TEC | Tecnológico de Costa Rica

# Project Plan

## Daniel Moya Sánchez

May 4, 2018

# 1   Name of the project

Design of Application Specific Instruction Set Processors (ASIPs) for Approximate Computing

# 2   Name of the institution

- Chair for Embedded System (CES), Kalrsruhe Institute of Technology (KIT), Germany, and

- Laboratorio Sistemas Embebidos y Electrónica Digital (SEED-Lab) of Instituto Tecnológico de Costa Rica (ITCR).

# 3   Confidentiality requirements

Due to the academic nature of this project, there are no special confidentiality requirements. However, results will not be published until the end of the project's work.

# 4   Problem description

In the current era where sophisticated applications are widely used (e.g. GPS systems, speech recognition, etc.) approximate computing helps delivering acceptable output while keeping metrics such as response time or energy efficiency at better levels. Approximate computing gives the freedom to tradeoff certain error level or allow quality degradation in the final output of an application (e.g. noise in the output signal) for lower energy consumption, area or execution time, thus giving the researcher a tool to adjust with the real and specific needs of a given application. For a given error-tolerant system, the framework on figure 1 can be applied to include the approximate computing paradigm [1].

The key elements of figure 1 consist of approximate kernels, which are the implementation (techniques) of the approximate functions, these could be done at a hardware layer or at a software layer; the identification of the error-tolerant parts and its specific details (e.g. impact analysis); and the quality management which implies a continuos evaluation to determine if the application meets the desired requirements [1].

One way to implement approximate computing is through ASIPs. An ASIP is a processor that uses an application-specific instruction set, this means that, although it can execute a wide range of applications, it is optimized for a specific one, in which the ASIP can execute with improved performance (for instance, energy consumption or execution time would be lower) compared to a General Purpose Proccesor (GPP). With the use of ASIPs, instructions
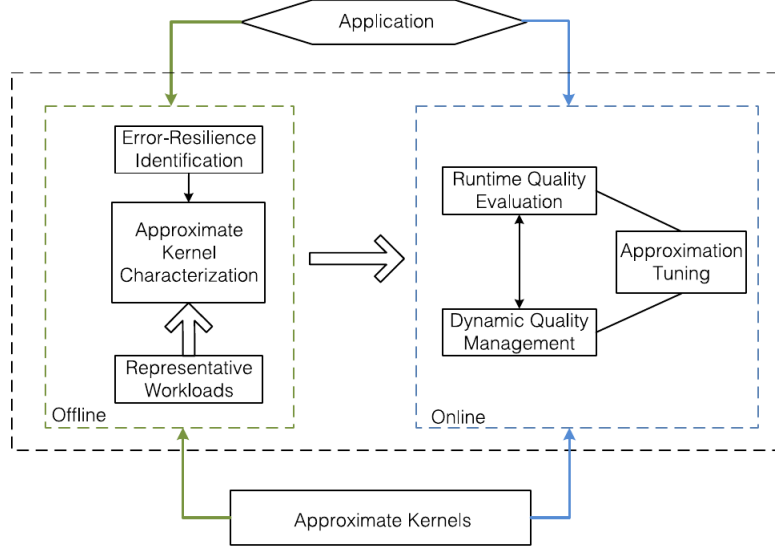
Figure 1: An existing approximate computing framework [1]

and even functions where there is error tolerance can be implemented as special approximated instructions, that reduce the resource consumption while keeping the error from the approximation under an acceptable threshold. Although Application Specific Integrated Circuits (ASICs) show better performance results, ASIPs possess more flexibility. Optimizations for an ASIP can be seen in different forms, including [2]:

- Instruction extension: Customized instructions can be made to extend the base Instruction Set Architecture (ISA).

- Inclusion or exclusion of predefined blocks: Not only specific software can be added to extend an architecture but also customized hardware in the form of specialized blocks; also, regular blocks not used can be excluded.

- Parameterization: Certain variables, such as cache sizes or number of registers, can be customized to adjust for a specific application.

ASICs represent a hardware solution to a problem which is very limited and have high costs and a high time-to-market, but achieve the greatest performance. Contrary, GPPs are seen as a software solution which are very flexible but they are the least efficient. ASIPs are in the middle of these two as they balance flexibility and performance to have a good trade-off between those variables.

The relationship between GPPs, ASIPs, and ASICs is shown in figure 2. Approximate computing can also be implemented on GPPs, due to their extremely flexible nature, however, because they are designed for any kind of computation, it falls on the programmers, or compilers, not on specialized hardware modules, to make the performance of software running
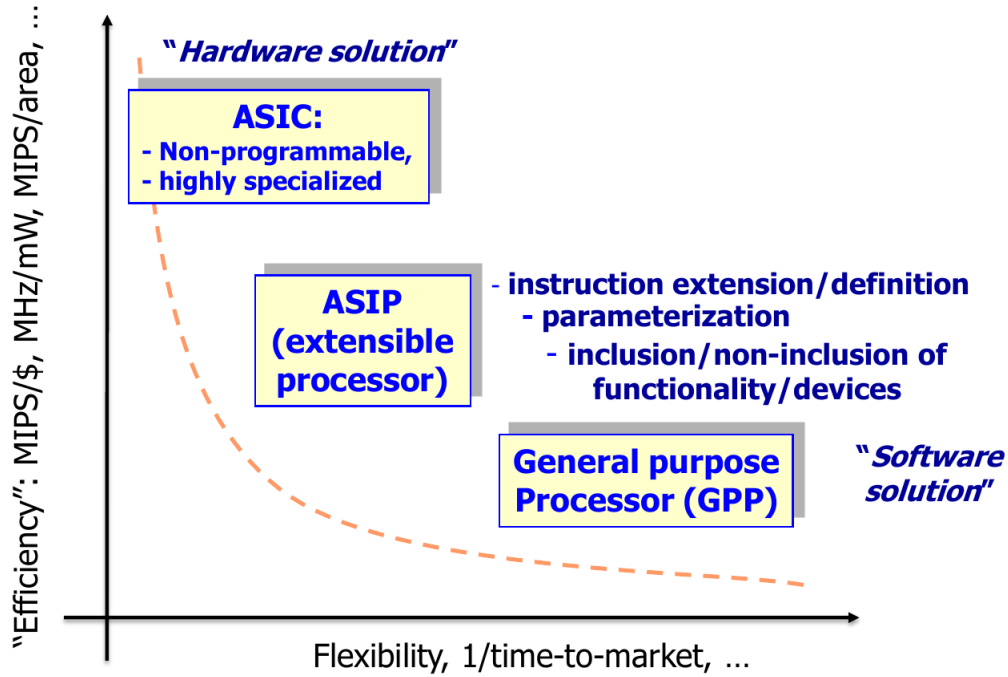
Figure 2: Comparison between GPPs, ASIPs, and ASICs [3]

on these systems as high as possible. On the other hand, since ASICs are a pure hardware solution, approximate hardware modules would be difficult to manage in terms of quality evaluation, this could cause high cost of development, since the hardware would not be programmable. ASIPs can adjust to specific requirements of a given application (through extended instructions) so that a better balance of cost savings and amount of error is achieved. This project focuses on that goal; to design ASIPs for a set of error-tolerant applications.

The environment in which the ASIPs will be developed consists of several software tools which include Design Compiler and Prime Time from Synopsys, ModelSim from Mentor Graphics, ASIPMeister, CoSy compiler, Xilinx ISE and the hardware platform will be a Xilinx Virtex-V board. Despite these restrictions, several different hardware designs can be used for specific application sections; as well as the implementation of said applications.

Since approximated computing is still in its infancy, a lot of research and testing is still needed, so the users of the developed ASIPs are the same research groups of which this project is a part of. This project is expected to help make approximated computing a more widespread tendency.

4

# 5 Objectives

## 5.1 General objectives

Evaluate the design of Application-Specific Instruction Set Processors (ASIPs) for error-tolerant applications regardless of the specific application area.

## 5.2 Specific objectives

This project has the following specific objectives:

1. Select 3 error-tolerant applications to be evaluated.

2. Develop, for each application, at least 1 instance of approximated hardware for error-tolerant sections.

3. Develop ASIP configurations using specific approximated instructions for the selected applications.

4. Identify how much a certain application can be optimized with the use of ASIPs to balance resource consumption and output error

# 6 Project stakeholders

Due to this project belonging to a research project, there are only a few stakeholders, who are described below:

- Jorge Castro: He is the project's supervisor and he has the general idea about the project itself and guides its course. He attempts to create new knowledge with the use of ASIPs for error-tolerant applications, using approximate computing techniques, and that their design become automated.

- Sajjad Hussain: He works with Jorge Castro on the general guidance of the project. He supports any issue with the tools in Germany so that the process of using the developing platform (ASIPMeister, Dlxsim, etc.) remains smooth. He has the same interest as Jorge Castro regarding the project.

- Jeferson González: He is the project's supervisor at the ITCR, and the person in charge of the SEED laboratory, from where he occasionally provides guidance and collaboration (such as lab equipment).

- ASIP users: Future users of the knowledge produced by this research.

# 7   Solution description

First, an application that can have an approximated behavior in any of its steps needs to be found, for this, several applications have to be examined and each needs to have source code that, without modifications, executes correctly in standard hardware blocks. Next, the selected application must be studied to determine whether an entire section can be approximated, only a certain operation can be approximated (e.g. a matrix multiplication), or if both can be approximated. Figure 3 shows a generic system where the latter is true.
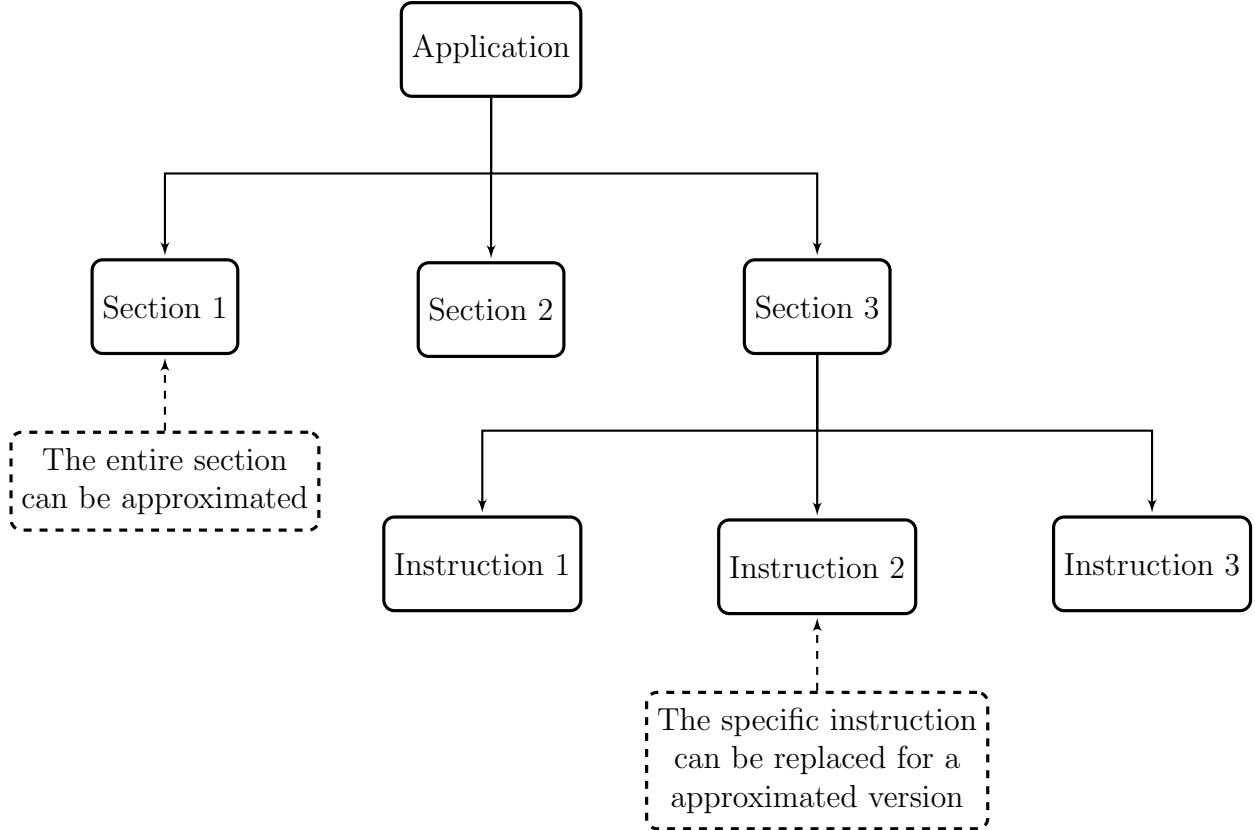


Figure 3: A possible situation to solve with this project

As seen in figure 3, that example of an application has three sections, from which the first one (this could be a preprocessing stage) can be entirely approximated, the second one cannot be approximated at all (we can think of this as a critical section of the application) and finally, in the third second, three specific instructions can be seen, from which the second one has a approximated version. The specific characteristics of the final applications selected have to be determined to execute an analysis similar to this one presented.

Once the approximate parts have been selected, the process of creating the ASIPs begin. These ASIPs are continuously tested to ensure that the application does not exceed a certain error threshold, but also, that a greater performance in energy, area or execution time is achieved compared to the original version.

Other solutions have been proposed, which include randomness in programs and inference

via probabilistic programming as software solutions; approximate computing with GPPs (as discussed in the problem description section) as a different processor architecture solution; improvements in the memory, storage, and interconnections for simpler circuits and finally neural accelerators as a different approximate computing paradigm [1].

# 8 Deliverables and criteria of acceptance

The expected deliverables are presented in table 1.

Table 1: Deliverables with the corresponding criteria of acceptance

| Name | Description | Criteria of acceptance |
|---|---|---|
| Deliverable-01 | List of selected approximate applications | Approval given by the supervisor |
| Deliverable-02 | Instances of approximated hardware | Approval given by the supervisor |
| Deliverable-03 | Configuration of approximated ASIPs | Approval given by the supervisor |
| Deliverable-04 | Comparison and analysis of obtained results (execution time, area and power vs error) | Execution of the test plan with satisfying results |
| Deliverable-05 | Project Plan document | Specifications given by the professor for this document |
| Deliverable-06 | Requirements document | Comply with the ISO/IEC/IEEE 29148:2011(E) standard. |
| Deliverable-07 | Design document | Comply with the ISO/IEC/IEEE 1016 standard and fulfill the requirements document |
| Deliverable-08 | Test plan document | Comply with the requirements document and approval given by the supervisor |
| Deliverable-09 | Final report | Specifications given by the professor for this document |

# 9 Risk analysis

Since most of the work is done from home (*ssh* to the Germany server) or at the SEED laboratory, few risks are considered. Table 2 summarizes this information.

Table 2: Risk analysis

| Risk | Type | Probability of occurrence | Impact (hours) | Risk exposure (hours) |
|---|---|---|---|---|
| Illness or any special medical condition | Personal | 0.5 | 8 | 4 |
| Difficulties understanding ASIP-related concepts | Personal | 0.5 | 8 | 4 |
| General server errors (missing files, permission restrictions, etc) | Tools | 0.75 | 24 | 18 |
| Delays when acquiring the hardware platform | Tools | 0.25 | 8 | 2 |
| ASIP configurations that exceed error threshold | Methods | 0.75 | 8 | 6 |
| Delays when acquiring the server environment set | Inputs | 0.75 | 8 | 6 |

# 10  Activities and effort budget

This section takes in consideration a total of 216 engineering hours; this is calculated by multiplying the 16 weeks by 12 extra-class hours, and then adding 4 hours times 6 because of the 6 weeks that there is no class attendance. These hours are then distributed among all the tasks, considering a risk reserve. Table 3 summarizes all the activities for the project.

# 11  Schedule

Considering the 4 months (16 weeks) of the semester, the project is scheduled as shown in figure 4, where the light blue color represents the initial tasks for the project, blue the development of the project itself and gray documentation tasks.

# References

[1] Qiang Xu, Todd Mytkowicz, and Nam Sung Kim. Approximate computing: A survey. *IEEE Design & Test*, 2018.

[2] Jörg Henkel. Closing the soc design gap. *Computer*, 36(9):119–121, 2003.

[3] Jörg Henkel. Design and architectures for embedded systems (esii). 2006.

Table 3: Activities and effort budget

| ID | Activity | Engineering hours | Risk reserve (hours) | Total (hours) |
|---|---|---|---|---|
| 01 | Get to know the software platform | 30 | 2 | 32 |
| 02 | Find appropiate error-tolerant applications and identify the sections that can be approximated | 30 | 2 | 32 |
| 03 | Implement the ASIPs in the error tolerant applications found | 60 | 4 | 64 |
| 10 | Implement two small assembly codes for each special instruction developed (one with the special instruction and the other without it) for comparison | 14 | 2 | 16 |
| 09 | Compare execution time, area and power vs error in selected applications | 30 | 2 | 32 |
| 04 | Write Project Plan document | 8 | 0 | 8 |
| 05 | Write Requirements document | 8 | 0 | 8 |
| 06 | Write Design document | 8 | 0 | 8 |
| 08 | Write Final documentation | 15 | 1 | 16 |
| Result | | | | 216 |

Table 4: Schedule for the entire project

| Activity | Week | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
| Get to know the software platform | ■ | ■ | ■ | | | | | | | | | | | | | |
| Write Project Plan document | | ■ | | | | | | | | | | | | | | |
| Write Requirements document | | | ■ | | | | | | | | | | | | | |
| Write Design document | | | | ■ | | | | | | | | | | | | |
| Find appropiate error-tolerant applications and identify the sections that can be approximated | | | | ■ | ■ | ■ | | | | | | | | | | |
| Implement the ASIPs in the error tolerant applications found | | | | | | | ■ | ■ | ■ | ■ | ■ | ■ | | | | |
| Implement two small assembly codes for each special instruction developed | | | | | | | | | | | ■ | ■ | ■ | | | |
| Compare execution time, area and power vs error in selected applications | | | | | | | | | | | | ■ | ■ | ■ | ■ | |
| Write Final documentation | | | | | | | | | | | | | | | ■ | ■ |

10