논문 2019-14-07

차세대 위성 프로세서 선정을 위한 성능 분석

(Performance Analysis of Processors for Next Generation Satellites)

유 범 수, 최 종 욱*, 정 재 엽, 김 선 욱
(Bum-Soo Yoo, Jong-Wook Choi, Jae-Yeop Jeong, Sun-Wook Kim)

Abstract : There are strict evaluation processes before using new processors to satellites. Engineers evaluate processors from various viewpoints including specification, development environment, and cost. From a viewpoint of computation power, manufacturers provide benchmark results with processors, and engineers decide which processors are adequate to their satellites by comparing the benchmark results with requirements of their satellites. However, the benchmark results depends on a test environment of manufacturers, and it is quite difficult to achieve similar performance in a target environment. Therefore, it is necessary to evaluate the processors in the target environment. This paper compares performance of a processor, AT697F/LEON2, in software testbed (STB) with three development boards of XC2V/LEON3, GR712RC/LEON3, and GR740/LEON4. Seven benchmark functions of Dhrystone, Stanford, Coremark, Whetstone, Flops, NBench, and MiBench are selected. Results are analyzed with hardware and software properties: hardware properties of core architecture, number of cores, cache, and memory; and software properties of build options and compilers. Based on the analysis, this paper describes a guideline for choosing processors for next generation satellites.

Keywords : AT697F, GR712RC, GR740, performance analysis, processor.

Ⅰ. 서 론

위성탑재컴퓨터의 프로세서를 선정 할 때, 성능, 설계의 적합성, 개발환경, 비용, 인터페이스, 단종의 위험성 등 다양한 관점에서 검증이 필요하다. 그 중 위성비행소프트웨어 개발 관점에서 가장 중요한 것은 성능이다. 따라서 프로세서 개발사들은 프로세서의 성능지표를 벤치마크 (Benchmark) 결과로 제공하고, 위성 개발자는 해당 벤치마크 결과를 참고하여 해당 프로세서가 자신들의 위성에 얼마나 적합한지 판단한다 [1, 2].

하지만 프로세서 개발사들이 제공하는 벤치마크 결과를 그대로 적용하기에는 한계가 있다. 벤치마크 결과는 하드웨어 및 소프트웨어 형상에 따라 크게 달라지기 때문에 서로 다른 환경에서의 벤치마크 결과로는 개발자가 원하는 환경에서의 성능을 예측하기 어렵다.

이 때문에 새로운 프로세서를 도입하기에 앞서, 개발자들은 자신들이 필요로 하는 관점에서 프로세

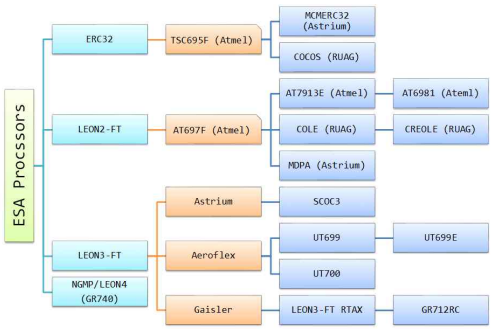


그림 1. 위성용 SPARC 프로세서의 종류
Fig. 1 SPARC processors for spacecraft

Corresponding Author (jwchoi@kari.re.kr)
Received: May 2 2018, Revised: June 25 2018, Accepted: Sep. 6 2018.
B.-S. Yoo, J.-W. Choi, J.-Y. Jeong, S.-W. Kim: Korea Aerospace Research Institute

ISSN : 1975-5066
http://dx.doi.org/10.14372/IEMEK.2019.14.1.51

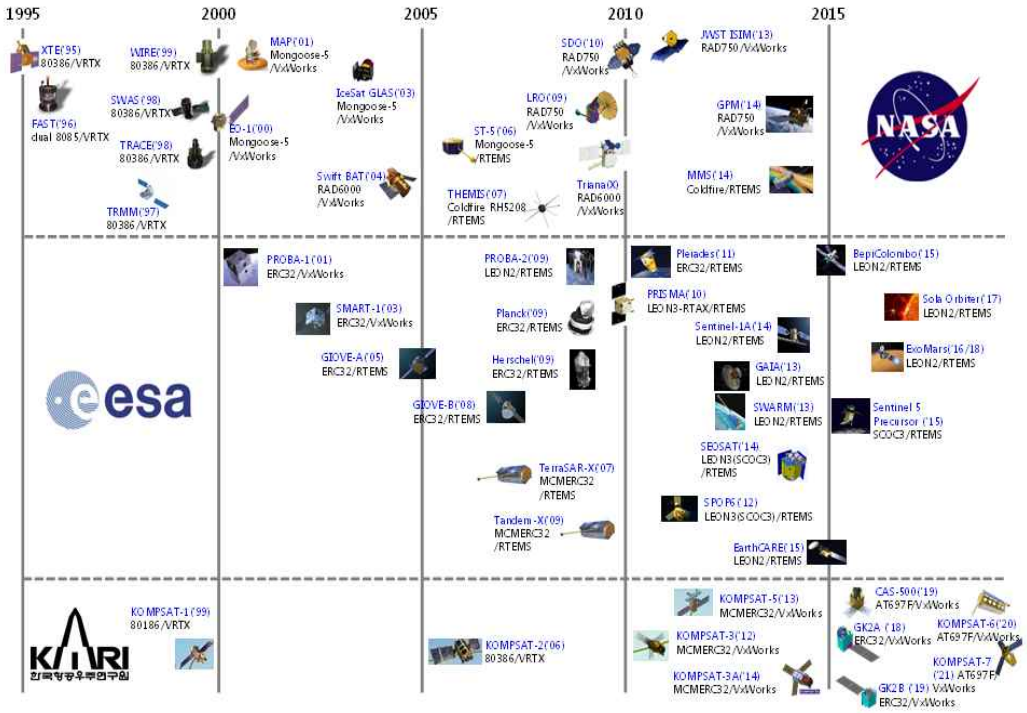


그림 2. 인공위성에 사용 중인 프로세서
Fig. 2 Processors for spacecrafts

서 성능에 대한 연구를 수행한다. 기존에 수행된 연구를 살펴보면 LEON3 프로세서를 적용함에 있어 소프트 프로세서와 하드 프로세서를 선택하기 위한 연구를 수행하거나, 캐시 (Cache)로 인해 성능이 어떻게 변하는지에 대한 연구를 수행하여 성능측면에서 프로세서를 비교하였다 [3, 4]. 이처럼 새로운 프로세서를 적용하기 위해서는 원하는 환경에 맞는 벤치마크 결과를 필요로 한다.

본 논문에서는 European Space Agency (ESA)에서 개발한 SPARC 계열의 LEON 2/3/4 위성 프로세서에 대한 성능을 다양한 하드웨어 및 소프트웨어 형상에서 벤치마크 프로그램을 이용하여 분석한다. 그림 1은 ESA의 SPARC 계열 프로세서를 보여준다. 이 중에서 LEON2 시스템은 AT697F 프로세서를 탑재한 Software TestBed (STB)를 이용하고, LEON3 시스템은 FPGA IP (Intellectual Property) 기반 LEON3 시스템인 XC2V 개발 보드 및 GR712RC 개발 보드를 이용한다. 그리고 LEON4 시스템은 GR740 개발 보드를 이용하여 성능을 분석한다.

성능측정을 위해 사용하는 벤치마크 프로그램은 Dhrystone, Coremark, Whetstone, Flops, Stanford, NBench, MiBench 총 7개로, 각 벤치마크 프로그램들의 특징을 고려하여 선별하였다. 측정결과를 하드웨어 및 소프트웨어 형상 특징에 따라 분석하며, 이를 통해 차세대 인공위성의 프로세서 선정을 위한 가이드라인을 제시한다.

논문의 구성은 다음과 같다. 2장에서는 현재 위성 탑재컴퓨터에서 사용되는 프로세서에 대해서 기술하며, 3장에서는 성능 분석을 위한 하드웨어 및 소프트웨어 형상, 그리고 사용한 벤치마크 프로그램에 대해 다룬다. 4장에서는 실험 결과 및 분석에 대해 다룬다. 마지막으로 5장은 결론으로 프로세서 선정을 위한 가이드라인을 제시한다.

II. 위성용 프로세서

1. 위성용 프로세서의 특징

인공위성 탑재컴퓨터에 사용되는 프로세서는 상용 프로세서와 달리 우주환경에서도 정상적으로 동작 할 수 있도록 외부 환경 변화에 강인하게 설계가 되어 있다. 위성용 프로세서는 래치업 (Latch-up),

표 1. 성능 측정을 위한 각 후보 시스템 형상
Table 1. System configuration for each candidate system

Factor	LEON2 AT697F	LEON3 XC2V	LEON3 GR712RC	LEON4 GR740
CPU type	ASIC	FPGA IP	ASIC	ASIC
Operating frequency	80MHz	40MHz	48/100MHz	250MHz
Core number	1 Core	1 Core	2 Cores (Only 1core used)	4 Cores (Only 1core used)
Floating point unit	MEIKO	GRFPU-lite	GRFPU	GRFPU
L1 instruction cache	32 KiB 4-way, 32 Bytes/line	8 KiB, 2-way, 32 Bytes/line	16 KiB 4-way 32 Bytes/line	16 KiB, 4-way 32 Bytes/line
L1 data cache	16 KiB 2-way 16 Bytes/line	8 KiB, 2-way, 32 Bytes/line	16 KiB 4-way 16 Bytes/line	16 KiB, 4-way 32 Bytes/line
Cache replacement algorithm	LRU	LRU/LRR/Random (LRU used)	LRU/LRR/Random (LRU used)	LRU/LRR/Random (LRU used)
L2 cache	-	-	-	disabled
SRAM	6MiB	-	8MiB	-
SDRAM	-	128Mi B	256MiB	128MiB

single event upset, single event transients, state machine control logic upset을 포함한 다양한 상황에 대응할 수 있을 뿐만 아니라, 위성이 운용되는 환경 조건도 만족해야 한다. 예를 들어 tolerate total dose의 경우, 저궤도 위성에서 사용하는 프로세서는 10-50kRad, 정지궤도 위성에서 사용하는 프로세서는 100~300kRad 요구 조건을 만족시켜야 한다. 자세한 요구 조건은 다음의 논문에 기술되어 있다 [5].

2. 위성용 프로세서 선정

그림 2는 대표적인 우주개발기관인 National Aeronautics and Space Administration (NASA), ESA, 그리고 한국에서 개발한 위성들의 탑재 프로세서를 연대순으로 나열한 것이다 [6]. NASA의 경우, 1990년도에는 인텔계열의 8085, 80186 및 80386 등의 프로세서가 사용되었으나, 2000년대에는 RAD750 및 RAD6000 계열의 프로세서를 사용하였다.

ESA에서는 초기 MIL-STD-1750 프로세서를 사용하였으나, 1990년대 이후 자체적으로 개발한 ERC32 프로세서를 2000년대 초중반까지 사용하였다. 이후 점점 요구하는 연산량이 커짐에 따라 차세대 프로세서인 LEON2/3 프로세서를 채택하여 현재까지 사용하고 있고, 동시에 멀티코어 (Multi core) 기반의 LEON4 프로세서를 개발하였다.

한국항공우주연구원 (항우연)에서 개발한 위성들을 살펴보면, 1999년, 2006년에 발사한 아리랑위성 1호/2호에서는 인텔계열의 80186 및 80386 프로세서를 사용하였다. 아리랑위성 3호, 5호 및 3A호에서는 수출 승인 문제로 더 이상 NASA 계열의 프로세서를 사용하지 못 하였으며, ESA 계열의 ERC32 프로세서를 확장한 MCMERC32 프로세서를 사용하였다.

위성에서 필요한 연산량이 증가함에 따라, 현재 항우연에서 개발 중인 아리랑위성 6호/7호, 차세대 중형위성, 그리고 시험용 달 궤도선 (Korean pathfinder lunar orbiter)에는 MCMERC32보다 뛰어난 성능의 LEON2-FT (AT697F) 프로세서가 탑재될 예정이다. 나아가 2020년 이후 새롭게 개발될 위성에서 LEON2-FT보다 뛰어난 멀티코어 프로세서를 사용하기 위한 연구가 필요하다.

3. 탑재 소프트웨어의 연속성

프로세서 선정에 있어서 고려해야 할 사항으로 탑재 소프트웨어의 연속성이 있다. 기존의 MCMERC32에서 사용하던 코드를 새로운 프로세서에서도 이용할 수 있으면 개발비용 및 기간을 크게 줄일 수 있다. MCMERC32와 LEON은 같은 ESA 계열의 프로세서로, GCC from VxWorks 와 RTEMS, 2개의 컴파일러가 MCMERC32는 물론 LEON 계열도 지원을 한다. 따라서 탑재소프트웨어의 연속성을 확보할 수 있다.

Ⅲ. 실험 형상 설정

1. 하드웨어 형상

프로세서 성능은 코어 (Core), 메모리 (Memory), 버스 (Bus)를 포함한 물리적인 구성과 운용을 위해 필요한 시스템 주파수 (System frequency), 메모리 대기 시간, 동작 주파수 (Operating frequency), 그리고 레지스터 (Register) 설정 등에 복합적으로 영향을 받는다. 표 1은 논문에서 사용한 AT697F, XC2V, GR712RC, GR740 4개의 보드에 대한 환경 구성을 나타낸다. 위의 요소 중에서 변경 가능한 요소들은 동일하게 구성하며 변경 불가능한 요소들에 대해서는 분석 시 그 차이를 반영한다.

1.1 동작 주파수

프로세서의 동작 주파수가 높으면 높을수록 명령어 (Instruction)를 빠르게 처리할 수 있다. 본 논문에서는 보드의 동작 주파수를 그대로 유지하여 실험한 후 동작 주파수에 따라 정규화 한다.

1.2 코어 개수

사용한 4개의 보드에는 싱글코어 (Single core) 프로세서와 멀티코어 프로세서가 모두 포함되어 있다. 본 논문에서는 싱글 코어에서의 성능 비교를 위해 GR712RC, GR740에서 1개의 코어만을 사용한다.

1.3 캐시

캐시는 프로세서와 외부 메모리 사이의 속도차이를 줄여주어 프로세서의 성능을 향상 시킨다. 캐시는 크게 instruction cache (Icache)와 data cache (Dcache)로 구성된다. 4개의 보드는 다양한 캐시 크기를 지니고 있고 이는 캐시의 적중률 (Hit ratio)로 이어져 성능에 영향을 줄 수 있다. 캐시의 자세한 동작 원리는 다음의 논문에 기술되어 있다 [7]. 캐시의 크기는 변경할 수 없기 때문에, 본 논문에서는 캐시의 다양한 형상에서 실험을 수행하고 결과 분석에 반영하였다.

추가로 4개의 보드에는 명령어의 지역성을 이용하여 명령어를 1개의 라인 (8 명령어)씩 가져오는 instruction burst (Iburst) 기능이 있다. 해당 기능도 실험에 포함한다.

1.4 메모리

프로세서가 외부 메모리에 접근할 때 대기 시간을 바탕으로 값을 안정적으로 읽는다. 대기 시간이 작을 경우 값이 깨질 우려가 있고, 대기 시간이 클 경우 보드의 성능을 떨어뜨린다. 따라서 본 논문에서는 보드에서 제공하는 값을 사용한다.

표 2. 각 시스템을 위한 빌드 옵션
Table 2. Build options for each candidate system

Target System	Build Options
AT697F	*-g -O3 -qbsp=leon2 -mcpu=leon -mfix-at697f*
XC2V	*-g -O3 -mcpu=leon3 -mfix-tn0013*
GR712RC	*-g -O3 -qbsp=gr712rc -mcpu=leon3 -mfix-gr712rc*
GR740	*-g -O3 -qbsp=gr740 -mcpu=leon3*

각각의 보드는 서로 다른 메모리 크기를 가지고 있다. 메모리가 클수록 개발자는 더 많은 프로그램을 올릴 수 있을뿐더러, 메모리를 유연하게 쓸 수 있도록 도와준다. 이는 추후 캐시를 고려한 프로그래밍을 통해 높은 캐시 적중률을 끌어내는데 도움을 준다. 하지만 본 논문에서는 캐시를 고려한 프로그램을 수행하는 것이 아니라 정해진 벤치마크 프로그램을 돌리는 것이기 때문에 메모리의 크기가 충분히 크다면 성능에 큰 영향을 주지는 않는다. 벤치마크 프로그램의 경우 4개 보드의 SRAM, SDRAM중 가장 작은 크기인 6MiB로도 충분하다.

1.5 기타

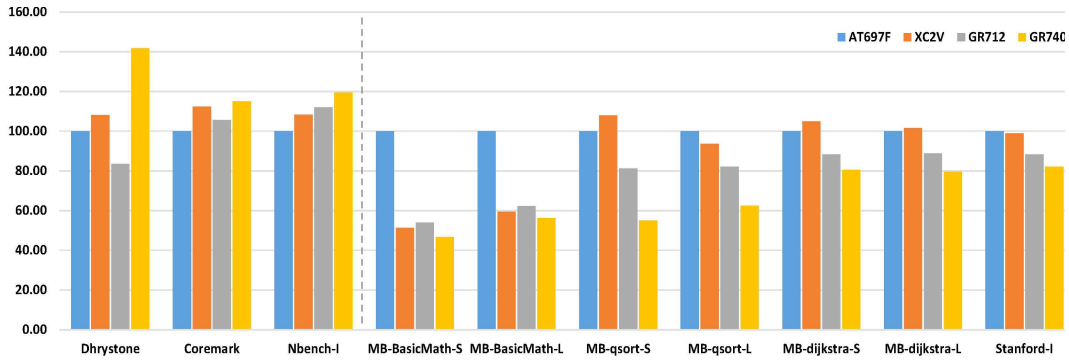
프로세서는 주변에 다양한 장치들과 버스를 통해 연결된다. 해당 버스에서는 프로세서뿐만 아니라 여러 주변장치가 이용하기 때문에 버스 상태에 따라 성능이 크게 달라진다. 본 논문에서는 프로세서 자체에만 집중하기 위해 다른 장치는 최대한 제외한다. 버스를 통해 접근하는 하드웨어 장치는 오직 메모리뿐이다.

2. 소프트웨어 형상

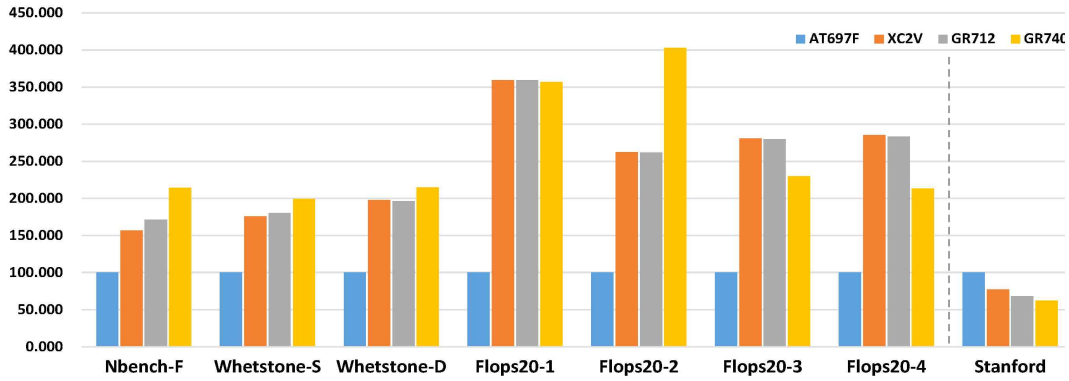
소프트웨어의 측면에서는 성능측정 소프트웨어의 빌드에 사용된 컴파일러의 버전, 프로세서 지원 여부, 최적화 레벨 (Optimization level), 그리고 빌드 옵션 (Build option)에 따라 영향을 받는다. 하드웨어 구성 요소와 달리 소프트웨어의 경우 환경 구성에 있어 많은 유연성을 가지고 있으므로 비교 대상 프로세서 간에 최대한 동일한 환경을 구성한다.

2.1 컴파일러

컴파일러는 사용자가 작성한 C 코드를 기계어 (Machine code) 로 바꿔준다. 바꿔준 기계어 코드는 컴파일러 버전에 따라 동일한 결과를 가져올 지라도 세부적인 구현이 달라진다. 본 논문에서는



(a) 정수 연산에서의 벤치마크 결과



(b) 실수 연산에서의 벤치마크 결과

그림 3. 정수/실수 연산에 대한 벤치마크 결과

Fig. 3 Benchmark results for integer & floating point operations

GCC compiler 7.2.0를 이용한 Bare C 2.0.2 cross compiler를 사용한다 [8].

2.2 빌드 옵션

컴파일러는 다양한 빌드 옵션을 통해 생성되는 기계어를 조절할 수 있다. 빌드 옵션에 따라 성능이 크게 달라지기 때문에 최대한 동일한 환경을 구성한다. 표 2는 각각의 보드에 대한 빌드 옵션이다.

3. 벤치마크 프로그램

본 논문에서는 프로세서들의 성능 측정을 위하여 정수 연산 관점에서 Dhrystone과 Coremark를, 실수 연산 관점에서 Whetstone과 Flops를, 정수 및 실수 관점에서 Stanford를, user application 관점에서 NBench와 MiBench를 선정한다.

3.1 정수 연산

Dhrystone은 가장 널리 쓰이는 마이크로프로세서의 정수 연산 성능 지표로, 특정 연산을 수행하고 그 결과를 수치화하여 제공한다.

Dhrystone은 가장 널리 쓰이지만 벤치마크 알고리즘의 노후화, 컴파일러 최적화 옵션 및 라이브러리 호출에 따른 성능편차, 표준 코드 관리라는 한계점을 지니고 있다 [9]. 이를 대체하기 위해 나온 Coremark도 같이 사용을 한다. Coremark 또한 Dhrystone과 마찬가지로, 수학 연산, state machine 연산, 순환 중복 검사 (Cyclic redundancy check) 연산과 같은 특정 연산을 수행하고 그 결과를 수치화하여 제공한다 [10].

3.2 실수 연산

Whetstone는 124개의 실수 연산으로 구현된 알고리즘을 반복적으로 수행하고, 그 결과를 수치화하여 제공한다 [9]. 결과 값은 단정밀도 (Single precision) 일 때와 배정밀도 (Double precision)일 때를 나눠서 2개를 제공한다.

Flops는 FADD, FSUB, FMUL, FDIV 명령어의 실수 연산을 기반으로 구성된 알고리즘들을 수행하고, 그 결과를 수치화 하여 제공한다 [11].

MFLOP-1, MFLOP-2, MFLOP-3, MFLOP-4, 총 4개의 값을 제공하며 해당 값들은 연산의 행렬화 (Vectorization) 및 FDIV 연산의 비율에서 차이가 있다.

3.3 정수 및 실수

Stanford는 8개의 정수 연산과 2개의 실수 연산으로 구성된 프로그램을 수행하고, 그 결과를 수치화하여 제공한다. 해당 프로그램들은 메모리를 이용한 연산이 빈번한 것이 특징으로, 정수 및 실수 연산에 메모리를 포함한 통합적인 성능 측정이 가능하다 [9].

3.4 User Application

앞에서 선택한 벤치마크 프로그램들은 특정 연산을 반복적으로 수행하여 프로세서의 성능을 측정한다. 하지만 실제 응용프로그램에서는 반복 연산으로만 구성된 프로그램이 거의 없다는 점에 착안하여, user application은 다른 벤치마크 프로그램에 비해 조금 더 실제 응용프로그램에 가깝게 만들어 놓은 벤치마크 프로그램들이다.

NBench는 정렬, 암호화, 네크워크, 수학 연산들로 이루어진 프로그램을 통해 정수 및 실수 연산의 성능을 종합적으로 측정하고, 수치화하여 정수 연산과 실수 연산의 결과를 각각 제공한다.

MiBench는 임베디드 프로세서의 광범위한 활용 측면에서 만들어진 벤치마크 프로그램으로, 프로세서의 성능 측정에 국한되지 않고 저장장치, 통신장치, 기타 I/O를 고려한 시스템의 종합적인 성능을 측정한다 [12]. 본 논문에서는 인공위성 시스템의 특성을 고려하여 MiBench 벤치마크 세트 중 기본 연산, 퀵 정렬 (Quick sorting), 다익스트라 (Dijkstra)를 사용하며, 입력 데이터 양에 따라 large (L), small (S), 2개로 나눠서 각각 진행한다.

Ⅳ. 실험 및 분석

1. 정수 연산

그림 3 (a)는 정수 연산 관련 벤치마크 프로그램을 4개의 프로세서에서 수행한 결과이다. 파란색, 주황색, 회색, 노란색은 각각 AT697F, XC2V, GR712RC, GR740의 성능을 나타내며, X축은 벤치마크 프로그램 종류, Y축은 결과 값을 나타낸다. 결과 값은 동작 주파수를 이용하여 정규화 한 뒤, AT697F의 성능을 기준으로 한 번 더 정규화 하였다. 앞의 3개 항목 (Bin)은 성능 지수로 값이 클수록 좋은 성능을 나타내며, 나머지 7개 항목은 수행 시간으로 값이 작을수록 좋은 성능을 나타낸다.

4개의 보드 중에 GR740이 모든 벤치마크에서 가장 좋은 성능을 보여주었다. 이는 LEON4의 코어 성능이 뛰어난 것을 의미한다. XC2V 보드는 AT697F보다 7개가 높고 3개가 낮게 나타났다. GR712RC의 경우 Dhrystone을 제외한 나머지 벤치마크에서 AT697F보다 좋게 나왔다. AT697F 보다는 XC2V, GR712RC가 평균적으로 뛰어나다고 할 수 있지만 상위 호환이라고 보기는 어려웠다. LEON3 내에서 FPGA와 ASIC의 경우, 6개 벤치마크에서 ASIC이 좋은 성능을 보여주었고 5개 벤치마크에서 FPGA가 더 좋은 성능을 보여주어서 큰 차이가 있지 않았다. 이는 ASIC이나 FPGA의 설계에 따라 충분히 바뀔 수 있는 수준이다.

2. 실수 연산

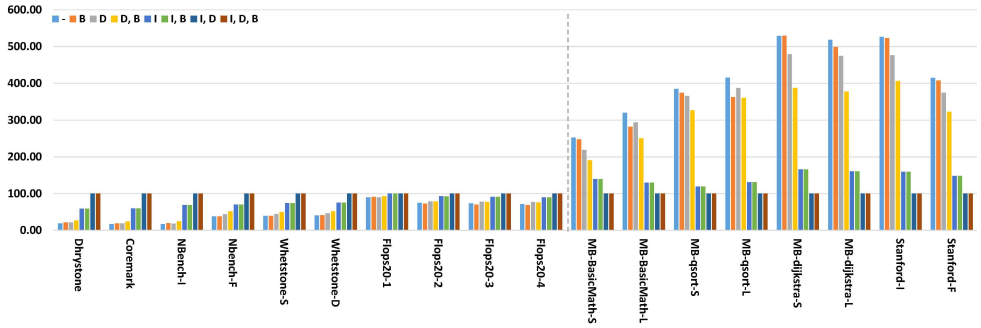
그림 3 (b)는 실수 연산 관련 벤치마크 프로그램을 4개의 프로세서에서 수행한 결과이다. 각각의 색과 축이 의미하는 바는 그림 3 (a)의 의미와 동일하며, AT697F의 결과와 동작 주파수를 이용하여 정규화 하였다. 앞의 7개 항목은 성능 지수로 값이 클수록 좋은 성능을 나타내며, 마지막 항목은 수행 시간으로 값이 작을수록 좋은 성능을 나타낸다.

AT697F의 실수 연산이 가장 느렸다. 이는 정수 연산이 실수 연산의 성능에 영향을 주었다는 점도 있지만, 주로 FPU의 구조적인 차이로 인해 발생하였다. AT697F는 MEIKO FPU를 사용하며 XC2V 보드는 GRFPU-Lite를, GR712/GR740의 경우 GRFPU를 사용한다. Meiko FPU는 FPU 큐 (Queue)가 없어서 실수 연산 시 정수 처리장치 (Integer unit, IU)가 실수 연산이 종료될 때까지 정지한다. 반면 GRFPU-Lite, GRFPU는 큐가 있어서 실수 연산 시에도 IU가 계속 수행된다. 이러한 차이 때문에 AT697F의 실수 연산 성능 지수가 가장 낮게 나왔다.

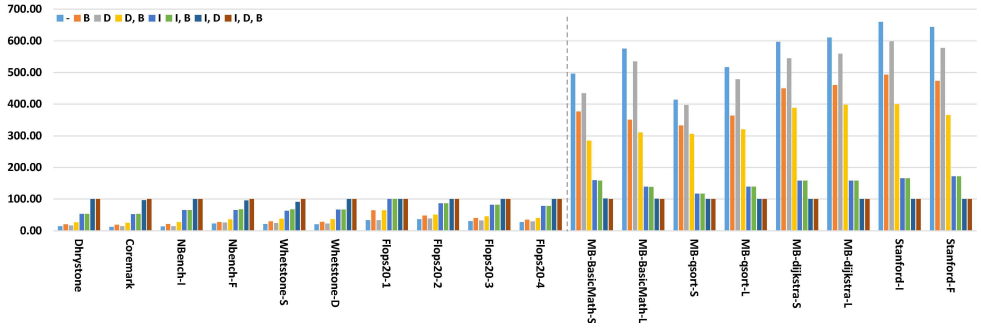
나머지 GRFPU와 GRFPU-Lite 를 사용한 프로세서들 가운데는 전반적으로 비슷하긴 하지만 GR740이 좋은 성능을 보였다. 이는 실수 연산이 정수 연산을 동반하였기 때문이다.

3. 캐시 연산

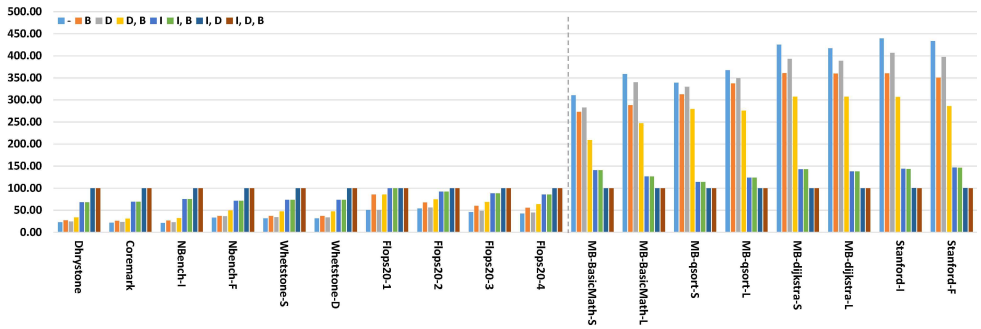
그림 4은 4개의 프로세서에서 Icache (I) 활성화/비활성화, Dcache (D) 활성화/비활성화, Iburst (B) 활성화/비활성화, 총 8개의 캐시 형상에 따른 성능을 측정한 결과이다. 각 축이 의미하는 바는 그림 3의 의미와 같으며, 각각의 결과 값은 Icache 활성화, Dcache 활성화, Iburst 활성화의 값으로 정



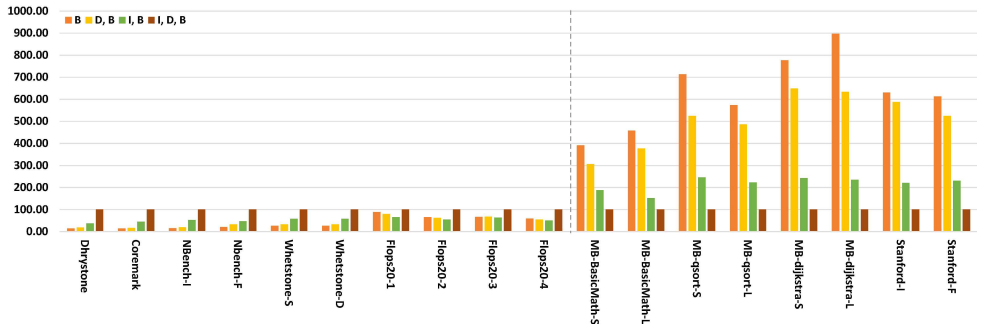
(a) AT697F 벤치마크 결과



(b) XC2V 벤치마크 결과



(c) GR712RC 벤치마크 결과



(d) GR740 벤치마크 결과

그림 4. 각 후보 시스템에서의 벤치마크 결과
Fig. 4 Benchmark results for each candidate system

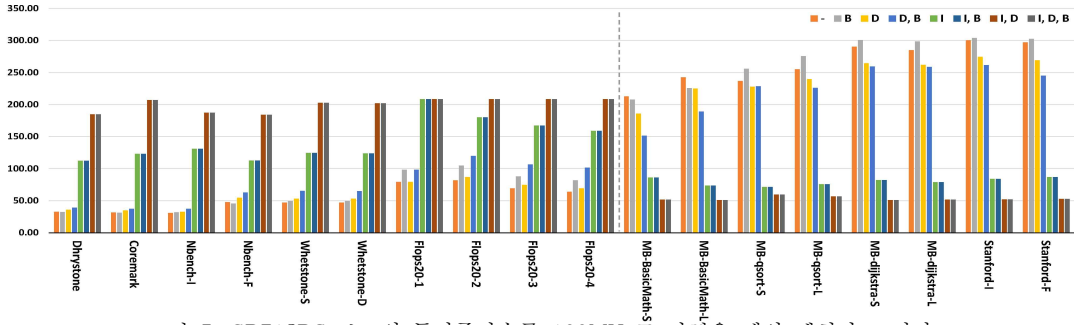


그림 5. GR712RC 보드의 동작주파수를 100MHz로 바꿨을 때의 벤치마크 결과
Fig. 5 Benchmark results for GR712RC with 100MHz operating frequency

규화 하였다. 보드별로 그래프를 분리하였기 때문에 동작 주파수로 정규화 하지는 않았다. 각 색이 나타내는 것은 옆의 범례에 나와 있다. 앞의 10개 항목은 성능 지수로 값이 클수록 성능이 뛰어난 것이며 뒤의 8개 항목은 수행 시간으로 값이 작을수록 성능이 뛰어난 것이다. GR712RC는 48MHz에서 동작시켰으며, GR740의 경우 항상 Iburst를 활성화여서 각각의 항목에 4개의 색만 표시하였다.

AT697F에서의 결과를 살펴보면, 정수 연산에서는 cache에 의해 속도가 최대 5배까지 빨라졌다. 하지만 실수 연산에서는 다른 3개의 보드에 비해 캐시 성능이 크게 떨어졌다. 이는 앞에서 말한 MEIKO FPU의 특징 때문으로 추정한다. 추가로 다른 보드들과 달리 Icache & Dcache disable일 때 burst 의 효과가 거의 없던 경우도 있었다.

XC2V & GR712RC는 매우 유사한 경향을 보여주었다. 다만 전체적으로 FPGA로 만든 XC2V에서 cache로 인한 성능 향상이 컸다. MiBench 함수들을 살펴보면, XC2V에서는 cache로 인해 6배 이상 좋아지지만 GR712RC에서는 4배 정도 좋아졌다. 그림 3의 XC2V보다 GR712RC의 결과 값이 더 좋은 것을 토대로 보았을 때, 이는 cache miss일 때 data 지연시간이 XC2V가 더 큰 것으로 나타났다.

GR740은 다른 3개와는 다른 양상을 보인 그래프들이 있다. Flops 계산에 있어서 Icache 또는 Dcache중 1개만 enable하였을 때, Icache & Dcache를 모두 disable했을 때보다 더 느려지기도 했다. 2개를 모두 enable하였을 때는 확실히 성능이 좋아졌다. 동일한 FPU 연산인 Whetstone에서는 일반적인 경향이 나타났기 때문에 Flops 벤치마크 프로그램의 특정 알고리즘이 캐시의 성능을 떨어뜨리는 것으로 추정한다.

3.1 Icache & Dcache

Icache로 인해 성능이 크게 향상된 것을 알 수 있다. 그 이유는 벤치마크 프로그램을 수행하는데 있어서 Icache를 Dcache보다 더 많이 참조하기 때문이다. 이는 일반적인 프로그램과 달리, 벤치마크 프로그램은 크기가 작을뿐더러 성능 측정을 위해 특정 코드를 반복 수행하는 경향이 있기 때문이다. 한번 코드를 수행했을 경우 대부분의 명령어들이 Icache에 저장되면서 높은 적중률을 보여주었다. 한 예로 캐시 시뮬레이터를 통해 XC2V 보드를 분석해 본 결과, MiBench의 basicmath-L의 경우 99%, Dijkstra-L의 경우 96% 이상의 Icache 적중률을 보여주었다.

Dcache의 경우 성능 향상이 있었지만 Icache의 성능 상승 만큼 크지는 않았다. 이는 벤치마크 프로그램에 데이터 접근이 큰 연산이 없었기 때문이다. 이런 벤치마크 프로그램의 단점을 극복하고자 메모리 복사를 성능 측정에 추가한 연구도 있다 [3].

3.2 Cache burst

4개의 프로세서 모두 Iburst를 활성화 하였을 때 성능이 크게 좋아졌다. 특히 Icache가 비활성화 되었을 때 Iburst로 인한 성능 향상이 컸다. 이는 IU 내부에 명령어들을 저장할 수 있는 임시 버퍼가 있다는 것을 의미한다. 다만 Icache가 비활성화 되었기 때문에 수행한 명령어들은 수행 후 사라지며, 매번 메모리에서 다시 불러왔다. 실재 명령어들을 가져올 때 라인 당 최대 4 동작 주기까지 발생할 수 있다 [13].

4. 동작 주파수

그림 5는 GR712RC에서 동작 주파수를 48MHz에서 100MHz로 바꾼 뒤 벤치마크 프로그램을 수

행한 결과이다. 각축의 의미는 그림 3의 의미와 동일하며 GR712RC 48MHz에서의 Icache & Dcache cache 활성화, Iburst 활성화의 값으로 정규화 하였다. 앞의 10개 항목은 성능 지수로 값이 클수록 성능이 뛰어난 것이며, 뒤의 8개 항목은 수행 시간으로 값이 작을수록 성능이 뛰어난 것이다.

Icache & Dcache cache 활성화, Iburst 활성화 상태를 살펴보면 동작 주파수가 2배 가까이 증가함에 따라 벤치마크 프로그램으로 측정한 성능도 2배 가까이 상승했다. 하지만 Icache & Dcache를 비활성화 했을 때에는 속도가 2배까지 나오지 않았다. 그 이유는 동작 주파수가 2배가 증가하더라고 메모리에서 값을 가져오는데 필요한 시간, 명령어들 간의 종속성 등에 의해 시간이 소요되면서 주파수에 비례한 성능이 나타나지 않았기 때문이다. 나머지 Icache, Dcache, Iburst에 따른 영향은 48MHz에서의 동작과 유사했다.

V. 결 론

본 논문에서는 다양한 벤치마크 프로그램을 이용하여 하드웨어 및 소프트웨어 형상에 따라 보드의 성능을 측정하여 전체적인 경향성을 파악하였다. 이를 통해 추후 차세대 위성의 프로세서를 선정하는데 있어서 현재의 프로세서와 비교를 통해 가이드라인을 제시한다.

우리가 측정한 성능 결과는 제조사가 제시한 성능과는 차이가 있었다. 제조사는 Dhrystone, Whetstone, Linpack, Coremark, Autobench, FPMark, Coremark-pro 7개 벤치마크 프로그램을 수행하고, 100MHz에서 동작하는 GR712보다 GR740이 평균 3.335배 뛰어난 값이 나왔다고 제시했다 [1]. 하지만 개발보드라는 동일한 하드웨어 형상과 우리가 원하는 소프트웨어 형상을 이용하여 벤치마크 프로그램을 수행할 경우, Coremark는 15% 높게, Whetstone은 5% 낮게, Dhrystone은 4% 높게 나왔다. 실수 연산에서는 GR712RC의 성능보다 GR740의 성능이 3.335배가 아닌, 평균 3.00배만 빠르게 나왔다. 여기에 하드웨어 형상까지 추가적으로 바뀐다면 제조사의 벤치마크 결과만으로는 성능 변화를 예측하기 더욱 어려울 것이다. 따라서 이와 같은 실측값을 바탕으로 프로세서를 선정해야 한다.

본 논문에서는 싱글 코어 성능을 비교하고 위해 GR712RC의 2개 코어 중 1개만을 사용하고 GR740의 4개 코어 중 1개만 사용했다. 코어를 더 쓰면 연산량이 증가하기 때문에 이는 잠재적은 성능 향상의 요소가 될 수 있다. 차세대 프로세서 선정에서는 FPU 큐가 있는 FPU를 탑재한 프로세서를 추천한다. 해당 특징은 성능을 크게 향상시킨다.

캐시의 경우 적중률에 따라 성능이 크게 달라진다. 본 논문에서 사용한 벤치마크 프로그램처럼 크기가 SRAM, SDRAM의 메모리 크기에 비해 작을 경우, 전부 캐시에 들어가 높은 적중률을 보여 주었지만, 실제 용량이 크고 복잡한 위성비행소프트웨어에서는 캐시에 다 들어가지 않기 때문에 벤치마크 프로그램 정도의 성능 향상을 기대하기는 어려울 것이다. 이 경우 캐시의 크기가 크면 클수록 높은 적중률을 보이게 된다. 하지만 캐시의 성능을 더 잘 보이기 위해서는 캐시의 크기보다는 메모리 배치 등을 고려한 프로그래밍 기법이 더 중요하다 [7]. 이 때 메모리의 크기가 크면 개발자가 좀 더 유연하게 메모리를 구성하여 캐시의 적중률을 높일 수 있다. 추가로, Iburst는 가급적이면 사용하는 것이 성능 향상에 도움이 된다.

동작 주파수가 성능과 정비례하지는 않았다. 동작 주파수의 증가량에 비해 성능의 증가량은 조금 떨어졌는데 이는 주변 하드웨어 요소 때문이다. 본 논문에서는 메모리만 다루었지만 추후 다른 하드웨어 요소들과 통신이 추가되면 동작 주파수 향상으로 인한 성능 향상은 더 줄어들 것이다.

마지막으로 하드웨어 복잡성으로 인해 벤치마크 프로그램을 이용한 성능측정 시 정확한 동작 예측에는 한계가 있다. 따라서 벤치마크 프로그램의 결과를 해당 프로세서의 절대적인 성능으로 삼는 것은 위험할 수 있다. 가장 좋은 벤치마크 프로그램은 실제 위성에서 동작하는 탑재소프트웨어이다. 하지만 탑재소프트웨어의 모든 요소를 개발 보드에서 동작시키는 것은 시간 및 비용이 많이 들뿐더러 모든 하드웨어 요소가 갖추어지지 않는 한 불가능하다. 따라서 본 논문에서는 탑재소프트웨어의 연산 특성을 고려한 벤치마크 프로그램을 선정하여 수행하였다.

본 논문을 통해 분석한 프로세서 성능은 추후 차세대 프로세서 선정에 반영될 것이다. 선정된 프로세서에는 위성의 임무에 맞게 하드웨어가 설계 및 연결이 되고, 위성의 탑재소프트웨어가 이식(Porting)된다. 그 후 정확한 연산량 분석을 포함한 엄격한 검증 프로세스를 통해 높은 신뢰도의 위성 개발에 사용될 예정이다. 뿐만 아니라, 추후 위성 운용에 있어서 형상을 바꿔야할 필요가 있을 때 유용하게 쓰일 것이다.

References

[1] GR740 technical note on benchmarking and validation, Cobham Gaisler, 2017.

[2] Benchmark performance UT699E/700 LEON 3FT, COBHAM Gaisler, 2017.

[3] D. Bekker, "Performance Analysis of Standalone and in-FPGA LEON3 processors," Proceedings of 10th Wrks. Spacecraft Flight Software, Johns Hopkins University, MD, 2017.

[4] R. Llorca-cejudo, O. Frandon, "Cache-induced Execution Time Variability of a Satellite On-board SW in a LEON-2 microprocessor," Proceedings of Data Systems in Aerospace, Vol. 694, 2011.

[5] R. Ginosar, "Survey of Processors for Space," Proceedings of Data Systems in Aerospace, pp. 1-5, 2012.

[6] J.-W. Choi, Y-J. Cheon, "Study of Next Space Processors for Development of Flight Software," Proceedings of Conference Korea Society of Aeronautical & Space Science, pp. 809-814, 2012 (in Korean).

[7] J.-W. Choi, J.-Y. Jeong, B.-S. Yoo, "Flight Software Operation for LEON2-FT/AT697F Processor cache," Proceedings of Conference Korea Society of Aeronautical & Space Science, pp. 358-362, 2016 (in Korean).

[8] Cobham Gaisler AB, "BCC User Manual," December 2017, Version 2.0.2

[9] W. J. Price, "A Benchmark Tutorial," Journal of IEEE Micro, Vol. 9, No. 5, pp.28-43, 1989.

[10] J. A. Poovey, T. M. Conte, M. Levy, S. Gal-On, "A Benchmark Characterization of the EEMBC Benchmark Suite," Journal of IEEE Micro, Vol. 29, No. 5, pp.18-29, 2009.

[11] A. Aburto, FLOPS 2.0 C program, 1992.

[12] M. R. Guthaus, J. S. Ringenberg, D. Ernst, T. M. Austin, T. Mudge, R. B. Brown, "MiBench: A Free, Commercially Representative Embedded Benchmark Suite," Proceedings of IEEE International Workshop Workload Characterization, pp. 3-14, 2001.

[13] M. Prieto, D. Guzman, D. Meziat, S. Sanchez, L. Planche, "LEON2 Cache Characterization-A Contribution to WCET Determination," Proceedings of IEEE. International Symposium Intelligent Signal Processing, pp. 1-6, 2007.

Bum-Soo Yoo (유 범 수)

He received the Ph.D. degree in electrical engineering from Korea Advanced Institute of Science and Technology in 2016.

Since 2016, he has been a Senior Researcher with the Satellite Bus Development Division, Korea Aerospace Research Institute. His current research interests include embedded systems and robotics.

Email: bsyoo@kari.re.kr

Jong-Wook Choi (최 종 욱)

He received the Ph.D. degree in electrical engineering from Chungnam National University in 2016.

He has been a Senior Researcher and Principal Researcher with the Satellite Bus Development Division, Korea Aerospace Research Institute after 2000 and 2016, respectively. His current research interests include embedded systems and simulator.

Email: jwchoi@kari.re.kr

Jae-Yeop Jeong (정 재 엽)

He received the M.S. degree in computer engineering from Chungnam National University in 2009.

From 2009 to 2013, he was a Senior Researcher in the Avionics Research Center, LIGNex1. Since 2014, he has been a Senior Researcher with the Satellite Bus Development Division, Korea Aerospace Research Institute. His current research interests include embedded system and real-time operating system.

Email: jyjeong@kari.re.kr

Sun-Wook Kim (김 선 욱)

He received the M.S. degree in computer engineering from Chungnam National University in 2008.

From 2008 to 2017, he worked for Samsung electronics as a senior researcher. Since 2017, he has been a senior researcher with the satellite bus development division, Korea Aerospace Research Institute. His research interest includes embedded system software, embedded processor architecture and embedded virtualization.

Email: sunwookkim@kari.re.kr