Instituto Tecnológico de Costa Rica
Área Académica de Ingeniería en Computadores

Proyecto de Diseño en Ingeniería en Computadores

Project Plan

Daniel Moya Sánchez

February 16, 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 However, results will not be published until the end of the project's work.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

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) compared to a General Purpose Proccesor (GPP). Although Application Specific Integrated Circuits (ASICs) present better performance results, ASIPs possess flexibility. Optimizations for an ASIP can be seen in different forms, including [CITA]:

- 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

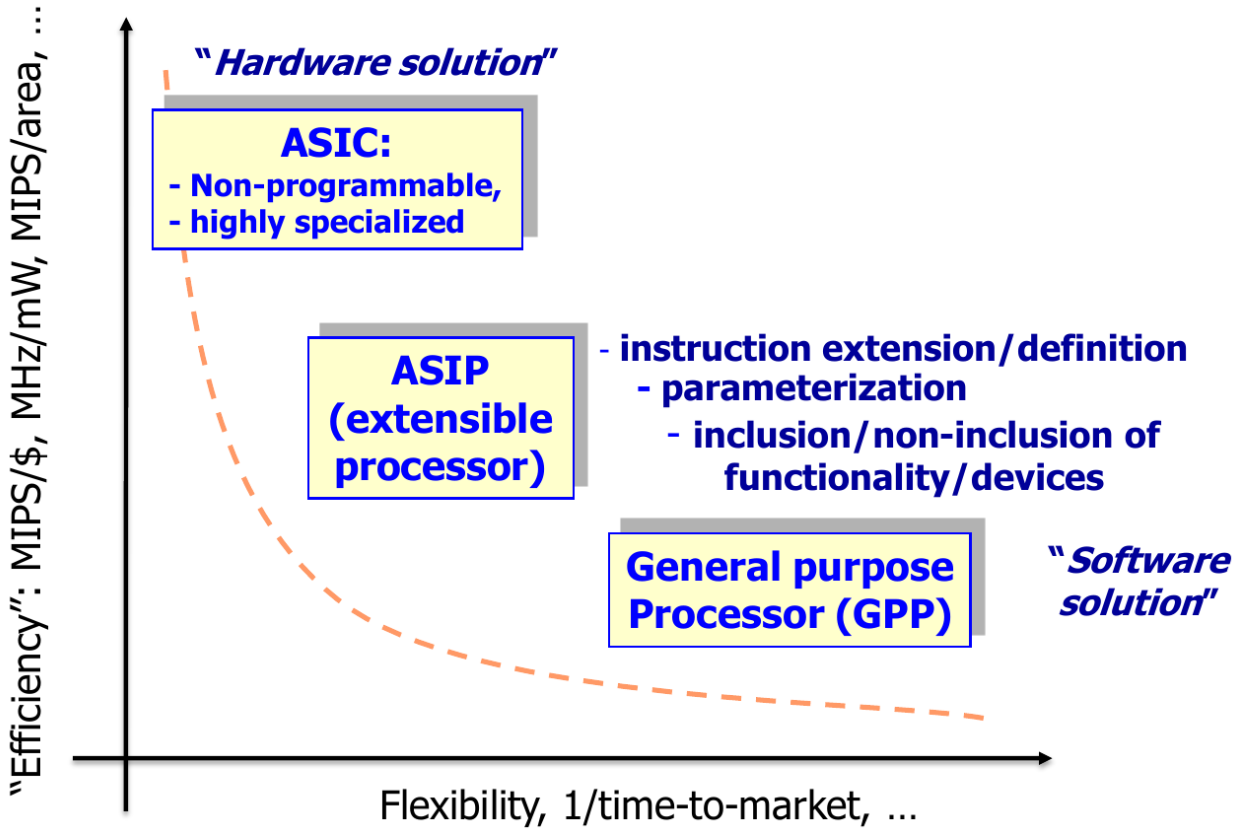

Figure 1: Comparison between GPPs, ASIPs, and ASICs

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.

ASIPs can also be used to adjust the balance between acceptable amount of error vs. the cost (economic, area, execution time) of an application; which is is the main focus of study of the project. Since different types of applications vary significantly in their requirements and specifications (e.g. where the error-resilient section is), different ASIPs have to be build for each one of them so that the best balance between 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. Regardless of these limitations, the project will allow for a custom hardware components choice with its design for specific sections. Also, for the error tolerant applications found, the software implementation and the tests for the general system will be able to be chosen between different options.

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 solid tendency.

5 Objectives

5.1 General objectives

Explore the design of Application-Specific Instruction Set Processors (ASIPs) for error-resilient applications.

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. Evaluate if the gain of the execution time, area, and power consumption is worth the error introduced by the approximate instructions.

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).

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 2 shows a generic system where the latter is true.

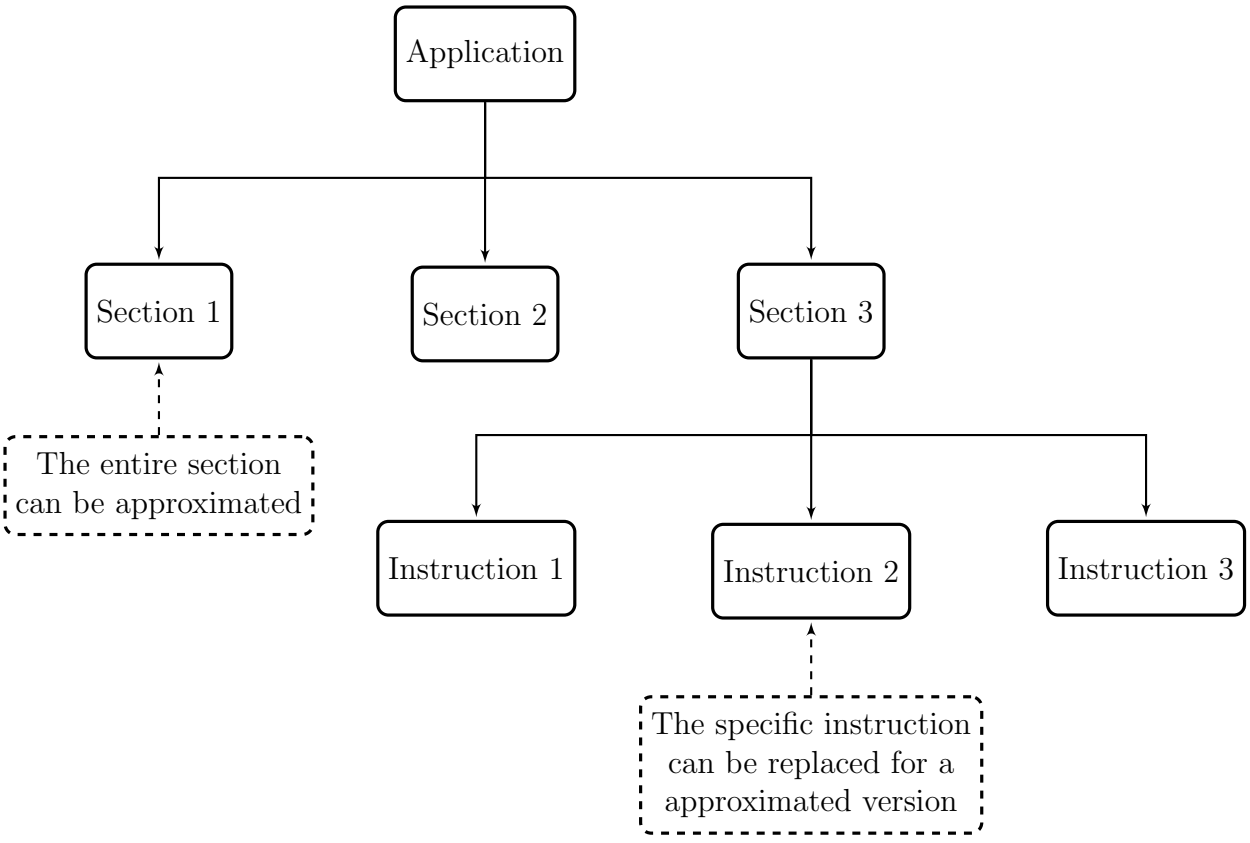

Figure 2: A possible situation to solve with this project

As seen in figure 2, 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, inference via probabilistic programming as software solutions; approximate computing with GPPs (as discussed in the problem description section) as a different architecture processor solution, improvements in the memory, storage, and interconnections for simplier circuits and finally neural accelerators as a different approximate computing paradigm. [CITA SURVEY]

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
Requirement 1.1	Instances of approximated hardware	[criterio]
Requirement 2.1	Configuration of approximated ASIPs	[criterio]
Requirement 3.1	Comparison and analysis of obtained results (execution time, area and power vs error)	Execution of the test plan with satisfying results
Requirement 4.1	Project Plan document	Specifications given by the professors for this document
Requirement 5.1	Requirements document	Specifications given by the professors for this document
Requirement 6.1	Design document	Specifications given by the professors for this document
Requirement 7.1	Test plan document	Specifications given by the supervisor for this document
Requirement 8.1	Final documentation	Specifications given by the professor for this document

9 Risk analysis

Since most of the work is done from home or at the SEED laboratory, few risks are considered. Table 2 summarizes this information.

Table 2: Risk analysis

Risk	Probability of occurrence	Impact (hours)	Risk exposure (hours)
Illness or any special medical condition	0.5	8	4
General server errors (missing files, permission restrictions, etc)	0.6	24	14.4
Delays when acquiring the hardware	0.25	8	2

10 Activities and effort budget

This section takes in consideration a total of 216 engineering hours; this is then distributed among all the tasks, considering a risk reserve. Table 3 summarizes all the activities for the project.

Table 3: Activities and effort budget

ID	Activity	Engineering hours	Risk reserve (hours)	Total (hours)
01	Get to know the software platform	24	2	26
02	Find appropiate error-resiliant applications and identify the sections that can be approximated	42	4	46
03	Implement the ASIPs in the error tolerant applications found	50	3	53
04	Redact Project Plan document	35	3	33
05	Redact Requirements document	35	3	33
06	Redact Design document	35	3	33
07	Redact Test plan document	35	3	33
08	Redact Final documentation	35	3	33

11 Schedule

Considering the 4 months (16 weeks) of the semester, the project is scheduled as shown in figure 4.

Table 4: Schedule for the entire project

	Week															
Activity	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Reading the corresponding literature about the project and understanding the concept of ASIPs																
Execution of the laboratory script to get to know the software tools like ASIPMeister, Dlxsim, etc																
Delivery of the "Plan Project" document																
Delivery of the "Requirements" document																
Delivery of the "Design" document																
Research of error-tolerant applications with desirable features																
Design of blocks with special approximated instructions																
Putting the blocks together to build the ASIPs																
Final results comparison																
Delivery of the "Final report" document																

References