韓國컴퓨터情報學會 論文誌
第15卷 第9號, 2010. 9.

2010-15-9-1-1

CHARMS: A Mapping Heuristic to Explore an Optimal Partitioning in HW/SW Co-Design

Olufemi Adeluyi *, Jeong-A Lee **

CHARMS: 하드웨어-소프트웨어 통합설계의 최적 분할 탐색을 위한 매핑 휴리스틱

아델루이 울루페미 *, 이정아 **

Abstract

The key challenge in HW/SW co-design is how to choose the appropriate HW/SW partitioning from the vast array of possible options in the mapping set. In this paper we present a unique and efficient approach for addressing this problem known as Customized Heuristic Algorithm for Reducing Mapping Sets(CHARMS). CHARMS uses sensitivity to individual task computational complexity as well the computed weighted values of system performance influencing metrics to streamline the mapping sets and extract the most optimal cases. Using H.263 encoder, we show that CHARMS sieves out 95.17% of the sub-optimal mapping sets, leaving the designer with 4.83% of the best cases to select from for run-time implementation.

요 약

하드웨어-소프트웨어 통합설계에서 다양한 설계제약 조건을 만족하는 임베디드 시스템 개발을 효율적으로 완료하기 위하여 하드웨어와 소프트웨어의 최적분할을 빠른 시간 안에 탐색하는 핵심기술이 필요하다. 본 논문에서는 다양한 하드웨어-소프트웨어 분할에 따른 매핑 조합 중 최적분할에 해당할 수 없는 조합들은 미리 선별하여 탐색대상에서 제외하는 것을 가능하게 하는 맞춤형 매핑 휴리스틱, CHARMS을 제시한다. CHARMS은 응용프로그램의 여러 태스크를 하드웨어 또는 소프트웨어로 매핑하면서, 단위시간 안에 처리되는 태스크의 수인 Parallelism과 일의 양인 Workload 로 Throughput을 예측하고 최적의 분할대상을 선별하는 기존의 휴리스틱보다 향상된 방법으로, 태스크들의 계산 복잡도를 고려하였으며, 설계제약 조건의 중요도를 다양하게 표현할 수 있는 weighted combo-metric을 활용한다. H.263 인코더 설계에서 CHARMS을 이용할 경우 매핑조합의 95.17%를 탐색 대상에서 제외할 수 있었음을 실험을 통하여 보였다.

▸ Keyword : hardware-software co-design, heuristic algorithm, embedded systems, partitioning, design space exploration

• 제1저자 : Olufemi Adeluyi 교신저자 : 이정아
• 투고일 : 2010. 07. 27, 심사일 : 2010. 09. 07, 게재확정일 : 2010. 09. 17.
* 조선대학교 컴퓨터공학과 박사과정 ** 조선대학교 컴퓨터공학과 교수
※ This study was supported (in part) by research funds from Chosun University, 2003.

I. INTRODUCTION

The increasing adoption of hardware/software (HW/SW) co-design techniques by embedded system designers has opened up a new frontier that has led to rapid developments within the embedded computing space. Co-design utilizes a collection of techniques for creating efficient application-specific systems that exploit the high performance of hardware as well as the flexibility of software[1].

At some point in the co-design process the system designer has to make a choice regarding the partitioning of the different parts of the application into either the hardware or the software platform. The need to achieve a performance-cost optimal implementation makes the partitioning process very important. The partitioning process produces a mapping set, which gives the flow of the application reflecting which modules are to be implemented in hardware and which modules are to be implemented in software during the live execution of the application. Fig. 1. shows how a mapping function links both application and architecture models.

A typical application implemented on an embedded platform is made up of several tasks, each usually with the option of being implemented either in hardware or software. This scenario can give rise to a large number of mapping cases that a designer can select from. Thus the HW/SW co-design presents the designers with a flexible approach for the live implementation of an application. However, this desirable design flexibility comes with the undesirable consequence of having a large number of mapping sets to choose from. The process of choosing can slow down the system and the designer also risks the possibility of choosing a set that does not give optimal results.

In this paper we present a technique called the Customized Heuristic Algorithm for Reducing Mapping Sets (CHARMS) as a solution to conflicting requirements of flexibility and mapping set optimization. CHARMS is a modification of the Heuristic Algorithm for Reducing Mapping Sets (HARMS) proposed by Ahn et al [2]. It pays attention to the level of complexity of the individual tasks that make up an application. It also considers a number of metrics that affect the performance of the system in the process of streamlining the available mapping sets.

Our CHARMS approach enables designers to narrow down the available mapping sets to a group of cases that represent the best solutions based on the design objectives, which are presented to the CHARMS process as a weighted sum of the performance influencing metrics [3]. CHARMS has been designed to support throughput, power, temperature, area and metricX as design metrics. metricX is a metric the user can define prior to the CHARMS process.

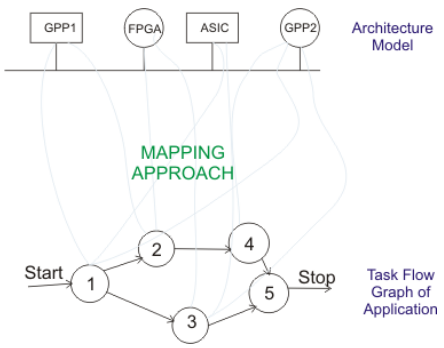

Fig. 1. Linking Application and Architecture Models through a Mapping Approach

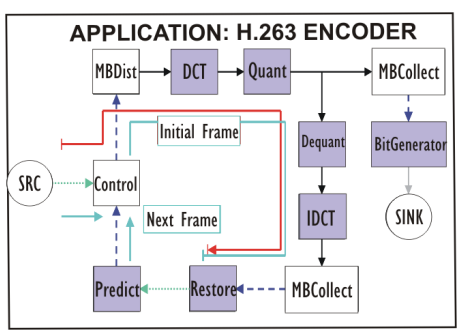

Fig. 2. Task Graph for the H.263 Encoder

All the metrics have user defined weights and metrics that are not of interest are assigned the weights of zero. In this paper we have described the CHARMS process for the implementation of the H.263 encoder. The task graph for the H.263 encoder is shown in Fig. 2.

The remainder of this paper is arranged as follows: the related work is presented in Section II, the description of the CHARMS process is described in Section III and the performance evaluation is presented in section IV. The cohnclusions and future work are then presented in Section V.

II. RELATED WORK

In order to identify best mapping cases in hardware-software co-design, performance metrics can be chosen and their values can be calculated for the generated mapping sets. This procedure, along with the adoption of a Y-chart approach, constitutes a core part of the CHARMS process.

The Y-chart approach [4] generates a performance model based on performance metrics that allow designers to compare models and make decisions motivated by specific goals. The Y-chart approach provides designers a scheme to compare architectural instances based on quantitative data. This Y-chart process is shown in Fig. 3.

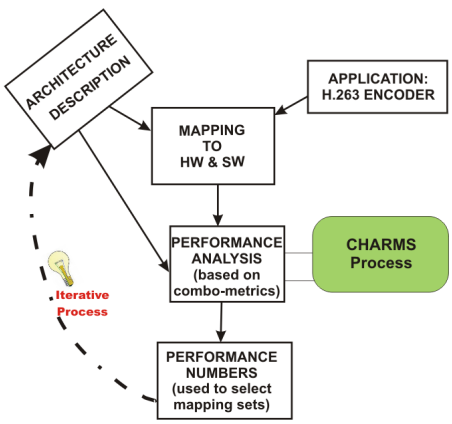

Fig. 3. The Y-chart approach for CHARMS

Embedded system designers regularly use HW/SW partitioning, mapping and related techniques for optimizing their system performance. The HW/SW co-design Architecture Model usually consists of an Application Specific Integrated Circuit (ASIC) or a Field Programmable Gate Array (FPGA) as the hardware platform and a General Purpose Processor (GPP) as the software platform.

Partitioning and Scheduling (P&S) are key procedures in the Y-chart process. The different P&S approaches available can be grouped under many classes. These classes of algorithms include.

(i) Iterative Improvement Algorithms
(ii) Constructive Algorithms
(iii) Genetic Algorithms

Iterative improvement algorithms begin with a P&S solution before iterating to choose solutions with better metrics. Constructive algorithms synthesize the performance incrementally in adherence to a predefined set of rules and are usually effective for large design spaces. Genetic algorithms follow a concept in biology that involves the sets in the solution space evolving or mutating over a period of time to improve the quality of the solutions.

CHARMS is an extension of the, Heuristic Algorithm for Reducing Mapping Sets (HARMS) proposed before [2]. The HARMS streamlines the mapping cases by first excluding impossible mapping cases based on the FPGA size constraint and then uses the values of workload and parallelism as a basis for reducing the mapping sets.

A number of approaches to hardware-software partitioning utilize fixed granularity for the base blocks. Henkel and Ernst departed from this approach by dynamically varying the granularity of the base blocks to reflect the peculiarities of the applications and constraints imposed[5]. For CHARMS we also support the variation of the task granularity.

III. THE CHARMS PROCESS

CHARMS improved on HARMS by introducing a

feature that analyzes the complexity of the various tasks with a view to creating a virtual grouping of blocks and sub-blocks with similar levels of complexity.

CHARMS also uses multiple metrics to determine the quality of a mapping set by computing the weighted values of system performance influencing metrics. This increases design flexibility and gives a more holistic view of the requirements for a mapping set optimization process.

The resulting mapping sets are analyzed in the Y-chart process by being subjected to the analysis of the Y-Sim. The Y-sim simulates the mapping cases by allowing the data units to flow from the beginning to the end of the application in order to extract performance metrics that will be used in the analysis. Y-Sim is the partitioning tool working in tandem with the Y-chart approach. It is designed to perform hardware-software partitioning for reconfigurable systems and it serves as a conflict manager in the process. The Y-Sim is made up of three parts, which is described in detail below.

Architecture Simulator: This is the part that models the hardware resource in question gives a prediction of the expected performance metrics for specific applications that are mapped unto the resource. It receives the data unit from the application simulator via the architecture- application mapping controller. For functional elements that have their exclusion flags set, the architecture simulator prevents conflicts from occurring in scenarios where many subtasks are mapped to the same architecture ID or functional element. The data unit is returned to the processing element of the application simulator from which it originated after the processing of the request and the calculation of the performance metrics. Exclusion flags are associated which GPPs since they do not support parallelism.

Application Simulator: This simulates the data units in the architecture model and controls the flow between the processing elements. The processing elements add their function request to the data units and then send it to the architecture simulator via the architecture-application mapping controller. The processing element that sends the data unit to the architecture simulator receives back the processed data unit and forwards it to the next process. The application simulator also calculates the delay and execution time for processing all the data units.

Architecture-Application Mapping Controller: This serves as an "in-between" for the architecture and application simulators. It generates all the possible mapping cases from the sub-tasks of the application simulator and all the functional elements of the architecture simulator.

1. Task Complexity Sensitivity

HARMS optimizes the available mapping set by using throughput as the constraint. The throughput increases with increased parallelism but reduces with increased workload as shown in Eq. 1. The parallelism reflects the ability of the system to run simultaneous tasks in the application. It is the ratio of the execution time assuming only one task can be run at a time to the execution time as a result of simultaneous execution of tasks. Eq 2 shows how the parallelism is calculated.

$$Throughput \propto \frac{Parallelism}{Workload} \quad \cdots\cdots (1)$$

$$Parallelism = \frac{\tau_{GPP} + \sum_{a=0}^{k} \tau_{FEa}}{t_{end}} \quad \cdots\cdots (2)$$

where τGPP is the time the GPP uses in managing the simulation. τFEa is the time taken for task "a" on the appropriate GPP or FPGA functional element, FE, k is the total number of tasks, tend is the total execution time of the simulation

The values of the workload used for analysis in HARMS are based on the number of tasks processed. The procedure assumes that every task has the same level of complexity.

However, many multimedia applications are said to be stream-based and contain tasks that have a significant variation in their levels of complexity.

By assuming a comparable level of task complexity, the HARMS process precludes some mapping sets, which may potentially provide more optimal solutions. In CHARMS, however, we iteratively sub-partition the tasks with a view to creating a virtual set of tasks with comparable complexity. This process increases the cardinality of the mapping set but creates an equal opportunity for all tasks to constitute the mapping sets and eventually provides us with more optimal and better streamlined mapping cases.

2. Performance Metrics and Flexibility

In many embedded system applications the throughput may not be the factor used in determining the hardware-software partitioning process and where it is a factor it may not be the primary factor. In such a case HARMS and its throughput metric approach will not serve the purpose. We have addressed this in CHARMS by presenting a scenario where the designer can use a primary metric and, if required, several secondary metrics to determine how the mapping set is streamlined.

In CHARMS we have designated the primary metric as the most important factor in the streamlining process. For a design processes that requires the use of more than one metric, we define and use a combo-metric in the streamlining process. The combo-metric is a weighted sum of the contributing metrics, with weights that reflect the level of importance of that metric.

At all times the primary metric is allocated the highest weight. CHARMS allows the designer to define the contributing metrics and their weights. Eq. 3 shows how the combo-metrics are calculated.

$$[combo\_metrics] = \frac{1}{no\_of\_metrics}[metrics][weights]$$
$$= \frac{1}{b}\begin{bmatrix} m_{11}\ldots m_{1b} \\ \ldots\ldots\ldots \\ \ldots\ldots\ldots \\ \ldots\ldots\ldots \\ m_{n1}\ldots m_{nb} \end{bmatrix}\begin{bmatrix} w_1 \\ . \\ . \\ . \\ w_b \end{bmatrix} = \begin{bmatrix} c_1 \\ . \\ . \\ . \\ c_n \end{bmatrix} \quad \ldots\ldots (3)$$

where W, P and T are workload, parallelism and throughput respectively. Also wi is the weight assigned to metric i, cj is the combo-metric for the node j in the application and mnb is the bth metric value for the nth PE node in the FE under consideration.

3. Profiling Process

Short time to market realities have encouraged designers to opt for electronic system level (ESL) design techniques that enable them to explore the design space early in the design process at higher levels of abstraction before being bogged down with the laborious synthesis and verification issues of the lower abstraction levels [6].

The CHARMS analysis relies on the results of the preliminary Y-chart process from such ESL design techniques. CHARMS thus takes advantage of this approach and quickly evaluates the performance of the different application tasks on the available software and hardware platforms. Having a profile database is also important when a designer wants to accommodate the possibility of system reconfiguration at some point in the design cycle.

The profiling process involves the modeling of both the application and the architecture. In this paper the application is the H.263 encoder while the architecture comprises of the hardware (FPGA: Altera Stratix II EP2S60) and the software (Processor: NIOS II softcore processor). The H.263 encoder, like other stream-based applications, is modeled using Directed Acyclic Graphs (DAG) with tasks as the vertices and communication as the edges. Each node in the task graph represents a task that may have moderate to large granularity. The directed edges in DAGs represent data dependencies and/or control dependencies between the tasks. [7].

4. Steps in the CHARMS Process

CHARMS, implemented in C++, utilizes the profile results of the Y-chart process described above. It traces the data from start to end in a way that supports parallelism, taking into account the communication between processes, dependencies, I/O activity, FPGA cell capacity, resource conflict resolution and scheduling.

For simplicity this data-intensive application model uses First Come First Served (FCFS) scheduling. A pre-partition hardware and software timing analysis is carried out and results stored for each task, indicating the execution time when implemented in the FPGA and General Purpose Processor (GPP) respectively. Timing analysis is done in order to calculate the throughput and parallelism.

CHARMS process is briefly given below:

(a) Arrange the "n" tasks in an increasing order of the combo metric under consideration, combo_metric, from combo_metric(1) for task with lowest combo metric value to combo_metric(n) for task with the largest combo metric value

(b) Select a partition set candidate node A, where task_i in A is such that the combo_metric(task_i)〉〉 average combo_metric over all the tasks

(c) Partition this task_i in A obtained from (b) above into two subtasks task_i1 and task_i2. The new values of the combo_metric in i1 & i2 should be as close as possible to each other, preferably half of the original value, subject to the availability of branching points within the program

(d) Re-execute the code sequentially to get new mapping sets

(e) Repeat the steps above, selecting the next largest task as required, until the nodes in the task graph have similar values of the combo_metric or until streamlined mapping set includes desired combo-metric cases. The designer can specify the quality of desired mapping cases based on the combo-metric and CHARMS checks this requirement after each iteration and will only proceed with further iteration if this requirement has not been met.

IV. PERFORMANCE EVALUATION

This section describes the process of implementing a H.263 encoder with both the HARMS and the CHARMS and gives some experimental results.

1. Design Flexibility with Combo-Metric

The CHARMS process bases its decision of the best mapping cases on a weighted combination of metrics, which we refer to as combo-metrics. These metrics, priority levels and default weights are shown in Table 1.

Although we do not directly compare HARMS with CHARMS as it relates to the flexible combo-metric approach in this paper, it is worth noting that CHARMS processes an iteration for the same duration as HARMS. As such, being able to select mapping sets based on multiple metrics, rather than just one metric, without an extra timing overhead presents designers with a flexible approach for sieving out the best mapping cases for use at runtime.

2. Profile Results for H.263 Encoder Algorithm

In this section we compare the CHARMS and HARMS using throughput as the only metric. For conformity with the HARMS analysis, we have set the weight of the throughput metric in CHARMS to "1" and have set all the other metrics to "0".

The throughput profile results of the original 11 nodes of the H.263 encoder (Fig. 2) are shown in Table 2. Using the Y-chart process and CHARMS the encoder is run with a source video file (SRC) to result in an encoded file (SNK). The MBCollect refers to the frame storage used as a reference for the next encoding process or as input to the BitGenerator.

Other tasks include the quantization (Quant), dequantization (Dequant), Discrete Cosine Transform (DCT), Inverse DCT (IDCT) and the others which are used for motion estimation and compensation. Each of the 11 tasks are run on both

the software and hardware platforms to extract timing results with which the throughput is estimated using Eq. 1 & 2. As expected the results are better in HW.

Table 1. Metrics used in CHARMS

S/No	Metric	Priority	Weight
1.	Throughput	Primary	5
2.	Power	Secondary	4
3.	Temperature	Tertiary	3
4.	Area	None	2
5.	*metricX*	None	0

Table 2. Throughput Profile for H.263 Encoder Tasks

Processing Element	SW FE: (NIOS II Processor)	HW FE: (Altera Stratix II EP2S60)
Control	360	65682
MBDist	660	157637
DCT	936	166568
Quant	25266	1874821
Dequant	33107	1403309
IDCT	1205	132514
MBC Collect1	841	197047
Restore	3362	481669
Predict	1465	367820
MBC Collect2	841	197047
Bit Generator	931	153258

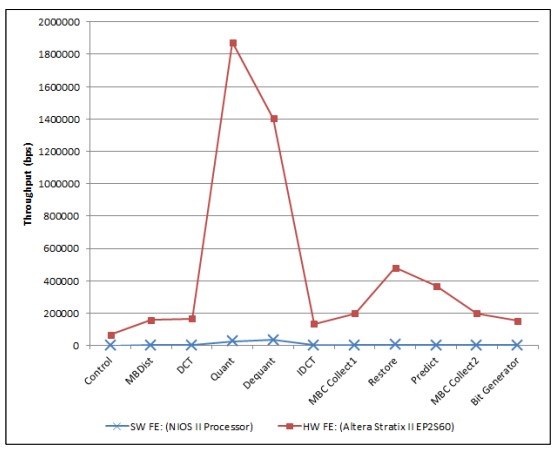

Fig. 5. Comparison of the throughput values for the architecture processing elements

3. Effect of CHARMS on the H.263 Encoder

For the analysis we have run both algorithms (CHARMS and HARMS) thrice in succession. HARMS was able to discard 956 mapping cases of [2], while CHARMS was able to discard 3,898 cases of the total possible 4,096 cases, thus representing an efficiency of 95.17%.

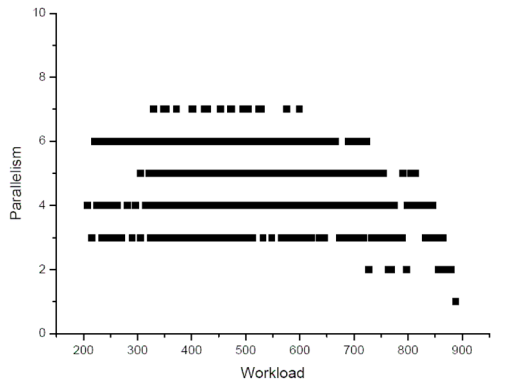

Fig. 6. Graph showing mapping sets from which HARMS extracts cases

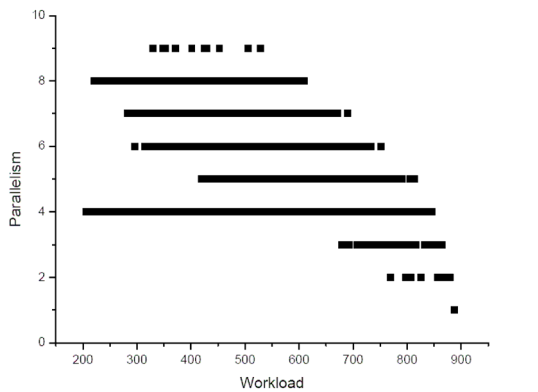

Fig. 7. Graph showing mapping sets from which CHARMS extracts cases

The calculation of the efficiency of the streamlining process, η, is given below.

$$\eta = \frac{(Total\ Mapping\ Sets) - (Streamlined\ Mapping\ Sets)}{Total\ Mapping\ Sets}$$

As shown in Fig. 6. and Fig. 7., both HARMS and CHARMS select mapping cases from the total mapping cases. The figures show the penultimate cases just prior to the final extraction of the best cases. Many of the cases have very similar characteristics hence the appearance of some continuous lines. Also we invite the readers to observe that with CHARMS we get some more mapping cases in its penultimate stage than we do with HARMS. This is because the

CHARMS process leads to finer task granularity. Similarly, the readers should observe that CHARMS mapping cases have higher levels of parallelism (and by effect, higher cases of throughput) than the HARMS cases.

V. CONCLUSION AND FUTURE WORK

In this paper we have presented the Customized Heuristic Algorithm for Reducing Mapping Sets (CHARMS) as an improvement of the Heuristic Algorithm for Reducing Mapping Sets (HARMS). The extensions have been based on the use of multiple metrics (combo-metrics) for increased design flexibility and the sensitivity to individual task complexity in order to create a level playing field for the tasks in their bid to constitute mapping cases. The result has been increased design flexibility, improvement in the efficiency of the streamlining process and the availability and choice of mapping cases of higher quality.

For future work we plan to test the CHARMS approach with a broader suite of applications.

REFERENCE

〔1〕 De Micheli, G. Ernst, R. and Wolf. W., "*Readings in Hardware/software co-design*," Morgan Kaufmann Publishers, 2002.

〔2〕 Ahn S., Kim J., Lee Jeong-A. "Heuristic Algorithm for Reducing Mapping Sets of Hardware-Software Partitioning in Reconfigurable System," Asia-Pacific Computer Systems Architecture Conference, 2004.

〔3〕 Kienhuis, A.C.J., "Design Space Exploration of Stream-based Dataflow Architectures," Ph D Thesis, Delfts University of Technology, 1999.

〔4〕 Adeluyi O., "A Co-Design Approach for Embedded Multimedia Applications: a case study of H.264," MSc Thesis, Chosun University, Korea, June 2009.

〔5〕 Henkel J. and Ernst R., "An Approach to Automated Hardware/Software Partitioning using a Flexible Granularity that is Driven by High-Level Estimation Techniques," VLSI Systems, IEEE Transactions, April 2001.

〔6〕 Z. Luke and Z. Raida, "Multi-objective Optimization of Wire Antennas: Genetic Algorithms versus Particle Swarm Optimization," Radio Engineering, Vol. 14, No 4, Dec 2005.

〔7〕 Beuxa S., Boisa G., Nicolescua G., Bouchebabab Y., Langevinb M. and Paulinb P., "Combining mapping and partitioning exploration for NoC-based embedded systems," Special Issue on HW/SW Co-Design: Systems and Networks on Chip, Journal of Systems Architecture, Volume 56, Issue 7, pp. 223-232, July 2010.

〔8〕 이지근, 김명훈, 이상설, 정성태, "Design of an Efficient VLSI Architecture and Verification using FPGA -implementation for HMM(Hidden Markov Model) -based Robust and Real-time Lip Reading," 한국컴퓨터정보학회논문지, 제 11권, 제 2호 159-167쪽, 2006년 5월.

〔9〕 Si-hyun Lee, "The Design and implementation of parallel processing System using the Nios(R) II embedded processor", 한국컴퓨터정보학회논문지, 제 14권, 제 11호, 97~103쪽, 2009년 11월.

저 자 소 개

Olufemi Adeluyi

2001 : Obafemi Awolowo University 공학사

2007 - 현재 : 조선대학교 컴퓨터공학과 석사 (09), 박사과정

관심분야 : HW/SW co-design, bio-inspired computing

이 정 아

1982 : 서울대학교 공학사

1985 : 인디애나대학교 공학석사

1990 : UCLA 공학박사

1995 - 현재 : 조선대학교 컴퓨터공학과 교수

1990 - 95 : Univ. of Houston 조교수

관심분야 : Configurable Computing