JKIICE Journal of the Korea Institute of Information and Communication Engineering

한국정보통신학회논문지 Vol. 25, No. 1: 20~26, Jan. 2021

에너지 효율적인 FPGA 가속기 설계를 위한 하드웨어 및 소프트웨어 공동 설계 플랫폼

이동규[1] · 박대진[2*]

Hardware and Software Co-Design Platform for Energy-Efficient FPGA Accelerator Design

Dongkyu Lee[1] · Daejin Park[2*]

[1]Ph.D. Student, School of Electronic and Electrical Engineering, Kyungpook National University, Daegu, 41566 Korea
[2*]Assistant Professor, School of Electronic and Electrical Engineering, Kyungpook National University, Daegu, 41566 Korea

요 약

오늘날의 시스템들은 더 빠른 실행 속도와 더 적은 전력 소모를 위해 하드웨어와 소프트웨어 요소를 함께 포함하고 있다. 기존 하드웨어 및 소프트웨어 공동 설계에서 소프트웨어와 하드웨어의 비율은 설계자의 경험적 지식에 의해 나뉘었다. 설계자들은 반복적으로 가속기와 응용 프로그램을 재구성하고 시뮬레이션하며 최적의 결과를 찾는다. 설계를 변경하며 반복적으로 시뮬레이션하는 것은 시간이 많이 소모되는 일이다. 본 논문에서는 에너지 효율적인 FPGA 가속기 설계를 위한 하드웨어 및 소프트웨어 공동 설계 플랫폼을 제안한다. 제안하는 플랫폼은 가속기를 구성하는 주요 성분을 변수화해 응용 프로그램 코드와 하드웨어 코드를 자동으로 생성하여 설계자가 적절한 하드웨어 비율을 쉽게 찾을 수 있도록 한다. 공동 설계 플랫폼은 Xilinx Alveo U200 FPGA가 탑재된 서버에서 Vitis 플랫폼을 기반으로 동작한다. 공동 설계 플랫폼을 통해 1000개의 행을 가지는 두 행렬의 곱셈 연산 가속기를 최적화한 결과 응용 프로그램보다 실행 시간이 90.7%, 전력 소모가 56.3% 감소하였다.

ABSTRACT

Recent systems contain hardware and software components together for faster execution speed and less power consumption. In conventional hardware and software co-design, the ratio of software and hardware was divided by the designer's empirical knowledge. To find optimal results, designers iteratively reconfigure accelerators and applications and simulate it. Simulating iteratively while making design change is time-consuming. In this paper, we propose a hardware and software co-design platform for energy-efficient FPGA accelerator design. The proposed platform makes it easy for designers to find an appropriate hardware ratio by automatically generating application program code and hardware code by parameterizing the components of the accelerator. The co-design platform based on the Vitis unified software platform runs on a server with Xilinx Alveo U200 FPGA card. As a result of optimizing the multiplication accelerator for two matrices with 1000 rows, execution time was reduced by 90.7% and power consumption was reduced by 56.3%.

키워드 : 공동 설계 플랫폼, FPGA, Alveo U200, Vitis 통합 소프트웨어 플랫폼, 하드웨어 가속

Keywords : Co-design platform, FPGA(Field programmable gate array), Alveo U200, Vitis unified software platform, Hardware acceleration

Received 25 November 2020, **Revised** 28 November 2020, **Accepted** 30 November 2020

*** Corresponding Author** Daejin Park(E-mail:boltanut@knu.ac.kr, Tel:+82-53-950-5548)
Assistant Professor, School of Electronic and Electrical Engineering, Kyungpook National University, Daegu, 41566 Korea

Open Access http://doi.org/10.6109/jkiice.2021.25.1.20 print ISSN: 2234-4772 online ISSN: 2288-4165

ⓒThis is an Open Access article distributed under the terms of the Creative Commons Attribution Non-Commercial License(http://creativecommons.org/li-censes/by-nc/3.0/) which permits unrestricted non-commercial use, distribution, and reproduction in any medium, provided the original work is properly cited.
Copyright ⓒ The Korea Institute of Information and Communication Engineering.

Ⅰ. 서 론

사물 인터넷(IoT)은 센서와 소프트웨어를 내장하고 있는 임베디드 장치들이 연결되어 자료를 수집, 교환, 처리하는 네트워크이다. IoT 장치들은 센서에서 받은 데이터를 내장된 소프트웨어 알고리즘을 통해 처리하고, 다른 장치에 전달한다. 최근 IoT는 홈 IoT, 헬스 케어, 스마트 시티 등의 여러 분야에 확장되었고 그에 따라 장치에서 처리하는 데이터의 양도 빠르게 증가하고 있다 [1].

임베디드 장치들은 내장된 프로세서로 실행하는 소프트웨어 알고리즘만으로 데이터를 처리한다. 임베디드 장치의 부족한 프로세서 성능으로는 대용량 정보를 처리해야 하는 최근 IoT 장치의 요구 조건을 만족할 수 없다. 따라서 대부분의 IoT 시스템에서 임베디드 장치로 구성된 에지들은 데이터 수집, 데이터 전처리 같은 단순한 작업만을 처리한다. 엣지의 부족한 연산 능력을 보완하기 위해 클라우드 컴퓨팅이나 엣지 컴퓨팅 방식 등 서버의 자원을 사용하여 복잡한 연산을 빠르게 처리한다 [2]. 클라우드 컴퓨팅은 인터넷을 통해 서버의 자원을 이용하여 연산 속도를 높이는 기술로 빠른 연산이 가능하지만 데이터 전송이 많아 네트워크 속도에 영향을 많이 받는다 [3]. 엣지 컴퓨팅은 분산 컴퓨팅의 일종으로 가까이 있는 서버에서 데이터를 처리함으로써 응답 속도와 데이터 전송량을 줄인다 [4].

프로그램 처리 속도를 높이기 위해 하드웨어 가속기를 이용하는 기술이 IoT에 접목되었다 [5, 6]. 범용 프로세서는 다양한 연산을 수행할 수 있지만, 연산 속도가 느리다. 반면 하드웨어 가속기는 특정한 연산만을 수행할 수 있지만, 프로그램 처리 속도가 매우 빠르다. 엣지는 작은 하드웨어 가속기를 추가하여 이전보다 복잡한 연산을 전력 소모 대비 실행 시간 효율이 높이고, 서버도 하드웨어 가속기를 추가하여 프로세서만을 사용할 때보다 높은 연산 능력을 갖추고 있다.

가속기는 크기가 커질수록 다양한 연산을 제공할 수 있어 응용 프로그램을 더 빠르게 수행할 수 있지만, 하드웨어가 소모하는 전력이 커진다. 반대로 가속기의 크기가 작아지면 하드웨어가 소모하는 전력이 작아지지만, 프로그램 실행 시간이 길어져 전체 소모 전력이 커질 수 있다. 그러므로 하드웨어와 소프트웨어를 융합하여 설계할 때에는 소프트웨어가 처리하는 부분과 하드웨어가 처리하는 부분의 비율을 적절하게 배분해야 한다 [7, 8].

본 논문에서는 행렬 곱셈 연산을 예제로 제한된 상황에서 최적 에너지 효율을 가지는 가속기를 설계하기 위한 플랫폼을 제안한다. 제안하는 플랫폼은 가속기를 구성하는 요소를 변수화하여 입력되는 변수에 따라 가속기와 응용 프로그램을 재생성하고 하드웨어에 합성, 적재, 실행한다. 하드웨어를 쉽게 합성하고 적재하기 위해 우리는 FPGA(field-programmable gate array)를 이용한다. FPGA는 장치에서 제공하는 게이트 배열(gate array) 사이의 연결을 이용하여 하드웨어를 표현하는 집적 회로(integrated circuit)로 고정된 하드웨어인 ASIC (application-specific integrated circuit)에 비해 하드웨어를 합성하는 시간이 짧아 다양한 하드웨어를 합성하고 실행하는 플랫폼에 적합하다 [9, 10].

Ⅱ. 시뮬레이션 환경

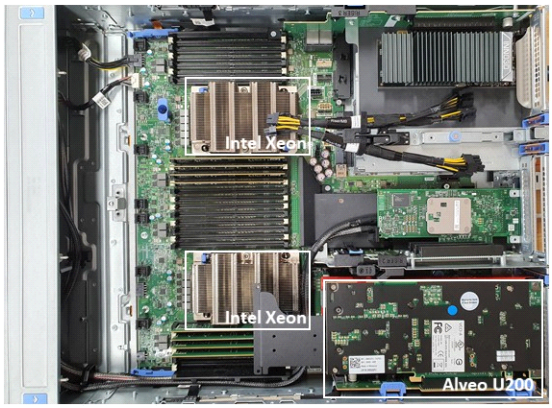


Fig. 1 Alveo U200 server platform

제안하는 플랫폼은 2개의 Intel Xeon Bronze 3204 프로세서와 RAM 16GB, Xilinx Alveo U200 FPGA 카드를 포함하는 그림 1의 서버에서 동작한다. Intel 사의 Xeon Bronze 3204 프로세서는 6개의 스레드와 1.9 GHz의 동작 속도, 2133MHz의 메모리 속도를 가진다. Xilinx 사의 UltraSacle 아키텍처 기반의 Alveo U200 FPGA는 892,000개의 LUT(look-up table)을 가지고 있으며, INT8 정밀도에서 18.6 TOPs(tera-operation per seconds)의 속도를 가진다.

우리가 사용한 Vitis 통합 소프트웨어 플랫폼은 Xeon 프로세서에서 실행되는 응용 프로그램과 Alveo U200 FPGA에 합성되어 동작하는 가속기 사이의 연결을 자동으로 생성하여 소프트웨어와 하드웨어가 독립적으로 개발될 수 있도록 도와준다. Vitis 플랫폼은 C/C++, Python 같은 고수준 언어와 RTL(register-transfer level) 코드 같은 저수준 언어를 사용하여 하드웨어를 합성할 수 있다. 그림 2는 GCC(GNU compiler collection)를 이용한 응용 프로그램 생성과 Vitis 플랫폼을 이용한 하드웨어 합성 과정을 나타낸다. C/C++로 작성된 파일들은 GCC에 의해 각각 컴파일되고 링커에 의해 통합되어 하나의 실행 파일을 생성한다. Vitis 플랫폼은 GCC와 비슷한 합성 과정을 거친다. C/C++, Python, OpenCL, RTL 등으로 기술된 하드웨어들은 각각 Vitis 컴파일러를 통해 Xilinx 목적 파일로 컴파일된다. Xilinx 목적 파일들은 Vitis 플랫폼의 링커를 통해 목표하는 하드웨어에 맞는 FPGA 이진 파일로 만들어진다.

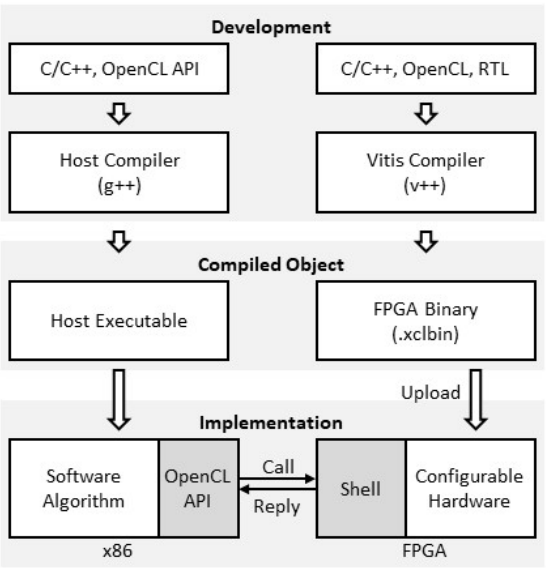


Fig. 2 The application program compilation process using GCC and hardware synthesis process using Vitis platform

Vitis 플랫폼에서 설계된 하드웨어들은 응용 프로그램에서 OpenCL API를 통해 특별한 설정 없이 즉시 사용할 수 있다. 또한, 하드웨어의 적재와 응용 프로그램을 운영체제(operating system)에서 관리할 수 있는 단순한 프로그램의 형태로 실행하여 사용자에게 FaaS(FPGA-as-a-service) 환경을 제공할 수 있다. 그림 3은 Vitis 플랫폼을 이용한 가속기 설계 흐름을 나타낸다. Vitis 컴파일러는 C/C++, Python, OpenCL 등의 고수준 언어로 작성된 하드웨어, 또는 저수준 언어인 RTL로 기술된 하드웨어를 컴파일 및 링킹 과정을 통해 Alveo FPGA에 적재할 수 있는 비트스트림으로 만든다. 응용 프로그램은 C/C++로 작성된 알고리즘과 OpenCL API로 작성된 가속기 실행 명령으로 이루어진다.

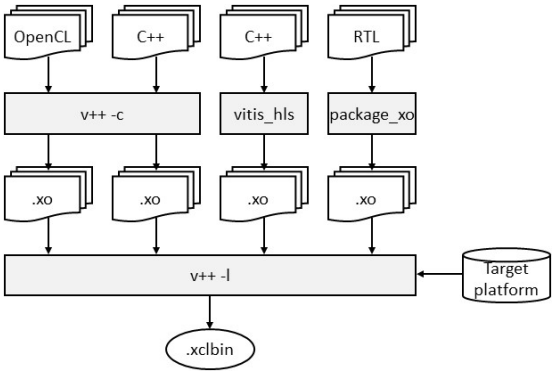


Fig. 3 Accelerator design flow with Vitis platform

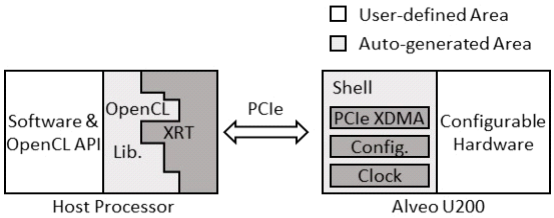


Fig. 4 Concept of the Alveo FPGA and host processor

그림 4는 Vitis 플랫폼을 기반으로 하는 호스트 프로세서와 Alveo FPGA의 연결 배치를 나타낸다. 호스트 프로세서와 Alveo FPGA는 PCIe 인터페이스를 이용하여 하드웨어 제어 명령을 주고받으며 연산을 수행한다. XRT(Xilinx runtime library)는 하드웨어 통신 계층을 추상화하여 하드웨어의 지식이 없어도 설계를 할 수 있도록 도와준다. 사용자는 함수의 형태로 하드웨어 제어를 더 추상화한 OpenCL API를 이용하여 소프트웨어 알고리즘과 하드웨어를 통합한 프로그램을 설계한다. Vitis 플랫폼은 FPGA에 적재될 하드웨어와 하드웨어의 통신 및 제어에 필요한 연결 드라이버, 클록 생성 블록, 구성 정보를 함께 생성한다. 하드웨어는 Alveo의 재구성 가능한 하드웨어에 적재되고, 연결 드라이버와 구성 정보는 쉘 구역에 적재되어 호스트 프로세서와 Alveo FPGA 간의 매끄러운 연결을 지원한다.

Ⅲ. 효율적인 FPGA 가속기 설계 플랫폼

3.1. Accelerator Based on High-level Language

효율적인 FPGA를 위한 플랫폼을 설계하기 위해 본 논문에서는 2개의 정방 행렬을 곱하는 하드웨어를 사용한다. 제안하는 플랫폼은 다양한 크기의 행렬 곱셈 가속기를 생성하고, 알고리즘을 통해 큰 행렬을 연산하여 실행 시간과 하드웨어의 크기를 측정한다. 행렬 곱셈은 연산에 집중적인 작업이기 때문에 가속기의 성능을 확인할 수 있는 지표가 될 수 있다.

그림 5는 본 논문에서 사용한 행렬 연산 가속기의 설계를 보여준다. 제안하는 플랫폼에서는 가속기의 설계를 단순화해 하드웨어 크기를 쉽게 변수화하기 위해 하드웨어 기술 언어(hardware description language)로 C++을 사용했다. 가속기는 두 개의 행렬 데이터 *matA*, *matB*와 각 행렬의 행과 열 개수를 입력으로 받고 하나의 행렬 데이터 *matC*를 출력한다. 가속기는 호스트의 전역 메모리 접근을 위한 마스터 AXI(advanced extensible interface) 버스와 스칼라 데이터와 제어 신호를 위한 슬레이브 AXI4-Lite 버스를 Vitis의 HLS(high-level synthesis) pragma로 사용한다. 가속기는 버스트 읽기를 통해 전역 메모리에 저장된 두 개의 행렬을 읽고 병렬적으로 연산한 후 버스트 쓰기를 통해 전역 메모리에 결과를 다시 저장한다. 가속기의 각 동작은 HLS를 통해 최적화된다.

Algorithm 1: C++ based matrix multiplication accelerator

```
1  mmult(matA, matB, matC, a_row, a_col, b_col)
2  % matA, matB : Read-only matrix
3  % a_row, a_col : Number of rows and columns of matA
4  % b_col : Number of columns of matB
5  % matC : Matrix result

6  % AXI bus definition with HLS
7  Set HLS m_axi bus for global memory
8  Set HLS s_axilite bus for data and control

9  Set local memory localA, localB, localC, divide dimension

10 % Memory burst read
11 localA ← matA
12 localB ← matB

13 % systolic matrix multiply
14 localC = localA × localB

15 % Burst write
16 matC ← localC
```

Fig. 5 C++ based matrix multiplication accelerator algorithm

3.2. Host Application with OpenCL API

Vitis 플랫폼 기반 응용 프로그램은 소프트웨어 알고리즘 실행뿐만 아니라 OpenCL API를 통한 가속기 설정 및 실행을 통해 소프트웨어와 하드웨어의 통합 설계를 가능하게 한다. 그림 6은 응용 프로그램에서의 가속기 실행 알고리즘을 나타내고, 그림 7은 시뮬레이션 환경에서 응용 프로그램과 가속기의 실행 흐름을 나타낸다. 그림 7의 검은 화살표는 응용 프로그램의 제어를 나타내며 회색 화살표는 가속기 내부에서의 제어 흐름을 나타낸다.

응용 프로그램은 가속기를 설정하고 실행하기 위해 호스트에서 사용 가능한 장치를 찾아 설정한다. OpenCL API는 큐를 통해 하드웨어를 제어하므로 컨텍스트와 명령어 큐를 생성 및 설정한다. Vitis 도구를 통해 합성된 이진 파일은 응용 프로그램의 명령에 따라 FPGA에 적재된다. 응용 프로그램에서 전역 메모리에 연산을 위한 데이터를 전송한 후 커널을 실행하면 가속기는 전역 메모리의 데이터를 읽고, 연산한 후 결과를 전역 메모리에 저장한다. 응용 프로그램은 전역 메모리에 저장된 결과를 읽어 다음 연산을 실행한다.

Algorithm 2: Host application with OpenCL API

```
1  Set available device
2  Create context queue, command queue
3  Load xclbin to FPGA
4  Set buffer for data transition
5  Set kernel arguments

6  % matrix multiplication using accelerator
7  for row ≤ Data size do
8    for col ≤ Data size do
9      for window ≤ Data size do
10       Insert data to buffer
11       Send buffered data to global memory
12       Launch the kernel
13       Receive data from global memory
14       Calculate result
15     end for
16   end for
17 end for
```

Fig. 6 Kernel execution algorithm in host application

3.3. Parameterized Design Platform

제안한 하드웨어 가속기와 응용 프로그램은 변수화를 통해 가속기가 한 번에 처리할 수 있는 행렬의 크기와 응용 프로그램 알고리즘을 변경할 수 있다. 그림 8은 본 논문에서 제안하는 FPGA 최적화 플랫폼의 개념을

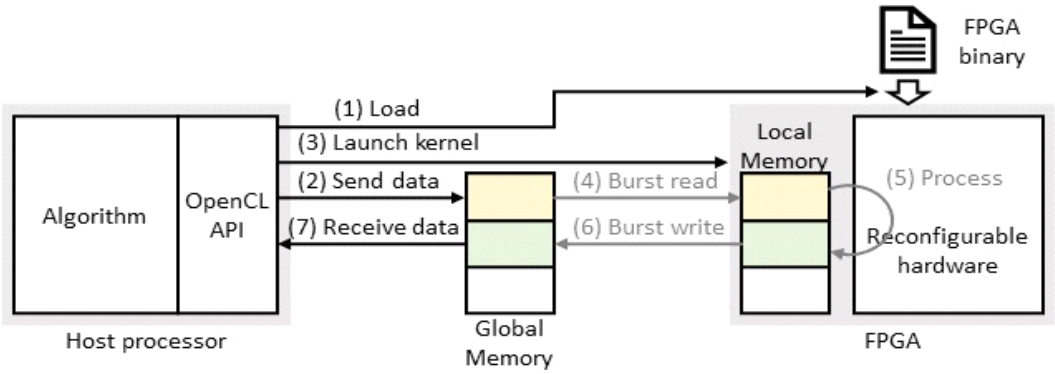


Fig. 7 Execution flow of the software hardware co-design system

나타낸다. 초기 변숫값을 가지는 매개 변수화된 프로그램 코드와 가속기 코드는 Vitis 개발환경에서 실행 파일과 이진 파일로 각각 변환된다. 생성된 응용 프로그램과 가속기는 시뮬레이션 층을 구성하는 서버 환경에서 실행된다. 탐사 층에서는 시뮬레이션의 결과에서 데이터를 추출하여 다음 변수를 결정하고, 생성 층에서는 결정된 변수를 이용하여 새로운 시뮬레이션 코드를 만든다.

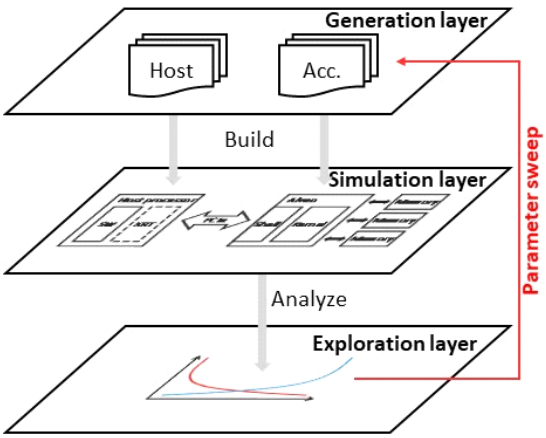


Fig. 8 Concept of proposed platform for accelerator optimization

Ⅳ. 결과 분석

본 논문에서 제안한 플랫폼을 검증하기 위해 각 행과 열의 크기가 1000인 두 개의 행렬 곱셈 연산을 제안한 플랫폼과 하나의 프로세서에서 실행한 결과를 분석하였다. 제안한 플랫폼은 매개변수로 가속기가 한 번에 연산할 수 있는 정방 행렬의 행 크기를 이용하였다. 그림 9는 소프트웨어만으로 행렬 연산을 했을 때의 연산 시간과 행 크기 별 실행 시간을 나타내고, 그림 10은 행 크기별 FPGA의 LUT 사용량을 나타낸다. Intel Xeon Bronze 3204 프로세서에서 실행된 행렬 곱셈 연산은 평균적으로 13.25초가 소요된다. 그림 10의 결과를 보았을 때 가속기의 행 크기가 커질수록 LUT의 개수로 표현되는 하드웨어 크기는 점점 커진다. 그림 9에서 하드웨어의 크기가 커질수록 프로그램의 실행 속도는 점점 빨라지지만, 하드웨어의 크기 증가 대비 실행 속도 감소는 점점 줄어든다.

그림 11은 변수에 따른 전력 소모 예측이다. 전력 소모는 수식 (1)의 방법으로 예측하였다. Alveo U200이 동작할 때 소모되는 평균 전력(P_{HW})은 100W, Intel Xeon 3204 프로세서가 소모하는 평균 전력(P_{SW})은 85W, Alveo U200의 전체 LUT 개수 N은 892,000개이다. 가속기가 소모하는 평균 전력이 LUT의 개수에 비례한다고 가정했을 때 전체 소모 전력은 가속기와 프로세서의 평균 전력을 더한 값에 실행 시간을 곱한 값이다. 예측 결과 하드웨어의 크기가 커질수록 전체 소모 전력은 줄어들지만 크기가 일정 이상이 되면 오히려 전력 소모가 늘어나는 것을 볼 수 있다. 실험 결과에서는 행의 크기가 14일 때 전체 전력 소모가 가장 작다.

$$E_{Exec} = (P_{HW} \times \frac{n_{LUTs}}{N} + P_{SW}) \times t_{Exec} \quad (1)$$

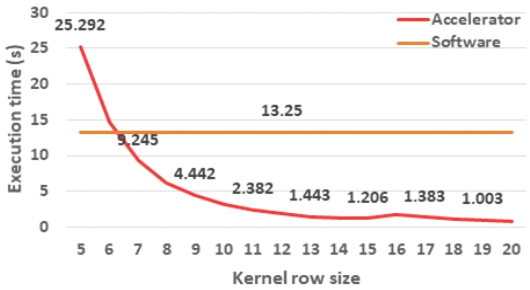


Fig. 9 The program execution time according to the accelerator's row size

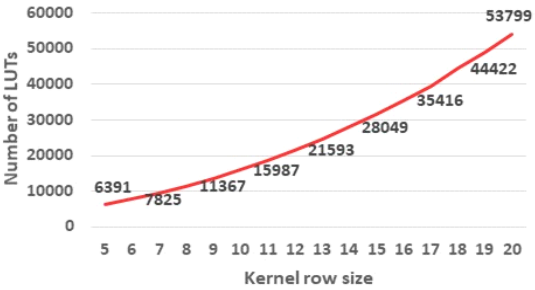


Fig. 10 The number of LUTs according to the accelerator's size

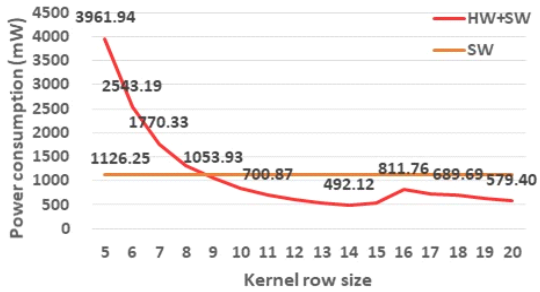


Fig. 11 Power consumption estimation according to the accelerator's size

V. 결 론

본 논문에서는 소프트웨어와 하드웨어 합동 설계에서 에너지 효율적인 하드웨어를 찾기 위한 하드웨어 플랫폼을 제안하였다. 제안한 플랫폼은 하드웨어의 크기를 변수화하고, 변수에 따라 하드웨어를 기술하는 코드와 하드웨어를 실행하는 응용 프로그램의 코드를 반복적으로 재구성 및 실행함으로써 다양한 변수에서의 프로그램 실행 속도와 하드웨어 크기를 결과로 제공한다. 그 결과 프로그램을 실행하는데 필요한 전체 소모 전력을 예측할 수 있어 사용자가 에너지 효율적인 FPGA 하드웨어를 만들 수 있도록 도와준다. 플랫폼에서 설계한 에너지 효율적인 FPGA 가속기의 실행 시간은 응용 프로그램만을 사용했을 때보다 90.7% 감소하였고, 전력 소모는 56.3% 감소하였다. 제안한 플랫폼은 하드웨어와 소프트웨어 코드의 비율을 적절하게 조절해 합동 설계를 효과적으로 할 수 있으므로 다양한 하드웨어 설계 분야에서 적용할 수 있을 것이다. 추후 연구에서는 대규모 데이터를 실시간으로 처리하는 의료 IoT 분야에서 엣지 및 서버의 하드웨어 통합 가속 시스템에 제안하는 하드웨어 합동 설계 플랫폼을 적용할 것이다.

ACKNOWLEDGEMENT

This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science and ICT (NRF- 2019R1A2C2005099), and Ministry of Education (NRF- 2018R1A6A1A03025109), and BK21 Four project funded by the Ministry of Education, and the EDA tool was supported by the IC Design Education Center(IDEC), Korea.

REFERENCES

[1] C. Zhu, V. C. M. Leung, L. Shu, and E. C. -. Ngai, "Green Internet of Things for Smart World," *IEEE Access*, vol. 3, pp. 2151-2162, 2015.

[2] F. Samie, V. Tsoutsouras, L. Bauer, S. Xydis, D. Soudris, and J. Henkel, "Computation offloading and resource allocation for low-power IoT edge devices," *2016 IEEE 3rd World Forum on Internet of Things (WF-IoT)*, Reston, VA, pp. 7-12, 2016.

[3] T. Pflanzner and A. Kertesz, "A survey of IoT cloud providers," *2016 39th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, Opatija, pp. 730-735, 2016.

[4] W. Shi and S. Dustdar, "The Promise of Edge Computing," in *IEEE Computer*, vol. 49, no. 5, pp. 78-81, May. 2016.

[5] F. Ge, N. Wu, H. Xiao, Y. Zhang, and F. Zhou, “Compact Convolutional Neural Network Accelerator for IoT Endpoint SoC,” *Electronics*, vol. 8, no. 5, pp. 497, 2019.

[6] M. Wielgosz and M. Karwatowski, “Mapping Neural Networks to FPGA-Based IoT Devices for Ultra-Low Latency Processing,” *Sensors*, vol. 19, no. 13, pp. 2981, 2019.

[7] D. Lee, J. Cho, and D. Park, “Efficient Partitioning of On-Cloud Remote Executable Code and On-Chip Software for Complex-Connected IoT,” *2019 IEEE International Conference on Big Data and Smart Computing (BigComp)*, Kyoto, Japan, pp. 1-4, 2019.

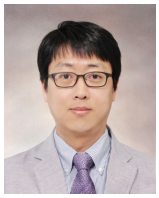
[8] K. S. Chatha and R. Vemuri, “An Iterative Algorithm for Hardware-Software Partitioning, Hardware Design Space Exploration and Scheduling,” *Design Automation for Embedded systems*, vol. 5, pp. 281-293, 2000.

[9] D. Lee, H. Moon, S. Oh, and D. Park, “mIoT: Metamorphic IoT Platform for On-Demand Hardware Replacement in Large-Scaled IoT Applications,” *Sensors*, vol. 20, pp. 3337, 2020.

[10] S. Sinha Roy, F. Turan, K. Jarvinen, F. Vercauteren, and I. Verbauwhede, “FPGA-Based High-Performance Parallel Architecture for Homomorphic Computing on Encrypted Data,” *2019 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, Washington, DC, USA, pp. 387-398, 2019.

이동규(Dongkyu Lee)

2018년: 경북대학교 전자공학부 학사
2019년~현재: 경북대학교 전자전기공학부 석박사통합과정
※관심분야 : 고성능 SoC 설계, 하드웨어-소프트웨어 Co-design, 인공 지능 IoT



박대진(Daejin Park)

2001년: 경북대학교 전자전기공학부 학사
2003년: KAIST 전기 및 전자공학과 석사
2014년: KAIST 전기 및 전자공학과 박사
2003년~2014년: SK Hynix/ Samsung (차세대 LSI 설계) 수석연구원
2014년~2016년: 경북대학교 전자공학부 연구조교수 (2014년 대통령 Postdoctoral Fellow 선정)
2016년~현재: 경북대학교 전자전기공학부 조교수
※관심분야 : 저전력 SoC 설계, 하드웨어-소프트웨어 Co-design, Dependable 스마트 IoT 시스템, Robust 임베디드 시스템