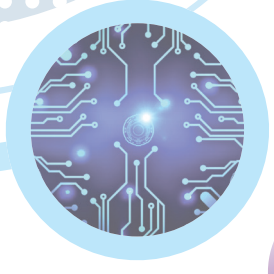
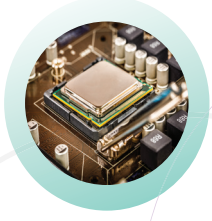
IDEC newsletter

Vol. 231 September 2016





[IDEC 논단]

컴퓨터의 성능을 결정짓는 가장 중요한 요소

CPU 아키텍처의 역사와 시사점

[기술동향칼럼]

하나의 칩에 집적 내장되어 다양한 기능을 수행한다

Multiprocessor System-On-Chip의 태스크 매핑 기술동향

[기획칼럼]

우리 일상에 어느새 가까워지고 있는

하늘을 나는 로봇 : 드론

반도체설계교육센터
IC DESIGN EDUCATION CENTER

2016년 MPW 모집내역

⊙ 2016년 MPW 설계팀 모집 완료 : 총 295팀 설계 참여
⊙ 2017년 MPW 진행일정 : 2016년 12월 말 결정 예정

MPW 진행일정 및 공정지원 내역

공정	회차구분 (공정_년도순서)	우선모집 (마감일)	정규모집 (마감일)	참여팀수/ 제작칩수	DB마감 (Tape-out)	Die-out	비고
삼성 65nm	S65-1601		2016.02.01	38 / 40	2016.08.01	2017.02.14	제작대기중
	S65-1602		2016.04.18	35 / 40	2016.10.17	2017.05.02	설계중
	S65-1603	2016.04.18	2016.06.20	40 / 40	2017.01.16	2017.07.31	설계중
MS 0.18um	MS180-1601		2016.01.18	29 / 25	2016.03.21	2016.08.22	제작완료
	MS180-1602		2016.02.01	29 / 25	2016.05.16	2016.10.17	칩제작중
	MS180-1603		2016.03.07	26 / 25	2016.07.18	2016.12.19	칩제작중
	MS180-1604	2016.02.01	2016.04.04	26 / 25	2016.09.19	2017.02.20	설계중
	MS180-1605	2016.04.04	2016.06.07	32 / 25	2016.12.05	2017.05.08	설계중
MS 0.35um	MS350-1601		2016.02.01	20 / 20	2016.06.13	2016.10.04	칩제작중
	MS350-1602	2016.05.02	2016.07.04	22 / 20	2017.01.16	2017.05.08	설계중

- 일정은 사정에 따라 다소 변경될 수 있음.
- 회차 표기방법 변경 : 공정코드–년도 모집순서 (예시) 삼성 65nm 2016 1회차 : S65–1601)
- 모집기간 : 모집 마감일로 부터 2주 전부터 접수
- Package 제작은 Die out 이후 1개월 소요됨.
- 기준 : 2016.08.30. (화)

문의처

⊙ 042-350-4428, yslee@idec.or.kr
⊙ IDEC 홈페이지 (http://idec.or.kr)

2016년 09월 교육프로그램 안내

수강을 원하는 분은 IDEC 홈페이지(www.idec.or.kr)를 방문하여 신청하시기 바랍니다.

⊙ 강좌 일정

센터명	강의일자	강의제목	분류
본센터	9월 2일	Innovus Implementation System(Block)	Tool강좌
	9월 7–9일	Xilinx ZYNQ Device 설계 교육	설계강좌
	9월 22–23일	Calibre xRC	Tool강좌
	9월 26–28일	AMBA AXI 기반 IP 설계와 검증	설계강좌
	9월 30일	Wireless communication system의 이해와 RF IC 설계 기초	설계강좌
성균관대	9월 10, 17, 24일	Verilog HDL 설계의 기초	설계강좌
	9월 29–30일	고성능 데이터 변환기	설계강좌
전남대	9월 29–30일	자동차 전장통신 CAN_FlexRay 통신설계 및 응용	설계강좌
전북대	9월 7일 – 12월 14일	S2S 과정 : 2단계 회로 설계	설계강좌

- 강좌일 : 9월 2일
- 강좌 제목 : Innovus Implementation System(Block)
- 강사 : 양희영 과장(Cadence Korea)

[강좌개요]

Cadence P&R Tool Encounter를 이용하여 전체 flow에 대한 설명 및 lab을 진행

[수강대상]

대학원생, 학부생, 관련 회사원

[강의수준]

중급

[강의형태]

이론+실습

[사전지식, 선수과목]

반도체공학, Physical Implementation

- 강좌일 : 9월 7–9일
- 강좌 제목 : Xilinx ZYNQ Device 설계 교육
- 강사 : 김민석 팀장(리버트론)

[강좌개요]

Xilinx ZYNQ 디바이스 Architecture 이해를 기반으로 Vivado SW 환경을 이용하여 ZYNQ 활용 프로젝트 진행과 디버깅 실습, 하드웨어를 기반으로 하는 설계 및 검증 진행

[수강대상]

Xilinx ZYNQ Device를 사용해야 하는 HW 엔지니어

[강의수준]

중급

[강의형태]

이론+실습

[사전지식, 선수과목]

Xilinx Vivado 사용 경험, HDL 기본 지식 및 Xilinx FPGA 사용 경험

- 강좌일 : 9월 22–23일
- 강좌 제목 : Calibre xRC
- 강사 : 변선수 과장(Mentor Graphics)

[강좌개요]

Calibre xRC를 사용하여 Layout 상의 Parasitic 저항 및 캐패시터를 추출하는 방법 및 Rule file generation과 Calibre xRC의 다양한 기능적인 부분에 대한 실습 병행 교육 진행

[수강대상]

Calibre xRC User

[강의수준]

초급

[강의형태]

이론+실습

[사전지식, 선수과목]

Calibre nmDRC/nmLVS에 대한 경험이 필요하나, 기본적인 내용이 포함되어 있어 처음 Tool을 사용하시는 분도 가능

- 강좌일 : 9월 26–28일
- 강좌 제목 : AMBA AXI 기반 IP 설계와 검증
- 강사 : 기안도 박사(다이나릿 시스템)

[강좌개요]

최근 반도체 칩에 주로 사용하고 있는 AMBA AXI를 기초부터 버스 기반 설계를 검증하는 기법으로 C–program 구동 BFM(Bus Functional Model)을 살펴보고, BFM을 사용하여 AXI 버스용 IP를 설계하고 검증하는 방법을 메모리, UART, Video controller, Camera controller에 적용

[수강대상]

시스템반도체 설계, 검증 업무 담당자

[강의수준]

초중급

[강의형태]

이론+실습

[사전지식, 선수과목]

Verilog design and simulation and synthesis, C programming, Text editor (e.g., vi or vim)

- 강좌일 : 9월 30일
- 강좌 제목 : Wireless communication system의 이해와 RF IC 설계 기초
- 강사 : 변영재 교수(UNIST)

[강좌개요]

MOS 기본 physics 복습을 통한 Wireless Principle들을 점검하고, RF/Wireless IC 설계에 없어서는 안되는 RLC 소자들을 Wireless IC 설계 입장에서 재조명 해본다. Passive 소자에 대한 이해를 바탕으로 Distributed System을 공부하고, S–parameter, Smith Chart 및 bandwidth estimation technique을 숙지한다. 이를 기초로 고주파 영역의 증폭기, LNA, MIXER, PLL 설계를 공부하고 대표적인 architecture 실용 예제를 분석하여 RF IC 설계에 대한 이해도를 높인다.

[수강대상]

전공자 중 대학교 4학년 및 석사 1–2년, 직장인 및 타전공자

[강의수준]

초급

[강의형태]

이론+실습

[사전지식, 선수과목]

전자회로 1, 2

문의

KAIST IDEC 이한나 (042–350–8536, lhn1224@idec.or.kr)

서울대학교 혼성신호IC및시스템(MICS) 연구실, 트랜지스터 지식이 없이도 아날로그 IC를 설계할 수 있는 아날로그 셀 라이브러리(Analog Cell Library; ACL)를 IDEC MPW(Multi-Project Wafer)의 삼성 65nm 공정용으로 제작 및 배포

서울대학교 전기정보공학부 김재하 교수의 혼성신호IC 및 시스템 연구실(이하 MICS 연구실, http://mics.snu.ac.kr)은 아날로그 IC의 대표적인 예인 위상동기루프(Phase-Locked Loop; PLL)를 트랜지스터 기반이 아닌 셀 기반으로 쉽게 설계하고 이를 실제 칩으로 제작할 수 있는 아날로그 셀 라이브러리(Analog Cell Library; ACL) 및 모델 기반의 설계방법을 KAIST 반도체설계교육센터(IDEC)를 통해 국내 각 대학에서 사용할 수 있도록 배포를 시작했다고 밝혔다. 이 ACL을 활용하면, IDEC의 대학 칩제작 지원 프로그램에서 제공되는 삼성 65nm CMOS 공정을 통해 원하는 PLL을 빠르면 단 하루만에 설계하고 최종 GDSII 파일을 제출할 수 있다.

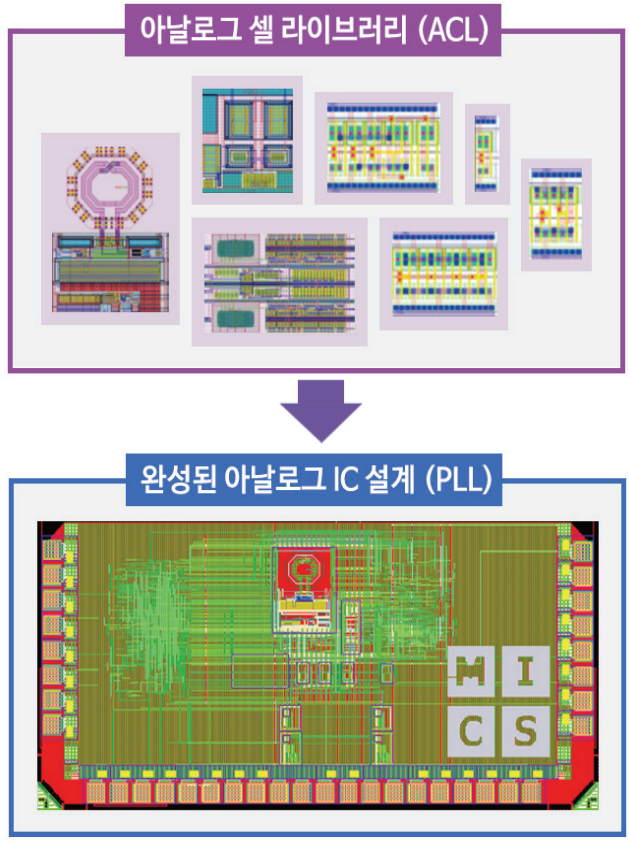
IC 설계는 다루는 신호의 성격에 따라 크게 아날로그 IC와 디지털 IC의 설계로 나뉘어지는데, 디지털 IC는 설계자동화툴의 발전에 힘입어 마이크로프로세서 등 복잡한 시스템도 모델과 표준 셀 라이브러리(standard cell library)를 활용해 단기간에 개발이 가능한 반면, 아날로그 IC는 상대적으로 작은 회로임에도 불구하고 설계자의 직관과 경험, 수작업에 의존한 기존 설계방식 때문에 사물인터넷(IoT) 등 아날로그 회로가 다수 필요한 응용분야에서 큰 걸림돌로 여겨져왔다. 이번에 서울대학교 MICS 연구실에서 배포하는 ACL은 셀 기반의 설계방식이 디지털 설계뿐만 아니라 아날로그 설계에도 적용될 수 있다는 것을 처음 보였다는 데에 큰 의미가 있다.

즉, 아날로그 IC도 디지털 IC처럼 동작 기술언어(behavioral language)를 통해 원하는 동작을 모델로 기술하고 모델 기반의 시뮬레이터를 통해 그 성능을 확인한 후 셀 라이브러리와 합성툴, P&R 툴을 통해 트랜지스터 수준의 회로에 대한 전문가적 이해 없이도 쉽게 IC의 설계, 검증, 제작이 가능한 길이 열린 것이다. ACL은 모델기반 시뮬레이터로는 사이언티픽아날로그사(Scientific Analog, Inc., http://www.scianalog.com)의 XMODEL을 사용하며, 합성 및 P&R 툴로는 Synopsys사의 Design Compiler과 IC Compiler를 사용한다.

이 ACL을 활용하면 8GHz 대역의 LC PLL 또는 1~2GHz 대역의 ring PLL을 다양한 아이디어로 설계할 수 있으며, MICS 연구실은 이미 이를 활용해 설계한 IC들을 ISSCC, VLSI 등 세계 최정상급 학회에 발표하여 그 성능의 우수성을 인정받은 바 있다. 제공된 ACL은 현재 삼성 65nm 공정 사용을 위한 비밀유지계약을 체결한 설계자에 한해 KAIST, 한양대 등 10개 대학교 11개 연구실에서 설계에 활용하고 있다.

김재하 교수는 "이 ACL이 대학들이 아날로그 설계인력 양성에 필요한 교육기간을 단축시키고, 기업들이 빠른 시장 변화에 대응한 아날로그 IC 제품을 적기에 개발하는데 도움을 주기를 기대한다."고 말했다. 김재하 교수는 아날로그 모델기반 시뮬레이터인 XMODEL을 개발한 사이언티픽아날로그사의 창업자이기도 하다. XMODEL은 20개 이상의 국내외 기업과 대학들에서 활발하게 사용하고 있다.

IDEC은 멀티프로젝트웨이퍼(MPW) 칩 제작 지원, 반도체설계자동화(EDA) 툴 지원, 다양한 교육프로그램을 제공하며 학교와 기업을 연결하는 역할을 하고 있다. 박인철 IDEC 소장은 "이 ACL로 인해 아날로그 설계의 초점이 단위회로 설계에서 지능형반도체를 위한 시스템 설계로 옮겨가기를 기대한다"고 말했다.



CPU 아키텍처의 역사와 시사점

박일 소장 (SK하이닉스 메모리시스템연구소)

간단히 표현한다면 컴퓨터는 연산을 담당하는 CPU(Central Processing Unit), 데이터의 제공과 임시저장을 담당하는 메모리, 그리고 데이터를 영구히 보관하는 storage로 이루어져 있다. 이 중 CPU는 컴퓨터의 성능을 결정짓는 가장 중요한 요소로 자리잡아 왔다. 따라서 컴퓨터의 성능을 이야기하는 것은 내장된 CPU의 성능을 이야기하는 것과 동일시 되어왔으며, 이는 마치 인간이 숨을 쉬는 것만큼 자연스럽게 받아들여져 왔다. 컴퓨터라는 단어는 English WriteRichard Braithwaite가 1613년에 쓴 The Yong Mans Cleanings라는 책에서 처음으로 나왔다. 이때는 계산을 잘하는 사람의 의미였으며, 연산기계의 의미가 된 것은 19세기 무렵이다. 기계적인 형태의 컴퓨터는 1940년대 이후 진공관을 사용해 만들어졌으며, 1950년대 중반 이후에는 트랜지스터의 발전으로 컴퓨터도 커다란 도약을 맞이한다. 그 후 Integrated Circuit의 개발로 1960년대 중반에 또 한번의 격변기를 거치게 된다. 컴퓨터는 이렇게 생각보다 긴 역사를 가지고 있으나, 우리에게 익숙한 컴퓨터 구조는 1970년대 이후에 갖추어졌다. 이 글에서는 1970년대부터 컴퓨터의 주요 구성요소가 된 CPU 혹은 Microprocessor의 역사와 변화의 이유를 이야기하려 한다.

CPU의 기본동작은 다음 3가지로 이루어 진다; 1) Fetch: 명령어를 메모리에서 가져오기, 2) Execution: 명령어에 따라 연산을 수행하기, 3) Commit: 수행결과를 메모리나 스토리지에 보내 컴퓨터의 정보를 변경하기. CPU의 역사는 이 3가지를 어떻게 구현하여 성능을 향상시키느냐의 이야기이다. 초기의 CPU는 하나의 명령어를 fetch하고 execution하고 commit을 하는 것이 하나의 step으로 되어 있었으며, 한 step이 끝나야 다음 명령어를 수행할 수 있었다. 한 step은 한 clock과도 유사하니, 초기 CPU는 IPC(Instruction Per Clock)가 1이었다고 볼 수 있다. CPU의 최대성능은 1IPC라는 생각은 매우 오랫동안 진리처럼 받아들여졌다.

CISC의 출현

1960년대에는 메모리의 크기가 매우 작았다. 따라서 작고 빠른 메모리에 가급적 많은 작업내용을 지시하는 명령어를 포함시키는 것이 컴퓨터 성능의 중요한 요소였다. 그러다보니 하나의 명령어가 많은 작업을 의미할 수 있도록 복잡하게 구성될 필요가 있었으며, 이런 요구에 맞는 CISC(Complex Instruction Set Computing)가 출현하게 된다.

CISC의 쇠퇴와 RISC의 출현

IPC가 1이 한계라면 컴퓨터의 성능을 끌어올리는 방법은 clock 속도를 올려 주어진 시간동안 작업량을 늘리는 것이 유일한 해답이다. 이를 위해서는 CPU의 구조를 단순화해야 하며, 이런 요구를 만족시키는 RISC(Reduced Instruction Set Computing)가 1980년대 초반에 탄생한다. 하지만 단순한 여러 개의 명령어를 조합해야 CISC의 한 명령어와 동일한 작업을 수행할 수 있었던 RISC의 탄생은 컴퓨터 당 메모리 용량의 증가를 요구하게 된다. 1980년대 중반에는 clock 속도를 더 높이기 위해 pipeline 기법이 널리 적용된다. 이렇듯 CPU는 속도를 올리는 방향으로 급속히 진화하게 되고, 메모리는 용량을 늘리는 방향으로 진화하게 된다.

SIMD의 출현

초기에 컴퓨터는 미사일의 예상궤도 계산이나 암호화 등 인간이 수행할 수학적 계산을 돕는데 주로 사용되었다. 따라서 연산 명령어는 단순하나 다양한 데이터를 반복적으로 처리하는 패턴이 다수였다. 이에 적합하게 개선된 CPU가 SIMD(Single Instruction Multiple Data)이다. 즉, 한번에 하나의 명령어만을 수행하되 여러 개의 데이터를 같은 방식으로 동시에 처리하는 것이다. SIMD는 Cray의 vector 컴퓨터에서 꽃을 피워 1970, 80년대를 풍미하게 된다. 현대에는 SIMD가 독립 CPU 형태로 살아남지는 못했으나, 연산 가속기(HW Accelerator)의 형태로서 발전되어 CPU의 중요한 요소로 남게된다. Video codec 엔진, GPU 등이 SIMD 아키텍처의 좋은 예제들이다.

SIMD의 쇠퇴와 ILP 시대 도래

SIMD와 같은 구조는 많은 양의 데이터가 같은 방식으로 독립적으로 처리되는 scientific application에서는 유리하지만, 우리가 일상생활에서 사용하는 컴퓨터의 업무에서는 장점이 적용되지 않는 경우가 다반사이다. 일반적인 현대 컴퓨터 작업의 특징은 데이터간의 의존성, 명령어 간의 의존성, 입력 데이터의 의존성, 조건명령에 대한 의존성이 높아서, 데이터 처리 독립성이 매우 높아야만 나타나는 SIMD의 장점을 살릴 수 없다. 이런 특성의 업무들을 모아서 integer application이라고 표현한다. 컴퓨터의 보급이 늘어나고 보다 다양한 영역에 적용되면서 integer application 성능이 중요한 지표로 떠오르게 된다. 이에 SIMD는 독립 CPU로써 설 자리를 서서히 잃게 된다. Integer application의 성능을 올리기 위해서는 겉으로는 보이지 않으나, 프로그램 안에 녹아 들어있는 독립적인 명령어와 데이터만을 축출한 후 이들만을 선택적으로(프로그램 순서에 상관없이 미리 혹은 동시에) 수행할 수 있는 기술이 필요하다. 이 기술을 Instruction-Level Parallelism(IPL)라고 부르며, 이는 하드웨어 또는 소프트웨어로 구현할 수 있다. 하드웨어로 구현한 대표적인 첫 제품이 1996년에 발표된 DEC의 Alpha 21264이며, 소프트웨어로 구현된 대표적인 첫 제품이 2001년에 발표된 Intel의 Itanium이다.

하드웨어 접근의 ILP 발굴

이전까지 CPU는 3가지 기본동작(Fetch, Execution, Commit)을 프로그램의 순서대로 수행하는 in-order machine이었다. 이 조건 하에서 컴퓨터의 성능을 늘리기 위해 여러 개의 ALU를 하나의 CPU 안에 내장하는 방식으로 컴퓨터는 진화된다. 하지만 문제는 여러 개의 ALU를 동시에 사용할 명령어 수가 충분치 않았다는 것이다. 그 이유는 명령어는 항상 프로그램 순서대로 수행되는 것을 기대했기 때문이다. 이 벽을 허문 것이 Out-of-Order(OoO) execution machine이다. OoO는 fetch와 commit은 프로그램 순서로 하되, execution만은 하드웨어가 알아서 순서를 바꿔서 성능이 최대화되는 방향으로 수행할 수 있는 구조이다. 이를 가장 완벽히 개발한 첫 제품이 DEC의 Alpha 21264(1996년)이었으며, 향후 CPU 구조의 정석으로 자리잡게 된다. 일단, 명령어를 OoO로 수행할 수 있게 되면서 많은 수의 ALU의 내재화와 pipeline 폭의 증가를 통해 ILP의 발굴과 CPU의 성능 증가가 빠른 속도로 이루어진다. 이러한 구조의 CPU를 Superscalar라고 부른다. 현대 CPU는 4 ~ 20개의 ALU와 2 ~ 8폭의 pipeline을 가지고있다.

소프트웨어 접근의 ILP 발굴

하드웨어 접근의 ILP는 ILP 발굴의 모든 부담을 하드웨어에서 가져가는 것이다. 결과적으로 CPU의 구조는 이전에 비해 월등히 복잡해졌으며, 이는 동작 주파수 증가에 커다란 장애물로 작용하고 CPU 개발비용의 증가로 이어진다. 이와는 반대로, ILP 발굴비용과 복잡성을 소프트웨어(Compiler)로 옮기자는 흐름이 생긴다. 하드웨어는 과거처럼 in-order로 단순하게 만들어 비용을 낮추고 동작속도는 향상시키며 ILP 발굴은 소프트웨어가 담당하면, 낮은 가격과 높은 성능이라는 두 마리 토끼를 동시에 잡는다는 이유였다. 이의 첫 번째 제품이 2001년에 출시된 Intel의 Itanium이었다. 하지만, Itanium은 처참한 재앙으로 끝나게되며, 같은 유형의 제품이었던 IBM의 Power6(2006년) 역시 기대에 못 미치는 성능으로 회사에 타격을 주게된다. 이유는 간단했다. 일반적인 integer application들은 compile 시간에 발굴할 수 있는 ILP의 양이 극도로 제한적이기 때문에 compiler를 통한 ILP 발굴은 현실적으로 실현이 매우 어렵다. 이러한 사실은 Itanium 개발 초기부터 많은 아키텍처 전문가들에 의해 지적되었던 점으로서, 많은 전문가들은 Itanium의 실패를 예측하고 있었다.

Multi-core Processor의 등장

소프트웨어적인 접근이 실패한 ILP 발굴은 고가의 비용이 수반되는 하드웨어적 접근만이 남아 있는 듯 보였다. 그러나 하드웨어적 접근은 시간이 지날수록 투자비용 대비 성능이득은 줄어만 갔다. 하나의 반도체 chip에 집적할 수 있는 트랜지스터의 숫자는 18개월마다 2배씩 증가하는데 반해(무어의 법칙), 이를 성능으로 이어줄 기술은 고갈되어 갔다. 즉 단독 프로그램에 대한 실행속도의 향상은 포화상태(saturation point)에 가까워가고 있었다. 이를 돌파하기 위해 나타난 것이 Multi-core processor이다. 하나의 프로그램의 속도를 빠르게하기 어렵다면, 하나의 CPU에 core processor를 두 개 이상 넣어 core 숫자만큼의 프로그램들을 동시에 수행하여 컴퓨터의 전체적인 성능을 늘리자는 취지이다. 앞에서 언급했듯, 하나의 chip에 들어가는 트랜지스터의 수가 폭발적으로 증가했기에 가능한 디자인 방식이었다. 이 접근은 2000년대에 들어서면서 Intel과 AMD에 의해 널리 받아들여진다. 요즘은 스마트폰에 사용되는 CPU조차 8개의 core를 가지고 있는 것을 볼 수 있다.

CPU 구조의 현재 진행형

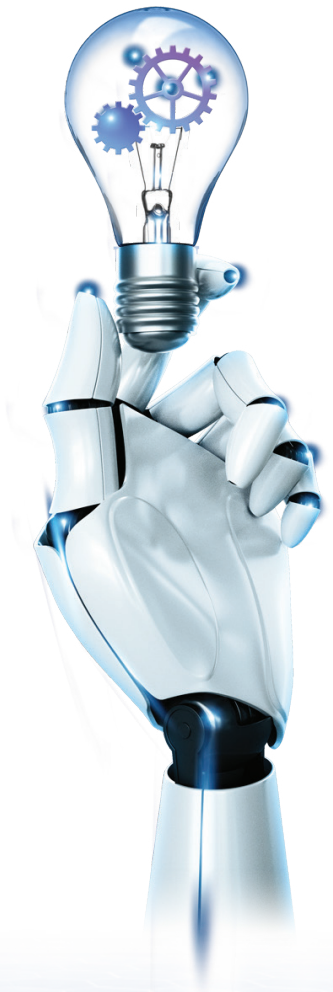
지금까지 소개한 CPU 구조들은 모두 Von Neumann 구조를 따르고 있다. CPU는 속도를 증가시키는 방향으로 진화했고, 메모리는 용량을 늘리는 방향으로 증가했다. 1970년대까지는 CPU와 메모리의 데이터 이동속도는 비슷했으나 2016년 오늘은 수배에서 수십배의 차이가 나고 있다. 이를 memory wall이라고 하며, Von Neumann 구조의 한계이다. 암달의 법칙에 따르면, 전체 컴퓨터에서 CPU가 차지하는 성능비중이 90%이고 메모리가 차지하는 성능비중이 10%라고 한다면, CPU의 속도향상이 천만배나 그 이상이 되어도 전체 성능향상은 10배를 넘기지 못한다. 사실이 이럼에도 불구하고 지금까지 세계의 반도체와 IT 업계는 CPU의 속도에만 집중해왔으며, 메모리의 성능향상은 상대적으로 뒷전이었다. CPU의 성능향상이 포화상태에 빠지는 지금, 우리는 메모리의 성능향상에 보다 큰 관심을 기울여야 할 것이다.

인공지능의 등장

알파고의 등장은 한국사회와 세계 IT 업계에 엄청난 충격을 안겨주었다. 구글, IBM과 같은 세계의 IT 리더 기업들은 인공지능이 IT의 미래라고 단정지으며, 무서운 속도로 투자와 개발을 진행 중이다. 인공지능을 구현하는 방법은 ANN(Artificial Neural Network)과 SNN(Synapse Neural Network)으로 나누어진다. ANN은 전형적인 Von Neumann 구조의 컴퓨터를 이용하는 반면, SNN은 Von Neumann 구조를 따르지 않는 PIM(Processing In Memory) 구조로써 메모리가 중심인 구조이다. ANN은 Backpropagation이라는 기술이 소개된 시점을 기준으로 한다해도 40년 가까운 긴 역사를 가지고 있으며, 구글의 알파고, IBM의 왓슨과 같은 유명한 인공지능 기계들도 ANN의 구조에 기반을 두고 개발되고 있다. 이에 비해 SNN의 역사는 미미하다. 하지만 SNN의 미래가 밝은 이유는 같은 양의 일을 수행하는데 소모되는 에너지가 ANN대비 1/20,000 밖에 불과하지 않다는데 있다. 그러면 SNN이 기존 컴퓨터 구조에 대비해 낮은 에너지를 소비하는 이유는 어디에 있을까? 그것은 인공지능의 메모리 접근 요구량이 기존 프로그램들 대비 폭발적으로 많다는데 있다. 메모리에서 데이터를 CPU로 이동하여 연산하고 메모리로 돌려보내는 기존 구조에서는, 인공지능에 필요한 데이터 이동에 소비되는 에너지가 정작 데이터 연산에 소비되는 에너지 대비 수십배에서 수천배에 달한다. 만약 데이터가 존재하는 곳에서 바로 연산을 처리한다면 데이터 이동에 소비되는 에너지를 대부분 절약하게 되며, 전체 에너지 소비를 획기적으로 줄일 수 있는 것이다. 컴퓨터 구조에서 메모리의 성능과 구조의 혁신적인 변화와 이를 위한 연구가 절실하며, 더 이상 미룰 수 없다는 것을 다시 한번 지적하는 좋은 예이다.

어떻게 준비할 것인가

과거 CPU 역사에서 보듯 CPU와 컴퓨터 구조의 발전은 그 시대의 문제점을 그 시대의 기술상황에 맞는 해답을 찾는 방향으로 이루어져 왔다. 최근 컴퓨터 아키텍처의 발전 방향은 짙은 안개에 가려져 있는 듯 하다. 무어의 법칙은 끝났다는 뉴스가 종종 나오며, Von Neumann 구조 역시 그 지휘가 흔들리고 있다. 이러한 현상들을 통해 컴퓨터에서 차지하는 CPU와 메모리의 역학관계가 급변할 것임을 짐작할 수 있다. AI의 화려한 등장은 메모리의 중요성을 한층 더 높여놓았다. 우리나라는 메모리반도체 산업에 기반을 두며 IT 강국으로 발전해왔다. 하지만 앞으로 우리가 지나가야 할 길은 우리가 오늘까지 지나온 길과는 사뭇 다르다. 표준화 기구에 의해 표준화가 끝나면 주어진 스펙에 따라 메모리 제품을 빨리 만들어내는 업체가 돈을 벌고 업계를 리드할 수 있는 것이 현재의 메모리 시장이다. 이런 메모리 시장은 앞으로도 존재할 것이다. 이와는 별도로, 컴퓨터 시스템 전체를 이해하고 이에 대한 요구사항을 메모리에 잘 녹아들게 만드는 메모리 솔루션 시장이 차세대 황금알로 떠오를 것으로 예상한다. 후자의 시장을 잘 준비하기 위해서는 과거 대비 다양한 전문가들이 메모리반도체 업계에 뛰어들어야 한다. 기존 메모리 업계가 공정 중심이었고 인력의 비중 역시 공정 전문가가 핵심이었다면, 이제는 Compiler, OS, Middleware, Application들을 잘 이해하는 소프트웨어 전문가, 컴퓨터 아키텍처 전문가, 회로설계 전문가의 참여가 필수적이다. 메모리 업계와 연관이 적을 것처럼 보이는 이런 전문가들을 메모리반도체 업계로 유도하기 위해서는 중장기적 계획과 많은 인내가 필요하다. 학교와 기업간의 산학연계, 주요 핵심 과제화를 통한 정부의 정책적 지원 등 전방위적 노력을 기울여야 할 때이다. 여러 IT 기술들을 한 군데 담으려는 융합공학 대학원들이 국내 대학들에 만들어지는 것은 긍정적인 현상으로 보인다. 시대의 변화는 위기이며 동시에 좋은 기회이다. 역사의 교훈을 바탕으로 미래를 지금 준비한다면 우리는 IT 강국으로 또 한번 도약할 수 있을 것이다.



Multiprocessor System-on-Chip (MPSoC)의 태스크 매핑 기술 동향

윤용희 석박사 통합과정, 김영환 교수 (POSTECH)

서론

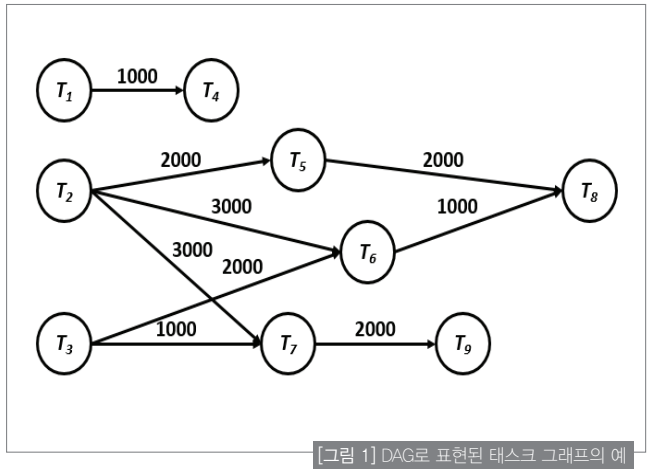
임베디드 시스템은 특정 기능을 수행하는 컴퓨터 시스템을 가리킨다. 임베디드 시스템의 용도는 매우 다양하여 현재 차, 비행기, 가전제품, 디지털카메라, 디지털 TV 등 수많은 종류의 전자장치에 내장되어 사용되고 있다.

임베디드 시스템은 하드웨어와 소프트웨어로 구성되며 하드웨어는 processor core, memory, IP module로 구성된다. 전통적인 임베디드 시스템은 대부분 제한된 기능만을 수행했으며, 따라서 하드웨어도 단순한 구조를 가지고 있었다. 하지만 최근 임베디드 시스템들이 보다 다양한 기능을 수행함에 따라 하드웨어 구조가 복잡해지고 있을 뿐 아니라 성능에 대한 요구가 증가하고 있으며, 전력소모나 발열 등에 대한 제한조건도 증가하고 있는 실정이다. 따라서 성능을 높이면서도 전력소모나 발열 등에 대한 제한조건을 만족시키기 위해 싱글 코어를 사용하던 기존의 패러다임에서 벗어나 하나의 칩에 여러 개의 코어를 집적시키는 multiprocessor system-on-chip(MPSoC)을 이용한 임베디드 시스템을 사용하는 것이 보편화되고 있다. 이러한 MPSoC 기반 시스템들의 성능 및 전력소모는 시스템 내의 하드웨어 자원을 얼마나 효율적으로 사용하는가에 따라 큰 차이를 보일 수 있다. 따라서 MPSoC 기반 임베디드 시스템의 성능 향상 및 소비전력 감소 등의 제한조건을 만족시키기 위해 태스크 매핑에 대한 연구가 임베디드 시스템 설계의 핵심적인 문제 중 하나로 자리잡게 되었다 [1].

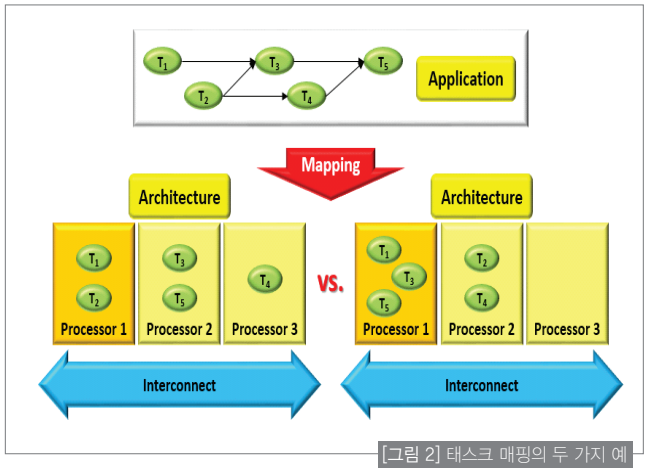
태스크 매핑은 애플리케이션을 구성하는 요소인 태스크들을 하드웨어 자원에 할당하고 스케줄링 하는 것을 말하며, NP-완전(NP-Complete) 문제로 알려져 있다 [2]. 태스크 매핑의 여러 제한조건을 만족시키는 최적해를 찾는 것은 너무 높은 복잡도가 요구되며, 따라서 다양한 휴리스틱 방법을 이용한 태스크 매핑 연구들이 진행되고 있다. 본 고에서는 MPSoC에서의 태스크 매핑에 대해 간략히 소개하고, 이에 대한 기존 및 최근 연구동향을 살펴보고자 한다.

태스크 매핑의 배경 이론

MPSoC의 태스크 매핑을 수행하기 위해 일반적으로 타겟 애플리케이션을 태스크 간의 병렬성, 의존성 및 기타 데이터를 나타낼 수 있는 directed acyclic graph(DAG) 형태의 태스크 그래프로 나타낸다. 그림 1은 태스크 그래프의 한 예로서, 9개의 태스크를 가진 애플리케이션을 DAG로 표현한 것을 보여주고 있다. 여기서 각 노드들은 코어들이 수행해야 할 태스크들을 나타내며, 각 노드는 해당 태스크의 workload에 대한 정보를 가지고 있다. 각 에지의 방향은 태스크들 사이의 의존성을 나타내며, 태스크들 간에 전달되어야 할 데이터 양에 대한 정보를 가지고 있다. 예를 들어, 그림 1의 애플리케이션 모델에서 태스크 4(T4)는 태스크 1(T1)이 선행되어야 실행될 수 있으며, 태스크 1(T1)에서 태스크 4(T4)로 1000만큼의 데이터 전송이 이루어져야 함을 알 수 있다.



[그림 1] DAG로 표현된 태스크 그래프의 예



[그림 2] 태스크 매핑의 두 가지 예

전술한 바와 같이 태스크 그래프로 나타나는 애플리케이션을 MPSoC 하드웨어 아키텍처에 할당하고 스케줄링 하는 것을 태스크 매핑이라고 하며, 태스크 매핑 알고리즘은 설계목적에 부합하는 태스크 매핑의 최적해를 찾는 알고리즘을 말한다. 그림 2는 태스크 매핑의 예를 보여주는 것으로, 태스크 레벨로 모델링된 애플리케이션을 주어진 하드웨어에 매핑한 두 가지 예를 보여주고 있다. 두 매핑은 서로 다르게 시스템 자원을 할당받아 태스크들을 수행한다. 다섯 개의 태스크를 수행하기 위해 왼쪽의 매핑은 세 개의 프로세서를 사용하고 오른쪽의 매핑은 두 개의 프로세서를 사용한다. 서로 다른 매핑의 결과는 시스템의 성능, 에너지 소모, 발열 등에서 큰 차이를 가져올 수 있기 때문에 목적에 맞는 매핑을 생성하는 태스크 매핑 알고리즘을 개발하는 것은 매우 중요한 연구주제이다.

태스크 매핑에 대한 연구들

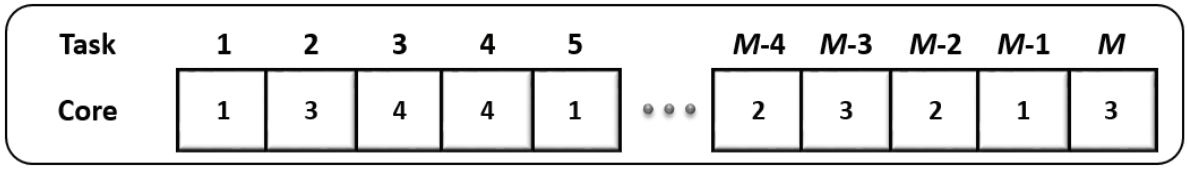
MPSoC에서 태스크 매핑은 타겟 시스템의 아키텍처 종류(homogeneous MPSoC 시스템 및 heterogeneous MPSoC 시스템), 매핑의 목적(에너지 감소, 성능 향상, 발열 감소), workload 형태의 종류(static workload 및 dynamic workload) 등 다양한 기준으로 분류되어 연구가 진행되어 왔다. 일반적으로 태스크 매핑은 workload의 형태에 따라 최적화 수행시점이 정해지고, 이를 기준으로 design-time 태스크 매핑과 run-time 태스크 매핑으로 분류한다. Design-time 태스크 매핑의 경우, 애플리케이션의 모든 태스크 관련 정보를 설계단계에서 완벽하게 예측하는 것이 가능한 임베디드 시스템의 설계에 주로 사용된다. 예를 들어, 멀티미디어 애플리케이션과 같이 동일한 일을 주기적으로 처리하는 임베디드 시스템에는 design-time 태스크 매핑 알고리즘을 이용하여 주어진 태스크들의 매핑을 설계단계에서 결정한다. 반면, run-time 태스크 매핑의 경우 애플리케이션의 태스크들을 설계단계에서 예측이 불가능한 임베디드 시스템의 run-time 관리에 주로 사용된다. 즉, 스마트폰 등과 같이 run-time에 일어나는 동작이 예측 불가능한 임베디드 시스템에는 run-time 태스크 매핑을 사용하여 태스크들의 매핑을 동작 중에 결정한다.

1. Design-time 태스크 매핑

Design-time 태스크 매핑은 시스템의 설계단계에서 수행하는 태스크 매핑을 말하며, 오프라인(offline) 태스크 매핑 또는 정적(static) 태스크 매핑이라고도 한다. 이 방식은 애플리케이션의 모든 태스크 관련 정보를 시스템 설계단계에서 이용 가능한 경우 사용된다. 이러한 design-time 태스크 매핑의 경우 애플리케이션이 동작 전에 결정되어 있기 때문에 연산량이 많더라도 최적 또는 최적에 가까운 매핑 결과를 찾아내는 알고리즘에 대한 연구들이 주로 진행되어 왔다. 태스크 매핑 문제의 크기가 작은 경우에는 최적의 태스크 매핑 결과를 찾아내는데 exact 알고리즘들이 사용되기도 하지만, 크기가 큰 태스크 매핑 문제는 유전 알고리즘(genetic algorithm) [3], 입자 집단 최적화(particle swarm optimization) [4], 개미집단최적화(ant colony optimization) [5]와 같은 휴리스틱 알고리즘들을 통해 최적에 가까운(near-optimal) 매핑 결과를 찾는 것이 일반적이다.

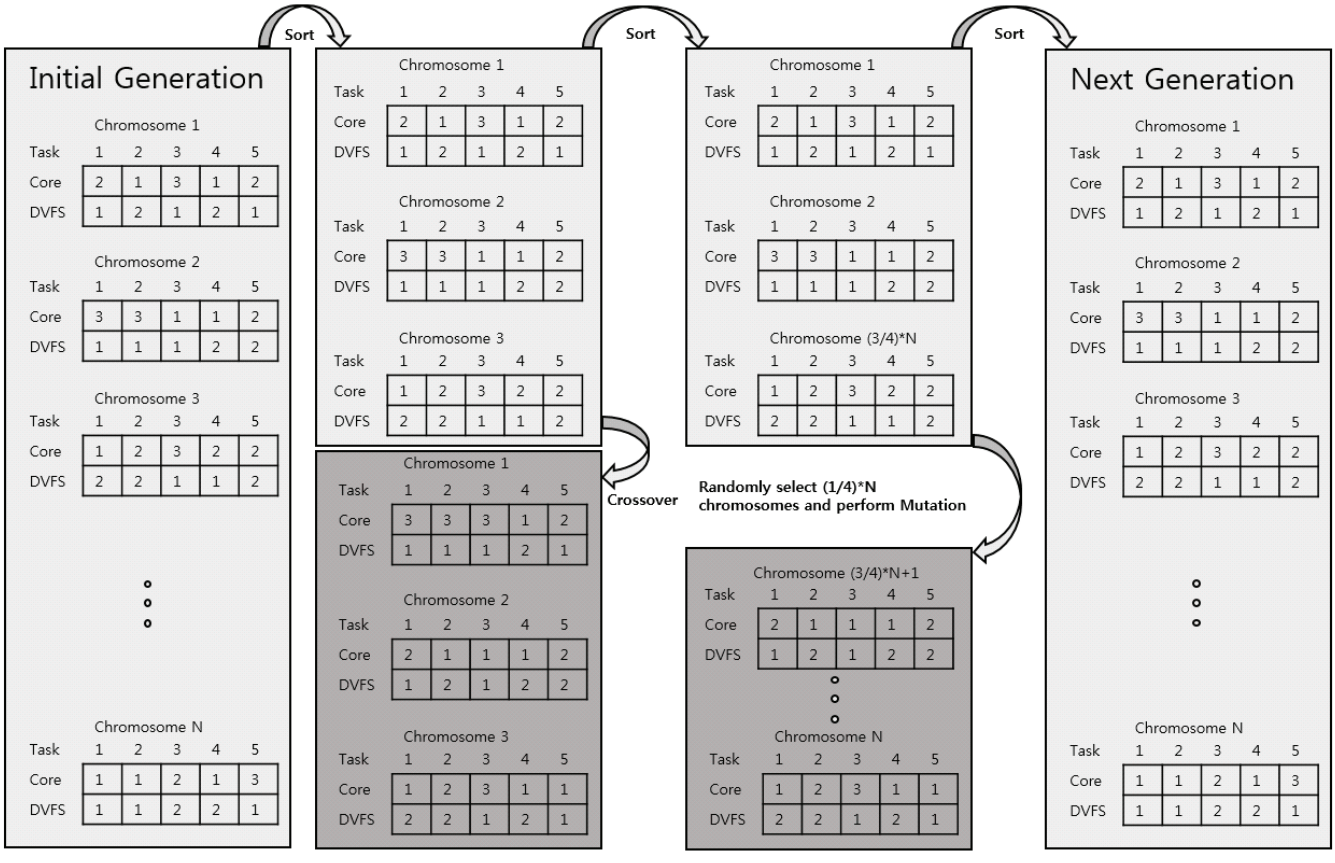
초기의 design-time 태스크 매핑 최적화 연구들은 주로 시스템의 동작속도 향상에 초점을 두고 애플리케이션의 수행시간을 최소화하는 방향으로 진행되었다. 담금질 기법(simulated annealing) [6]이나 타부 탐색(tabu search) [7], 입자 집단 최적화 [4] 등의 다양한 종류의 휴리스틱 알고리즘들을 사용하여 시스템의 성능향상에 중점을 둔 태스크 매핑 연구들이 이에 해당한다. 최근에는 전원 공급량이 한정된 휴대용 임베디드 시스템의 사용이 보편화되면서 시스템의 동작속도 뿐 아니라 에너지 소모에 대한 중요도가 더욱 높아지게 되었다. 이에 따라 수행시간에 대한 조건을 만족시키면서 시스템의 에너지 소모량을 최소화하는 것을 목적으로 하는 태스크 매핑 방법에 대한 필요성이 대두되었다. 이에 대한 대표적 연구로서, [8]에서는 유전 알고리즘을 사용하여 주어진 수행시간에 대한 제약조건 하에서 에너지 소모량을 최소화하는 매핑 알고리즘을 제안했다. 이 연구에서는 기존 방법들과 달리 엘리트 솔루션들 간의 교배를 통해 다음 세대의 일부를 구성함으로써 최적에 가까운 매핑을 빠르게 찾아내었다. [9]의 연구에서는 주어진 수행시간에 대한 제한조건 하에서 asymmetric MPSoC의 에너지 소모량을 최소화하는 태스크 매핑 알고리즘을 제안했다. 이 연구에서는 유전 알고리즘의 각 솔루션들을 수행시간과 에너지 소모량을 기준으로 (1) 시간 제약조건을 만족시키고 에너지 소모량이 줄어든 클래스, (2) 시간 제약조건을 만족시켰지만 에너지 소모량은 늘어난 클래스, (3) 시간 제약조건은 만족시키지 못했지만 에너지 소모량이 줄어든 클래스 및 (4) 시간 제약조건도 만족시키지 못하고 에너지 소모량도 늘어난 클래스의 4가지 클래스로 분류하여 각 클래스에 적합한 진화방법을 적용하여 최적화를 진행했다. 이를 통해, 이 연구에서는 진화과정에서 솔루션들의 다양성은 유지하면서, 더 효과적으로 솔루션들을 진화시켜 기존의 방법들에 비해 더 최적화된 매핑 결과를 얻을 수 있었다. [5]에서는 개미 집단 최적화를 이용해 inter-processor communication volume을 줄이고, 태스크들의 slack을 고려한 적정 동작 전압을 결정하여 시스템의 수행시간에 대한 제약조건은 만족시키면서 에너지 소모량을 줄이는 매핑 알고리즘을 제안했다.

이처럼 design-time 태스크 매핑 연구분야에서는 다양한 종류의 휴리스틱 알고리즘들을 사용하여 최적해를 찾는 방법에 대한 연구가 활발히 진행되어 왔다. 이들 중 가장 널리 사용되는 휴리스틱 알고리즘은 유전 알고리즘으로, [10]에서는 11개의 휴리스틱 알고리즘 중 유전 알고리즘이 태스크 매핑에 가장 적합하다는 것을 그들간의 비교를 통해 보여주었다. 유전 알고리즘은 자연의 진화과정을 모사해 최적화 문제를 해결하는 기법으로 여러개의 염색체(chromosome)를 세대에 걸쳐 교차(crossover)와 변이(mutation) 등을 통해 목표한 방향으로 변형시켜 나간다. 태스크 매핑 문제에서는 이 각각의 염색체가 하나의 매핑 솔루션으로 다뤄진다. 그림 3과 같이 하나의 염색체는 애플리케이션의 태스크 개수인 M개의 유전자(gene)를 가지고 있다. 여기서 각 유전자는 자신이 매핑된 코어의 번호를 가지고 있으며, 이를 통해 하나의 염색체는 하나의 태스크 매핑 결과를 표현한다.



[그림 3] 태스크 매핑에서의 유전 알고리즘 염색체의 표현

유전 알고리즘을 이용한 또 다른 연구로서, [3]에서는 주어진 시간 제약조건 하에서 각 코어별 동적 전압 및 주파수 조정(dynamic voltage and frequency scaling)이 가능한 homogeneous MPSoC 시스템의 에너지 소모량을 최소화하는 태스크 매핑 알고리즘(GeneS)을 제안했다. 즉, 이 연구에서 태스크 매핑 결과는 태스크와 코어 간의 매핑, 그리고 각 태스크를 수행할 때의 동적 전압 및 주파수 조정 레벨을 포함한다. 그림 4는 GeneS 알고리즘의 진화과정을 도식화한 것으로, 기본적인 유전 알고리즘 염색체의 표현인 태스크와 코어 간의 매핑과 더불어 각 코어별 전압 및 주파수 레벨도 포함된 것을 알 수 있다.



[그림 4] GeneS에서 염색체들의 진화과정 [3]

2. Run-time 태스크 매핑

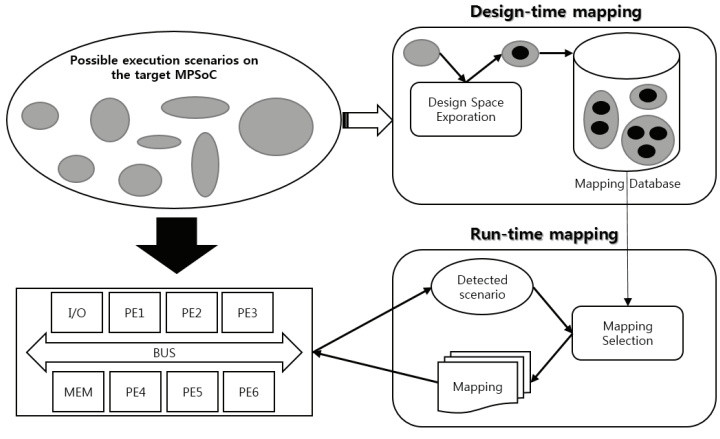
Run-time 태스크 매핑은 온라인(online) 태스크 매핑 또는 동적(dynamic) 태스크 매핑으로도 알려져 있으며, 시스템이 동작하는 중 태스크 매핑을 수행하는 것을 말한다. Design-time 태스크 매핑과 달리 run-time 태스크 매핑은 동적으로 변하는 workload를 다룰 수 있다는 강점을 가지지만, 시스템 동작에 영향을 주는 overhead도 존재한다. 시스템 동작 중에 태스크 매핑이 빠르게 이루어져야 하기 때문에 복잡한 연산을 필요로 하는 매핑 알고리즘을 사용할 수 없으므로, design-time 태스크 매핑에 비해 상대적으로 최적에 가까운 매핑 결과를 얻기는 힘들다. 최근 연구들 중에서는 run-time 매핑에 더 많은 시간을 투자하더라도 이를 통해 얻은 향상된 매핑 결과가 새로 추가되는 overhead를 충분히 보상할 수 있다는 주장도 있다 [11]. 하지만, 아직까지 대부분의 연구들은 최소한의 run-time overhead를 유지하며 향상된 매핑 결과를 찾아내는 것을 목표로 하고 있다.

Run-time 태스크 매핑의 연구분야에서는 주로 수행시간을 줄이는 것에 중점을 둔 연구들이 많이 수행되었다 [11-14]. 대표적으로 [11]에서는 communication requests와 Network-on-Chip(NoC) link에서의 load를 관측하여 communication overhead를 줄이는 방향으로 시스템의 수행시간을 줄이는 run-time 매핑 알고리즘을 제안하였고, [12]에서는 run-time에 workload의 변화를 관측하여 thread와 processor간의 매핑을 바꾸는 방식으로 애플리케이션의 수행에 필요한 시간을 줄일 수 있었다. 최근에는 run-time 태스크 매핑 연구분야에서도 design-time 태스크 매핑과 마찬가지로 수행시간을 줄이는 것에만 치중하지 않고 저전력을 요구하는 임베디드 시스템의 에너지 소모량을 줄이는 방향으로도 연구가 진행되고 있다. [15]의 연구에서는 run-time에 시스템의 idle processor들을 turn off 시키고, 각 태스크들의 time slack을 고려하여 동적 전압 조정을 적용해 에너지 소모량을 최소화하는 알고리즘을 제안했다. [16]에서는 사용자의 행동양식에 대한 정보를 이용하여 NoC에서의 run-time 태스크 매핑을 수행하는 방법을 제안했다. 이 방식은 NoC에서의 communication cost를 감소시킴으로써 시스템의 에너지 소모량을 줄일 수 있었다.

애플리케이션의 수행시간과 시스템의 에너지 소모량 외에도 시스템의 발열관리를 위한 run-time 태스크 매핑 연구가 이루어지고 있다. 예를 들어 [17]의 연구에서는 모바일 기기에 많이 사용되는 big.LITTLE 구조 [18]에서 run-time에 발생하는 발열문제를 태스크 이송(migration)을 통해 해결하는 방법을 제안했다. 이 연구에서는 big 코어에서 발열문제가 발생하면 big 코어의 전압과 주파수를 감소시키는 기존의 방법 대신, 해당 태스크들을 LITTLE 코어로 이송하는 방법을 사용했다. 이 방법은 big 코어의 온도를 빠르게 감소시킨 후 다시 big 코어에서 높은 주파수로 태스크들을 처리할 수 있게 함으로써 기존의 방법에 비해 10.6%의 성능 향상과 3.6%의 에너지 효율성을 얻을 수 있었다.

3. Hybrid 태스크 매핑

앞서 살펴본 것처럼 design-time 태스크 매핑의 경우 최적 또는 최적에 근접한 매핑 결과를 얻을 수 있다는 장점이 있지만, 동적인 workload는 다룰 수 없다는 단점이 있다. 반면에 run-time 태스크 매핑의 경우 동적인 workload를 다룰 수 있지만, 태스크 매핑을 run-time에 수행하기 때문에 run-time overhead가 발생하며 이를 최소화하기 위해 연산량이 작은 매핑 알고리즘을 사용하므로 매핑 결과의 질이 떨어질 수 있다는 단점이 있다. 최근 MPSoC에서의 태스크 매핑 연구에서 이러한 design-time 태스크 매핑과 run-time 태스크 매핑의 장점을 결합해 시스템에서 동적인 workload를 보다 효과적으로 다룰 수 있게 해주는 hybrid 방식의 태스크 매핑에 대한 연구가 등장했다. Hybrid 태스크 매핑 방식은 시스템에서 동작 가능한 모든 시나리오에 대해 design-time 태스크 매핑 알고리즘을 적용하여 최적 혹은 최적에 가까운 매핑 결과를 얻어낸다. 이 매핑 결과들을 시스템의 데이터베이스에 저장해 두었다가 run-time에 동작하는 관리자(run-time manager)를 통해 현재 돌아가고 있는 시나리오를 파악하여 최적의 매핑 결과를 적용한다. 그림 5는 이러한 hybrid 태스크 매핑의 과정을 도식화한 것으로, 앞서 설명한 것처럼 design-time과 run-time 태스크 매핑 기술이 결합되어 사용되는 것을 볼 수 있다 [19].



[그림 5] Hybird 태스크 매핑 방법 [19]

Hybrid 태스크 매핑은 그 목적에 따라 다양하게 사용될 수 있는데, [20]의 연구에서는 결함 감내 시스템(fault tolerant system)을 설계하기 위해 design-time에서 다양한 결함 시나리오에 대한 태스크 매핑 결과를 만들어놓고, run-time에서 결함이 발생할 경우 해당하는 결함 시나리오에 대한 태스크 매핑 결과를 사용할 수 있는 방법을 제안했다. 이 방식은 애플리케이션의 동작에 따라 생기는 event의 변화와 시스템의 결함에 의한 event의 변화를 run-time 관리자가 감지하여 design-time에 미리 최적화 시켜서 저장해 놓은 매핑 결과를 적용하므로 더 효과적인 run-time 동작을 보장할 뿐 아니라 시스템에 결함 발생 시 동작의 신뢰성을 높일 수 있었다. [21]의 연구에서는 처리율(throughput) 제약조건을 가진 애플리케이션을 효과적으로 MPSoC에 매핑할 수 있는 hybrid 태스크 매핑 알고리즘을 제안했다. 이 연구에서는 우선 design-time에서 다양한 시스템의 하드웨어 자원들의 조합에 대해 처리율(throughput)을 최적화하는 매핑을 생성하여 저장해두고, run-time에 현재 이용 가능한 시스템의 하드웨어 자원들을 고려하여 가장 적절한 매핑을 선택하는 방식을 사용한다. 이 방식을 통해 [20]의 연구는 H.263 디코더와 인코더 애플리케이션에서 기존의 run-time 태스크 매핑에 비해 각각 23%와 37%의 처리율 향상을 달성했다.

Hybrid 태스크 매핑은 기존의 design-time과 run-time 태스크 매핑에 비해 더욱 효과적인 시스템의 운영이 가능하지만 아직까지 많은 한계점들을 가지고 있다. 우선 hybrid 태스크 매핑은 시스템 설계 시 예측 가능한 run-time 시나리오만을 처리할 수 있기 때문에, run-time에 새로 추가된 애플리케이션에 의한 동적인 workload는 다룰 수 없다. 즉, 완전한 의미에서의 dynamic workload 처리는 보장하지 못한다. 또 시스템에서 동작할 애플리케이션의 종류가 많고 각 애플리케이션의 동작모드가 다양한 경우, 최적화를 수행할 시나리오의 수가 너무 많아지기 때문에 design-time 태스크 매핑에 비현실적으로 많은 시간을 소비할 수 있다. 예를 들어, 시스템에서 동시에 동작할 수 있는 애플리케이션의 개수가 10개이고 각 애플리케이션이 5개의 동작 모드를 갖는다면, 이들의 조합으로 구성된 전체 시나리오의 수는 약 6000만개에 이른다 [19]. 따라서 이를 실제 시스템에 바로 적용하기는 어렵다. 이러한 문제를 해결하기 위해 [2]에서는 design-time에서 애플리케이션 별로 그의 동작모드에 대해서만 최적화를 수행하여 매핑 결과를 생성해두고, run-time에서 현재 시나리오에서 동작하는 애플리케이션들과 그들의 동작모드를 감지하여 design-time에서 생성해놓은 태스크 매핑 결과들을 조합함으로써 새롭게 태스크 매핑 결과를 만들어내는 방법을 제안했다. 이 방법을 통해 기존의 run-time 태스크 매핑 대비 최대 105.9%의 성능향상과 23.5%의 에너지 효율성을 얻을 수 있었다.

결론

최근 고성능, 저전력 임베디드 시스템에 대한 요구를 만족시키기 위해 여러 개의 코어를 하나의 칩에 집적하는 MPSoC 기반 임베디드 시스템을 실생활에서 사용하는 것이 보편화되고 있다. 따라서, 이러한 MPSoC 시스템의 성능 향상 및 소비전력 감소 등을 위한 효과적 태스크 매핑을 찾을 수 있는 태스크 매핑 알고리즘의 개발이 중요한 연구주제로 대두되고 있다. MPSoC의 태스크 매핑은 크게 design-time 태스크 매핑과 run-time 태스크 매핑으로 나뉘어 연구되어 왔으며, 최근에는 이 둘의 장점을 합친 hybrid 태스크 매핑도 연구되고 있다. 본 고에서는 이와 관련된 주요 연구에 대해 간략히 소개했다.

현재 태스크 매핑의 연구는 주어진 시간이나 온도 등에 대한 제약조건 하에서 MPSoC의 성능, 전력소모 등을 최적화하는 것에 집중하고 있다. 현재 대부분의 임베디드 시스템은 제한된 숫자의 코어들을 집적한 quad-core 또는 octa-core MPSoC를 사용하고 있으나, 머지 않은 미래에 시스템 내의 코어 수가 수백개 또는 수천개까지 늘어날 것으로 전망되고 있다. 또 시스템 내의 코어들은 3D-NoC 등을 사용하여 3D 형태로 집적될 것으로 예상된다. 따라서 앞으로는 이러한 상황에 발맞춰 많은 수의 코어를 가지는 새로운 구조의 시스템들을 효과적으로 운용할 수 있는 태스크 매핑 알고리즘 연구가 이어질 것으로 기대된다.

| 저자 정보 |

윤 용 희
석박사 통합과정

- 소 속 POSTECH 전자전기공학과
- 주 연구분야 MPSoC power/thermal management
- E-mail ynpwh@postech.ac.kr
- Homepage http://soc.postech.ac.kr

| 저자 정보 |

김 영 환 교수

- 소 속 POSTECH 전자전기공학과
- 주 연구분야 MPSoC design methodology, DRAM analysis, image processing
- E-mail youngk@postech.ac.kr
- Homepage http://soc.postech.ac.kr/advisor/index.html

참고 문헌

[1] Singh, Amit Kumar, et al. "Mapping on multi/many-core systems: survey of current and emerging trends," Proceedings of the 50th Annual Design Automation Conference, ACM, 2013.

[2] Quan, Wei, and Andy D. Pimentel. "A hybrid task mapping algorithm for heterogeneous mpsocs," ACM Transactions on Embedded Computing Systems, vol. 14, no. 1, 2015.

[3] Wang, Yi, et al. "Overhead-aware energy optimization for real-time streaming applications on multiprocessor system-on-chip," ACM Transactions on Design Automation of Electronic Systems, vol. 16, no.2, 2011.

[4] Al Badawi, Ahmad, and Ali Shatnawi. "Static scheduling of directed acyclic data flow graphs onto multiprocessors using particle swarm optimization," Computers & Operations Research, vol. 40, no. 10, 2013.

[5] Kim, HyunJin, and Sungho Kang. "Communication-aware task scheduling and voltage selection for total energy minimization in a multiprocessor system using ant colony optimization," Information Sciences, vol.181, no.18, 2011.

[6] Orsila, Heikki, et al. "Automated memory-aware application distribution for multi-processor system-on-chips," Journal of Systems Architecture, vol. 53, no.11, 2007.

[7] Manolache, Sorin, Petru Eles, and Zebo Peng. "Task mapping and priority assignment for soft real-time applications under deadline miss ratio constraints," ACM Transactions on Embedded Computing Systems, vol. 7, no. 2, 2008.

[8] Yun, Yonghee, Eun Ju Hwang, and Young Hwan Kim. "Energy-efficient design time task mapping algorithm for MPSoC," 2015 International SoC Design Conference, 2015.

[9] Eun Ju Hwang. "Task Mapping for Energy Efficient MPSoC Design," Ph.D thesis, 2013

[10] T.D. Braun et al., "A Comparison of Eleven Static Heuristics for Mapping a Class of Independent Tasks onto Heterogeneous Distributed Computing Systems," J. Parallel and Distributed Computing, vol. 61, no. 6, pp. 810-837, 2001.

[11] Carvalho, Ewerson Luiz de Souza, Ney Laert Vilar Calazans, and Fernando Gehm Moraes. "Dynamic Task Mapping for MPSoCs," IEEE Design & Test, vol. 27, no. 5, 2010.

[12] Hong, Shengyan, et al. "Process variation aware thread mapping for chip multiprocessors," Proceedings of the Conference on Design, Automation and Test in Europe. European Design and Automation Association, 2009.

[13] Wang, Feng, et al. "Variation-aware task and communication mapping for mpsoc architecture," Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on, vol. 30, no. 2, 2011.

[14] Chen, Liang, Thomas Marconi, and Tulika Mitra. "Online scheduling for multi-core shared reconfigurable fabric," Proceedings of the Conference on Design, Automation and Test in Europe. EDA Consortium, 2012.

[15] Mehran, Armin, Ahmad Khademzadeh, and Samira Saeidi. "DSM: A Heuristic Dynamic Spiral Mapping algorithm for network on chip," IEICE Electronics Express, vol. 5, no.13, 2008.

[16] Chou, Chen-Ling, and Radu Marculescu. "User-aware dynamic task allocation in networks-on-chip," Design, Automation and Test in Europe, 2008.

[17] Kim, Young Geun, et al. "M-DTM: migration-based dynamic thermal management for heterogeneous mobile multi-core processors," Proceedings of the 2015 Design, Automation & Test in Europe Conference & Exhibition. EDA Consortium, 2015.

[18] P. Greenhalgh. "Big.LITTLE processing with ARM Cortex-A15 & Cortex-A7: Improving energy efficiency in high-performance mobile platforms,"http://www.arm.com/files/downloads/big LITTLE Final Final.pdf, 2011.

[19] Quan, Wei. "Scenario-based run-time adaptive Multi-Processor System-on-Chip," Ph.D thesis, 2015

[20] Schor, Lars, et al. "Scenario-based design flow for mapping streaming applications onto on-chip many-core systems," Proceedings of the 2012 international conference on Compilers, architectures and synthesis for embedded systems, 2012.

[21] Singh, Amit Kumar, Akash Kumar, and Thambipillai Srikanthan. "A hybrid strategy for mapping multiple throughput-constrained applications on MPSoCs," Proceedings of the 14th international conference on Compilers, architectures and synthesis for embedded systems, 2011.

하늘을 나는 로봇 : 드론

홍세화 이사 (바이로봇)

올해 초 한국 전통 방패연을 만드시는 리기태 선생님을 만날 기회가 있었다. 전통과 최신기술의 만남인 연과 드론을 주제로 한 기사 인터뷰를 위해 선생님께서 계신 북촌 한옥마을을 방문해 이런저런 이야기를 나눴다. 연초마다 연을 날리던 사람들이 드론을 날리기 시작했다고 하시며, 드론으로 인해 자칫 연에 대한 관심이 줄어들까 우려된다는 말씀이 기억에 남는다. 영화나 상상 속에서나 떠다니던 드론이 일상생활에서 심심치 않게 보이기 시작했다. 매년 1월 미국 라스베가스에서 열리는 세계최대 가전 박람회인 CES(Consumer Electronics Show)에서 드론 회사들로만 구성된 마켓플레이스(unmanned systems marketplace)가 생긴 것이 작년이다. CES의 운영조직인 CTA (Consumer Technology Association)는 최신 기술 또는 제품들을 조사하여 앞으로 20년 이상 세계의 먹거리가 될 만한 것들을 모아 새로운 마켓플레이스를 마련한다. 드론이 선택된 것은 드론이 한때 반짝하고 없어질 유행이 아니라는 것을 확실하게 보여준다. 드론은 비행로봇이다. 로봇 중에서도 움직이는 이동로봇(무인이동체라고도 한다.)으로 분류할 수 있고, 그 중에서도 하늘을 날 수 있는 로봇이다. 이동로봇의 범주 안에서 생각하면 현재 드론과 함께 이슈되고 있는 여러 첨단기술과의 연결고리를 쉽게 찾을 수 있을 것이다. 드론은 양력(揚力)을 발생시키는 날개의 형태에 따라 회전익 고정익(固定翼)형으로 구분한다. 일반적인 비행기 형태의 고정익 드론의 경우 이착륙시 활주로가 필요하다. 비행 중 양력을 유지하기 위해서는 일정속도 이상 지속적으로 움직여야하기 때문에 한 곳에 가만히 떠있는 것이 불가능하다. 하지만 바람을 잘 타면 작은 동력으로도 비행할 수 있어 장시간 비행시 유리하다. 회전익형은 일반 헬리콥터 모양을 떠올리면 된다. 날개가 동체와 독립적으로 회전하여 동체의 이동 없이도 양력이 발생한다. 따라서 이륙을 위해 별도의 활주로도 필요없고, 이륙 후 한 점에서 가만히 떠있을 수도 있다. 그러나 떠있기 위해서는 동체의 무게를 들 수 있을 만큼의 양력이 필요하고, 양력을 발생시키려면 지속적으로 날개(프로펠러)가 회전해야 하기 때문에 동력소모가 크다. 또한 회전하는 프로펠러로 인한 소음이 생각보다 크다. 드론을 처음 접하는 많은 고객들이 초소형 완구 드론 소리에도 놀라곤 한다.

고정익과 회전익 모두 개발 역사가 오래되었지만, 최근 드론이라는 키워드가 주목을 받게 된 것은 회전익의 한 종류인 쿼드콥터(quadcopter)

2015년 CES 현장 – 드론 전용 마켓플레이스
(출처:http://www.uasmagazine.com/articles/930/skysense-debuts-charging-pad-for-parrot-bebop-drone-at-ces-2015)

쿼드콥터 드론
(출처: 바이로봇 홈페이지)

때문이다. 프로펠러가 4개 달린 이 드론은 시스템 구정이 간단하고 조종이 비교적 쉽다. 프로펠러 개수에 따라 6개이면 헥사콥터(hexacopter), 8개이면 옥타콥터(octocopter)로 불리기도 하는데 가장 기본적인 형태는 쿼드콥터이다. 6년 전 즈음 현재 민수용 드론시장에서 1, 2위를 다투는 중국의 DJI사와 프랑스의 Parrot사가 일반 가전시장에 드론제품을 출시했다. 그 전까지 드론은 학교나 연구소, 특별한 취미가 있는 집단에서만 DIY(Do It Yourself) 형태로 조립되고 사용되었으나 이 제품들을 계기로 일반 가전매장에서 드론이 팔리기 시작한다.

다른 첨단기술들과 비슷하게 드론 역시 군사적 목적으로 개발되었다. 여전히 군사용 시장 크기가 민수용 시장을 훨씬 웃돌고 있지만, 성장률 면에서는 민수용이 급격히 올라가고 있다. 군수냐 민수냐 목표 시장에 구별없이 드론의 구성은 크게 4가지로 나눌 수 있다.

첫째, 내/외부의 환경 정보들을 분석해 이륙, 착륙, 이동 등 전반적인 비행과정을 제어하는 제어부. 우선 드론을 띄우기 위해서는 프로펠러를 원하는 속도로 돌려야한다. 각 프로펠러는 무게중심을 기준으로 방사형으로 배치되어있다. 각 프로펠러가 같은 크기의 힘을 내며 그 힘의 크기가 드론 동체 무게를 초과하게 되면 드론은 이륙한다. 회전속도를 더 높이면 더 빠른속도로 떠오른다. 이륙 후 프로펠러에서 발생하는 양력의 합을 동체의 무게와 같게 하면 드론은 그 자리에 가만히 떠있는다. 이론적으로는 동일한 모양의 프로펠러가 동일한 회전수로 돌면 동일한 양력이 발생한다. 그러나 현실세계에서는 각 프로펠러의 양력을 동일하게 맞추어 수직방향으로 이착륙하거나 하늘 위 한 점에 가만히 떠있게 하는 것이 매우 어렵다. 프로펠러의 회전수를 실시간으로 확인하며 각 회전수를 제어하기도 하고, 프로펠러에서 발생하는 힘에 따라 기울어지는 동체의 현재 자세를 측정하여 프로펠러를 제어하기도 한다. 드론 내부정보를 통해 드론을 안정적으로 띄웠다면 여기에 외부환경 정보들을 더해 스스로 비행하게 한다. 나의 현재 위치에서 목적지까지의 거리를 계산하여 최적의 비행경로를 설정해 비행할 수 있다. 또한 장애물을 만나면 어떻게 피할지를 결정하고 비행경로를 재설정한다. 상황을 판단하고 의사결정을 하는 부분이 드론의 제어부이다. 제어부에서 가장 중요한 것 중 하나가 처리속도이다. 비행제어에 사용할 수 있는 다양한 제어 알고리즘이 존재하지만, 이를 적용한 프로세서에서 충분한 속도가 나오지 않으면 사용이 불가능하다. 소프트웨어와 하드웨어 시스템의 조화가 매우 중요하다.

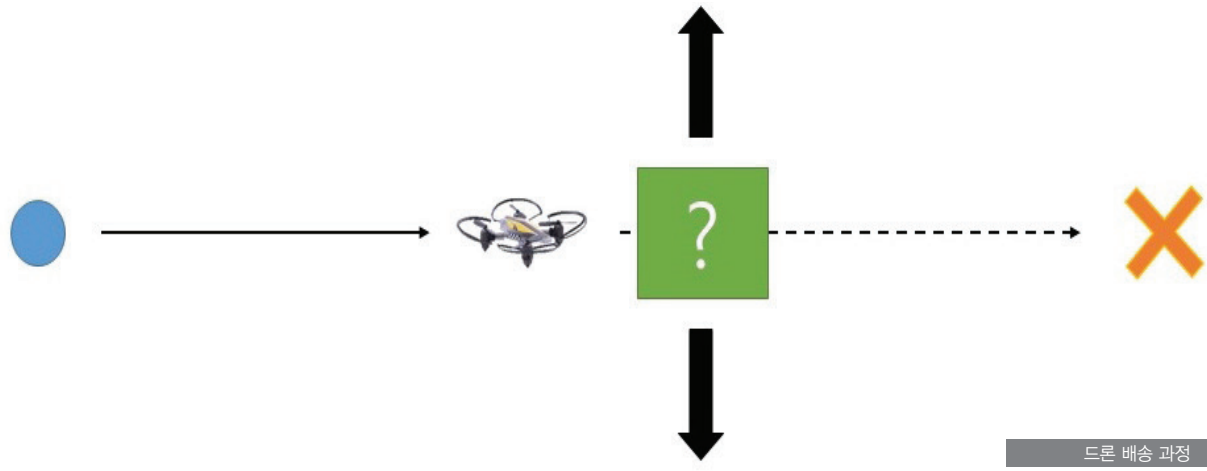
둘째, 제어부에서 필요한 각종 정보들을 수집하는 센서부. 수준 높은 비행제어를 위해서는 다양한 센서들이 필요하다. 순간의 기울어짐과 이동속도를 측정하는 자이로/가속도 센서, 비행고도를 측정할 수 있는 초음파와 기압센서, 지구 상의 절대위치를 측정할 수 있는 GPS, 주변 환경을 인식하여 다양한 정보를 수집할 수 있는 카메라 등 각종 센서들을 통해 다양한 데이터를 수집한다. 각각의 데이터에는 수집단계에서 발생한 많은 노이즈가 섞여있어 단독으로 사용하기가 매우 어렵다. 모든 데이터들이 상호보완 과정을 거치며 제어에 사용될 수 있는 의미있는 정보로 융합된다. 센서를 다양하게 사용할수록 드론은 더욱 똑똑해질 수 있지만, 센서의 크기나 무게가 비행성능에 직접적인 영향을 주기 때문에 센서의 적절한 선택이 중요하다.

셋째, 제어부에서 내려진 명령을 물리적으로 수행하는 구동부. 대부분의 소형 쿼드콥터는 전기신호로 움직이는 모터를 사용하여 프로펠러를 회전시킨다. 전원도 배터리를 사용한다. 전원은 모터의 용량에 따라 결정되며 모터의 용량은 프로펠러의 크기와 모양을

따른다. 초소형 완구 드론(회전익)의 경우 최대 비행시간이 10분을 넘지 못한다. 간혹 짧은 비행시간을 안타까워하며 배터리 용량을 키워달라는 소비자들의 요구가 있으나, 배터리 용량을 키우게 되면 배터리의 무게와 크기 역시 커져 드론의 전체 무게가 무거워진다. 결국 비행시간은 비슷하다. 구동부는 드론의 크기와 무게, 비행시간, 동체에 추가할 수 있는 무게(가반하중(可搬荷重), payload)를 결정하는 중요한 부분이다. 드론 비행규정의 경우 대부분이 동체의 크기와 무게를 기준으로 사용을 제한하고 있다. 사용 목적에 따른 드론크기, 비행시간, 추가로 장착하는 임무장비들을 잘 선택하여 구동부를 구성해야 한다.

넷째, 사용자와 드론의 인터페이스를 구성하는 조종부. 최근에는 다양한 인식기술들이 개발되어 조종부가 빠르게 진화하고 있다. 조종부의 시작은 우리가 쉽게 상상할 수 있는 RC 조종기의 형태이다. 큰 레버가 양쪽에 두 개 달려있어 드론의 상하좌우를 손가락으로 조종한다. 아주 기초적인 인터페이스다. 스마트 디바이스가 발전하면서 기계적 레버가 스크린상에 표현되기 시작했다. 좀 더 진화한 방식으로는 스크린에 지도가 보이고, 원하는 루트를 그리면 그것을 따라 드론이 스스로 비행한다. 손목에 차고있는 스마트시계의 움직임으로 드론을 조종하기도 하며, 음성명령으로 제어하기도 한다. 이러한 다양한 조종방법은 우선 드론의 안정적이고 안전한 비행이 보장되어야만 적용할 수 있다. 대부분의 사람들은 드론 자체의 기술보다는 드론으로 무엇을 할 수 있는지에 더 관심이 많다. 우리가 기대하는 드론 서비스와 현재 기술수준을 비교하기 위해 쇼핑한 물건을 드론으로 배달하는 과정을 상상해 보자. 모바일 쇼핑몰에서 원하는 물건을 구매한다(다양한 모바일 결재 서비스들이 나오면서 구매 결정에서 결재까지 몇 번의 클릭만으로 손쉽게 할 수 있다.), 원하는 목적지에서 가장 가까운 물류창고에 주문이 접수 되고 물류창고에 대기하고 있는 드론에 주문된 물건이 실린다(물류창고 안의 무수한 물건 중 주문상품을 구별하여 드론에 물건을 싣는 로봇까지 개발되고 있다.), 이미 저장되어 있는 지도정보를 기초로 목적지까지 날아가는 최적의 경로를 계산한 후 이륙한다. 드론은 현재의 배터리 잔량, 기체 상태 등을 모니터링하며 경로를 따라 안전하게 비행한다. 비행 중 다른 변수가 없다면 최적의 경로는 직선이 될 것이다. GPS 신호로 최종 목적지가 가까워 왔음을 인식하고, 바닥을 비추고 있는 카메라를 작동시켜 안전한 착륙지점을 찾는다. 배송물건을 목표지점에 두고 다시 이륙하여 물류창고로 복귀한다. 여기까지는 비행 중 특별한 변수가 없는 이상적인 상황이다. 그러나 우리가 살고 있는 현실세계는 더욱 복잡하다. 새로생긴 건물이 지도상에는 없을 수도 있다. 비행 중 갑작스러운 돌풍으로 비행경로에서 멀리 벗어날 수도 있고, 추운 날씨 탓에 배터리나 모터의 성능이 갑자기 저하될 수도 있다. 근처에서 비행하는 다른 드론과의 충돌위험이 존재하며(목적지가 유사할 경우 계산된 최적의 경로 또한 비슷할 수 있다.), 음식물 배달의 경우 냄새를 맡고 날아온 새들의 공격을 받을 수도 있다. 지도 상에 상세히 표시되지 않은 전신주나 전선, 나뭇가지에 걸릴 위험도 있다. 목표지점을 인식하고 착륙하는 도중 충성스러운 애완견의 습격을 받을 수도 있다. 우리가 기대하는 것은 이 모든 과정을 사람의 조종없이 드론 스스로 감당하는 것이다. 즉, 자율비행을 기본으로 하는 드론 서비스를 기대하고 있다. 하지만, 상용화에 성공한 민수용 드론 중에는 이런 갑작스러운 변수에 대처할 수 있는 수준의 제품이 아직 없다. 드론 서비스의 실현을 위해 발전시켜야 할 기술의 3가지 방향은 정보단말기화, 자율화, 네트워크화이다. 드론은 물리적인 공간을 이동하는 가장 효과적인 플랫폼이다. 이동을 위해서는 현재와 이동하게 될 공간에 대한 정보가 필요하다. 이 정보는 대부분이 비선형이며 시변적(time variant)일 것이다. 그러나 공간에 대한 지속적인 데이터 수집과 분석을 통해 시간에 따른 지역별 기후변화처럼 어느 정도의 패턴을 찾을

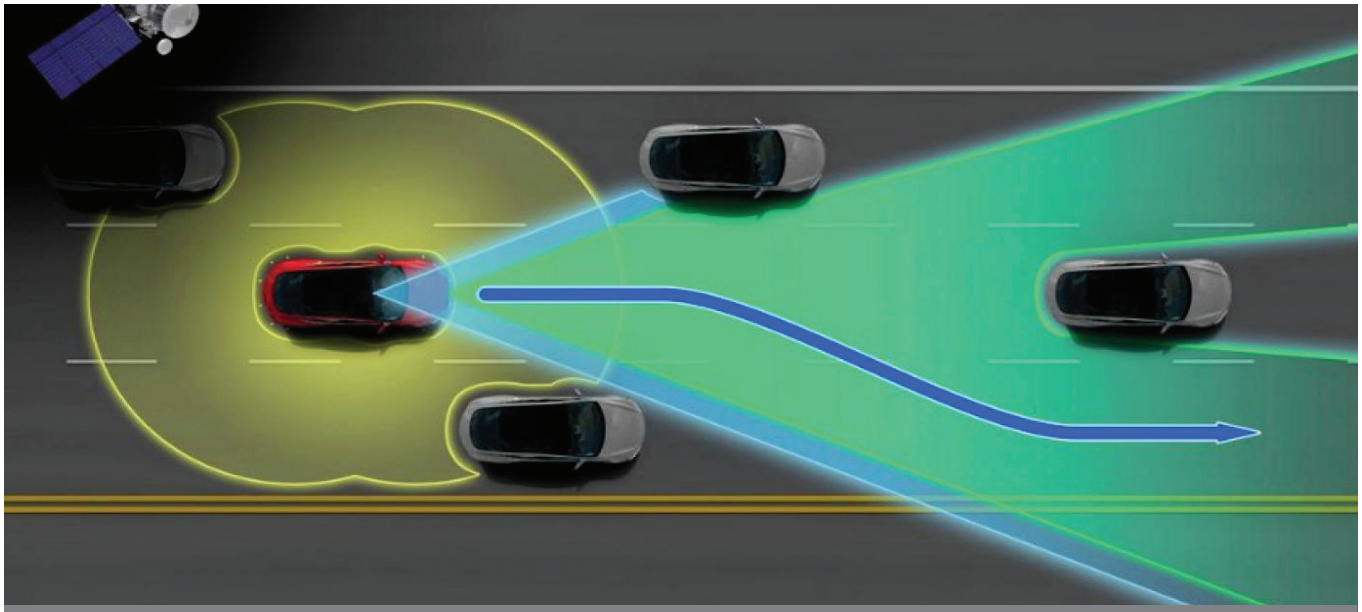
아마존에서 개발 중인 택배 드론
(출처:https://www.amazon.com/b?node=8037720011)



드론 배송 과정

수는 있을 것이다. 각종 센서가 장착된 드론이 정보단말기가 되어 이동하는 시간 내내 내, 외부의 다양한 정보를 수집하게 한다면, 현실세계의 복잡한 패턴분석에 도움이 될 것이다. 이렇게 모인 다양한 형태의 정보들을 우리는 빅데이터라 부른다. 이 대규모 데이터를 기반으로 각 상황별 판단기준이 만들어진다. 스스로 판단하여 움직이는 정도를 나타내는 자율성(自律性)은 드론 서비스에서 가장 중요한 요소이며, 자율성의 척도는 드론뿐 아니라, 모든 로봇 서비스에서도 중요하다. 자율성을 높이기 위해서는 최대한 많은 상황에 대한 판단기준이 있어야하고, 유사한 경험들을 분석하여 새로운 상황에 대한 판단기준을 스스로 만들어야 한다. 기존 자료들을 기초로 스스로 학습하여 새로운 방법을 찾아내는 최근의 인공지능 기술이 드론의 완전 자율화를 앞당기고 있다. 연초 구글에서 고정익 형태의 태양광 드론을 개발하여 시험비행을 한다는 내용이 언론에 보도됐다. 구글이 드론을 띄우는 이유는 전세계를 온라인으로 연결하기 위해서다. 인공위성처럼 드론이 일정 궤도를 비행하면서 인터넷망이 깔리지 않은 지역까지 서비스를 제공하겠다는 것이다. 항공촬영, 배송, 엔터테인먼트, 게임, 스포츠 등 전세계 다양한 분야에서 임무수행을 하고 있는 드론이 수집한 정보와 스스로 만들어낸 판단기준들이 온라인 네트워크 환경 속에서 실시간으로 공유되며 진정한 의미의 집단지성을 만들어갈 미래가 멀지 않았다.

구글의 태양광 드론
(출처:https://cdn2.vox-cdn.com/thumbor/CA1U4gMJ2xG2s-tSfPTKJdiVFxw=/0x0:1687x1125/1280x854/cdn0.vox-cdn.com/uploads/chorus_image/image/48681987/titan-aerospace-drone-google.0.0.jpeg)



테슬라의 자율주행 기능
(출처:https://i.imgur.com/6TcxiNL.jpg)

최근 테슬라 자동차의 사망사고 기사가 났다. 자동차의 자율 주행(Autopilot) 기능을 믿고 운전을 맡긴 운전자가 사고를 당한 것이다. 운전자의 기대는 자동차가 모든 상황을 분석하고 스스로 장애물을 피해 안전하게 목적지까지 운전해주는 것이었지만, 자동차의 인공지능은 운전자의 기대에 미치지 못했다. 지금까지 로봇은 제한된 공간에서 계산된 작업만 반복적으로 수행하는 산업용 로봇이 대부분을 차지했다. 돈을 벌 수 있는 서비스 로봇은 로봇 청소기뿐이라는 우스갯소리가 최근 몇 년 전까지만 해도 로봇 업계에 퍼져있었다. 그러나 스마트폰의 보급으로 성능 좋은 센서와 프로세서 등의 가격이 내려가면서 그동안 비싸게 구현되었던 로봇 기술들의 가격 또한 내려갔고, 사람들이 로봇에 거는 기대가 다시금 커지기 시작했다. 매장에서 상품설명을 하고 주문을 받는 인간모양의 휴머노이드 로봇, 버튼 하나면 멋진 항공영상을 촬영해주는 드론, 입으면 힘이 세지는 웨어러블 로봇 등 다양한 형태의 로봇들이 이미 일반인들도 쉽게 구매할 수 있을 정도의 가격으로 출시되고 있다. 가까운 미래, 드론을 포함하는 서비스 로봇은 우리가 사용하는 기계장치나 도구의 개념을 넘어 같은 생활공간에 공존하는 스스로 판단하고 행동할 수 있는 하나의 존재로 인식해야 할 것이다.

IDEC Newsletter | 통권 제231호

발행일 2016년 8월 31일 **발행인** 박인철 **편집인** 김태욱, 남병규 **제작** 과학문화사
기획 김하늘 **전화** 042) 350-8535 **팩스** 042) 350-8540 **홈페이지** http://www.idec.or.kr
E-mail kimsky1230@idec.or.kr **발행처** 반도체설계교육센터(IDEC)

반도체설계교육센터 사업은 산업통상자원부, 한국반도체산업협회, 반도체회사(삼성전자, SK하이닉스, 매그나칩반도체, 앰코테크놀로지코리아, 에이티세미콘)의 지원으로 수행되고 있습니다.