	TEST SLAVE
AHB Peripherals	
0x40010000 – 0x40010FFF	GPIO 0
0x40011000 – 0x40011FFF	GPIO 1
0x4001F000 – 0x4001FFFF	SYSCTRL

Fig. 3. CMSDK MCU Memory Map.
그림 3. CMSDK MCU 메모리 맵

2. ARM Cortex-M0의 소프트웨어 구조

ARM Cortex-M0 [5]기반의 커스텀 시스템 설계한 뒤, 설계 작동을 위해 PC에서 어플리케이션(Application) 코드를 설계해야 한다.

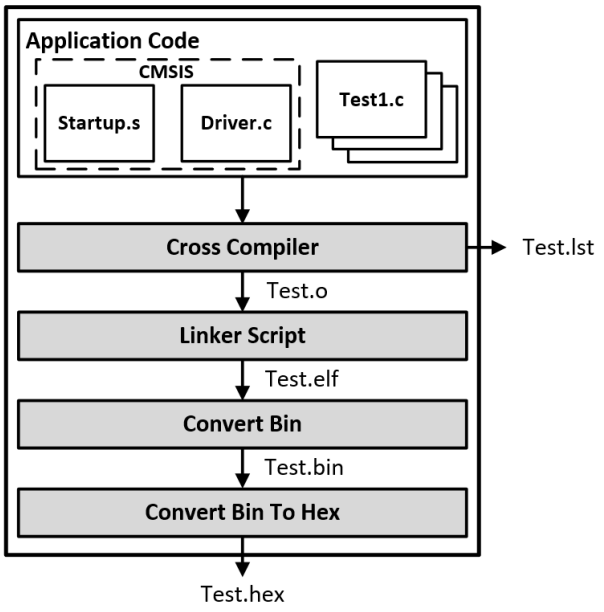

Fig. 4. ToolChain Configuration.
그림 4. 툴체인 구성

사용자는 하드웨어 표준화된 소프트웨어 인터페이스인 CMSIS를 기반의 어플리케이션 코드를 통해 시스템의 동작을 구성한다. 그리고 작성한 코드는 ARM Cortex-M0[5] 기반의 명령어(Instruction)로 변환하는 과정이 필요하다. 이를 처리해주는 것이 바로 툴 체인 [14]이며 그림 4와 같이 링커 스크립트(linker Script), 크로스 컴파일러(Cross Compiler), 컨버터(Converter)로 구성된다.

PC의 코어가 ARM Cortex-M0[5] 기반이 아닌 다른 종류의 코어 기반일 경우, 기존의 컴파일러로 변환할 경우 커스텀 시스템 동작에 오류가 발생한다. 따라서 PC의 코어 기반으로 컴파일 할 경우에는 어플리케이션 코드를 ARM Cortex-M0[5]에서 동작 가능한 명령어로 변환하는 과정이 필요하다. 이러한 과정을 위해 크로스 컴파일러를 이용하여 ARM Cortex-M0[5]기반의 오브젝트(.o) 바이너리 파일을 생성한다. ARM Cortex-M0 DesignStart [6]에서는 ds-5, gcc, keil[15], iar 4가지의 크로스 컴파일러를 지원하며 본 논문에서는 GCC 컴파일러를 활용하여 어플리케이션 코드를 실행 가능한 명령어로 변환하는 과정을 진행하였다. 그림 4와 같이 크로스 컴파일러인 GCC를 활용하여 오브젝트 파일인 .o파일과 리스트 파일인 .lst 파일을 생성한다. 오브젝트 파일은 링커스크립트의 입력으로 사용되고, 리스트 파일은 디버깅(debugging)을 하는 과정에 사용 할 수 있다[16, 17]. 크로스 컴파일러의 의해 생성된 바이너리 파일들은 메모리로 할당하기 전 각 메모리의 해당 섹션으로 위치시켜야 된다. 링커 스크립트는 이러한 과정을 진행하고 결과적으로 할당된 정보를 포함한 .elf 확장자 파일을 출력한다. 생성된 .elf 파일은 .hex 파일로 변환하여 ARM Cortex-M0[5]기반 시스템의 명령어 메모리(Instruction Memory)의 명령어(Instruction)로 활용된다.

ARM Cortex-M0 DesignStart[6]는 시뮬레이션을 위해 베릴로그 시뮬레이션 스크립트를 제공하고 있다. VCS 혹은 NC-verilog를 사용할 수 있으며 RTL 디자인과 어플리케이션 코드를 각각 컴파일 한 후 시뮬레이션을 진행한다.

III. UART 커스텀 시스템 설계 및 검증

테스트를 위해 그림 5와 같이 ARM Cortex-M0 [5] 기반의 UART [7] 시스템을 구현하였다. 동작 클럭은 ARM Cortex-M0 [5]의 최대 동작주파수인 50MHz를 사용하였으며 UART [7]로는 UART1(TX), UART2(RX)를 사용하였다. GPIO1의 TX 핀을 GPIO1의 RX으로 직접 연결하여 송신된 데이터가 바로 수신될 수 있도록 구성하였다. UART 보율(Baudrate) [7]은 식(1)과 같이 동작 주파수 50MHz 기준으로 Baudrate divider를 434으로 설정하여 115200bps에 근사한 115207bps라는 보율 값으로 설정하였다.

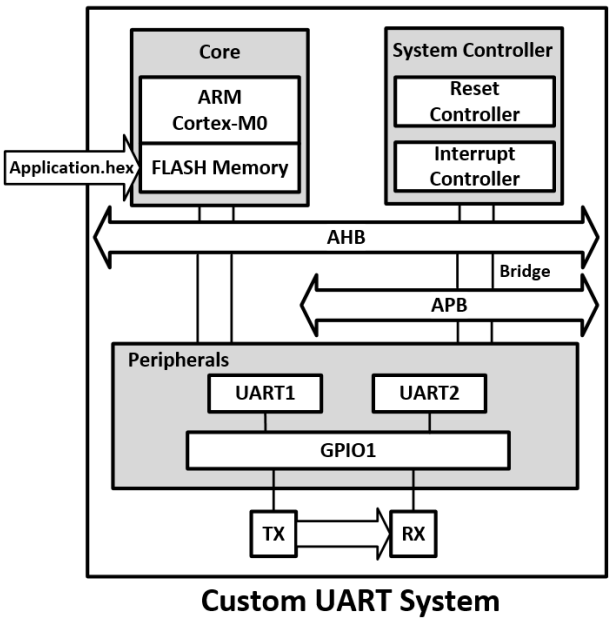

Fig. 5. UART System based on ARM Cortex-M0.
그림 5. ARM Cortex-M0기반의 UART 시스템

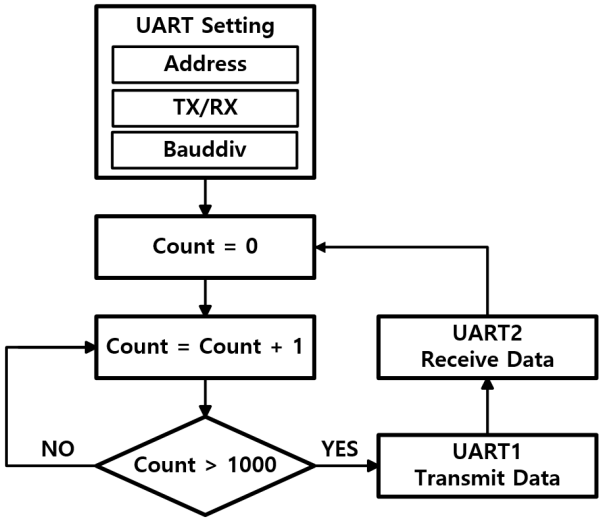

Fig. 6. Application code Flow Chart.
그림 6. 어플리케이션 코드 순서도

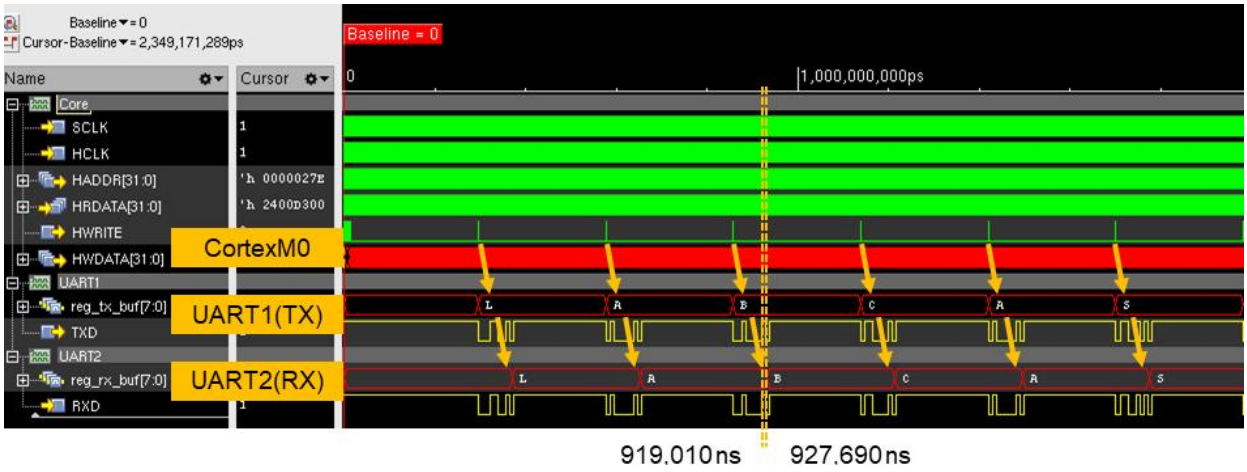

Fig. 7. UART simulation result.
그림 7. UART 시뮬레이션 결과

$$\text{Baudrate} = \frac{\text{InternalClock}}{\text{Baudrate Divider}} = \frac{\text{bits}}{\text{bits sec}} \quad (1)$$

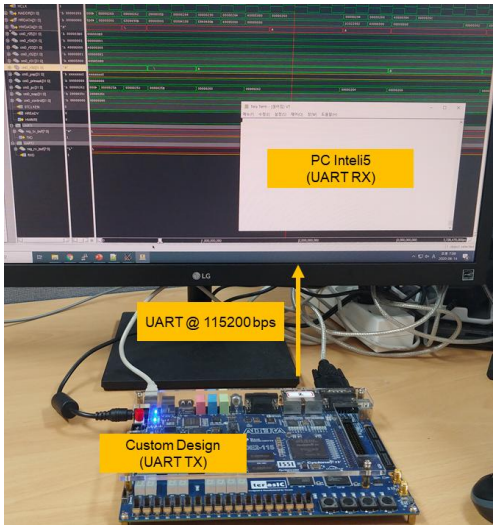

Fig. 8. FPGA implementation result.
그림 8. FPGA 구현 결과

앞서 기술한 설정 값을 적용한 어플리케이션 코드 코드는 그림 6과 같이 구성하였고, GCC를 활용하여 컴파일을 진행하였다. 헥사(hex) 파일 형태로 변환된 어플리케이션 코드는 그림 5과 같이 플래시 메모리에 초기화되며 플래시 메모리는 부팅 시 코어에서 명령어 메모리로서 동작한다. 초기 부팅 과정에서 코어는 MSP(Main Stack Pointer), 리셋 핸들러(Reset Handler)와 같은 전 처리과정을 진행하게 된다. 그 후 그림 6에 기술된 어플리케이션 코드가 수행되는데 예제에서는 “LABCAS”라는 ASCII 데이터를 순차적으로 송신하고 수신하였다. 그림 7은 Core에 들어온 “LABCAS”라는 ASCII데이터가 UART1에 전달되고 UART 시리얼 통신을 통해 UART2에 수신되는 일련의 과정을 나타내고 있다. 추가적으로 그림 7의 UART 한 비트의 지연시간이 8680 ns이고 이의 역수는 115207이므로 설정한 보율에 맞게 UART 시리얼 통신이 수행되고 있음을 확인할 수 있다. 끝으로 그림 8은 검증을 위하여 커스텀 디자인을 Intel FPGA DE2-115 보드에 구현한 시스템을 보여주고 있다. FPGA 보드는 50MHz 주파수로 동작하였으며 Baudrate는 115200bps로 할당하였다. FPGA 구현 결과 또한 시뮬레이션 결과와 동일함을 확인하였다.

Ⅳ. 결론

본 논문에서는 ARM Cortex-M0 DesignStart[6]를 활용하여 ARM Cortex-M0[5] 기반의 UART[7] 시스템을 구현하고 검증하였다. ARM에서 제공하는 ARM Cortex-M0 DesignStart[6]은 암호화된 코어와 주변기기에 대한 코드를 MCU와 FPGA 형태로 제공하여 간편하게 커스텀 시스템 설계가 가능하다.

References

[1] Y. J. Min and C. Y. Choi, “ICT Industry and Policy Trend Analysis in the 4th Industrial Revolution,” *E-biz*, 21, 2, pp. 103-118, 2020. DOI: 10.20462/TeBS.2020.04.21.2.103

[2] K. Y. Ahn, “FuriosaAI-developing specialized AI chips,” *Communications of Korean Institute*

of Information Scientists and Engineers, Vol.38, No.3, pp.37-38, 2020.
[3] STMicroelectronics, "Product," https://www.st.com/content/st_com/en.html
[4] ARM, "Processor," https://developer.arm.com/ip-products/processors
[5] ARM, "CortexTM-M0 Devices Generic User Guide," 2009.
[6] ARM, "Arm® Cortex®-M0 DesignStartTM Eval Revision: r2p0," 2017.
[7] J. Norhuzaimin and H. H. Maimun, "The design of high speed UART," *2005 Asia-Pacific Conference on Applied Electromagnetics. IEEE*, 2005. DOI: 10.1109/APACE.2005.1607831.
[8] ARM, "Cortex-M System Design Kit Technical Reference Manual," 2011.
[9] Y. Joseph, *The Definitive Guide to ARM® Cortex®-M0 and Cortex-M0+ Processors. Academic Press*, 2015.
[10] ARM, "AMBA 3 APB Protocol Specification v2.0," 2008.
[11] ARM, "AMBA 3 AHB-Lite Protocol Specification v1.0," 2008.
[12] Y. Yamagata and A. Yamawaki, "An Evaluation of Burst Transfer Inferred by a High-level Synthesis Tool," *TENCON 2018-2018 IEEE Region 10 Conference*. IEEE, 2018.
DOI: 10.1109/TENCON.2018.8650448
[13] G. Y. Jeong, J. S. Park, and S. C. Kim, "A Study on Multiplier Architectures Optimized for 32-bit RISC Processor with 3-Stage Pipeline," *The Institute of Electronics and Information Engineers*, pp.123-130, 2004.
[14] V. Kumar, "Embedded Programming with the GNU Toolchain," 2011.
[15] ARM, "ARM® Compiler v5.06 for μVision® Version 5," 2015.
[16] ARM, "Procedure Call Standard for the Arm Architecture," 2019.
[17] ARM, "ARM® Compiler armasm User Guide v6.6," 2016.

BIOGRAPHY

Sungryoung Lee (Member)

2019 : BS degree in Electronics Engineering, Chungnam National University.
2020~ : MS degree in Electronics Engineering, Chungnam National University.

Hoyong Yoo (Member)

2010 : BS degree in Electrical & Electronic Engineering, Yonsei University
2012 : MS degree in Electronics Engineering, KAIST
2016 : Ph.D. degree in Electronics Engineering, KAIST
2016 : Researcher, Samsung Electronics.
2016~ : Assistant Professor, Chungnam National University