Instituto Tecnológico de Costa Rica

Área Académica de Ingeniería en Computadores
(Computer Engineering Academic Area)

Programa de Licenciatura en Ingenería en Computadores
(Licentiate Degree Program in Computer Engineering)

Diseño asistido de aplicaciones aproximadas para sistemas computacionales personalizables

(Assisted-Design of Approximate Applications for Customized Computational Systems)

Informe del Anteproyecto para el Trabajo Final de Graduación
(Report of Pre-project for a Graduation Work in fulfillment of the requirements for the degree of Licentiate in Computer Engineering)

Daniel Esteban Moya Sánchez

Cartago, Mayo, 2018

Índice

1. Palabras Clave

Computación aproximada, arquitectura heterogénea, caracterización, aceleradores aproximados.

2. Introducción

Los sistemas de Tecnologías de Información (TI) buscan dar una mejor calidad de vida a las personas. En esta tarea, estos sistemas han tenido que enfrentar ciertos problemas, entre ellos el costo en área, potencia y tiempo de ejecución, los cuales restringen el rendimiento de un chip. Idealmente, una aplicación debe ajustarse a las necesidades reales del usuario y, en general, del área de aplicación, de forma que se dé un uso óptimo de los recursos. Actualmente, el diseño de procesadores no solo se enfoca en contar con más desempeño si no en tener un manejo de recursos apropiado. No obstante, algunos desafíos en este campo están dados por limitaciones físicas, por ejemplo:

- las características eléctricas de los transistores CMOS, las cuales restringen el consumo de energía en sistemas embebidos y lo cual es un aspecto que deben considerar los diseñadores de componentes para propósito específico en procesadores;
- la pared de memoria, que corresponde a la diferencia entre el crecimiento de la capacidad de procesamiento contra la velocidad de obtención de datos desde la memoria;
- y la pared de utilización, la cual limita el uso máximo de hardware simultáneamente debido a las capacidades de disipación de calor de un sistema.

Para poder enfrentar los problemas mencionados anteriormente, un área de investigación actual corresponde a *computación aproximada*, un paradigma de diseño que propone una reducción en la precisión o exactitud de la computación para obtener oportunidades de mejora en cuanto al consumo de área, potencia y tiempo de ejecución. Para aplicar dicho paradigma es necesario identificar aplicaciones tolerantes a errores y determinar, más específicamente, cuáles secciones o funciones dentro de estas pueden ser sustituidas por versiones aproximadas, de forma que se pueda generar un balance entre la calidad de la salida y el consumo general de recursos.

Este documento busca explicar los detalles relacionados a la propuesta de un anteproyecto para el diseño asistido de aplicaciones aproximadas en sistemas computacionales personalizables, considerando una plataforma con multiples aceleradores de hardware disponibles. Con la realización del proyecto se espera constribuir a la investigación en el campo de la computación aproximada, especialmente en el Instituto Tecnológico de Karlsruhe en Alemania, así como en el de área de ingeniería en computadores en general.

En la siguiente sección se presenta el contexto y los antecedentes del proyecto, que incluye una descripción de la institución donde se realizará el trabajo final de graduación y el área de conocimiento específica de este. Seguidamente, se realiza la justificación del problema, la especificación de los objetivos, la mención de los beneficios y beneficiarios con los resultados del

proyecto, los supuestos y limitaciones sobre los que parte el proyecto y un análisis de riesgos del mismo. Finalmente, se detalla la metodología que se seguirá en el proyecto, la cual incluye la tipificación del trabajo, descripción de las tareas que se realizarán, las herramientas que se utilizarán, los entregables que se fijan para cumplir con los objetivos propuestos, las estrategias de verificación y validación que se seguirán y, por último, el cronograma de trabajo propuesto.

3. Contexto y Antecedentes

3.1. Descripción de la Institución

El Instituto Tecnológico de Karlsruhe (KIT) surge en 2009 a partir de la fusión de la Universidad de Karlsruhe, fundada en 1825 como Universidad Fridericiana, y el Centro de Investigación de Karlsruhe. Se ubica en Karlsruhe, Baden-Württemberg, al suroeste de Alemania.

El KIT es una de las universidades técnicas más prestigiosas de Alemania, la cual se especializa en ciencias de la ingeniería. Según [1] para el 2017 contó con 25.495 estudiantes y 9.297 empleados. De acuerdo con [2] el KIT está divido en cinco divisiones:

- División I: Biología, Química e Ingeniería de Procesos.
- División II: Informática, Economía y Sociedad.
- División III: Ingeniería Mecánica y Eléctrica.
- División IV: Ambiente Natural y Construido.
- División V: Física y Matemática.

Las divisiones trabajan en aspectos de investigación, enseñanza e innovación. Los programas de investigación se organizan en programas Helmholtz, donde se le da apoyo a las investigaciones multidisciplinarias. Los departamentos en el KIT son los responsables de la educación universitaria. La Figura 1 resume la organización que posee el KIT en el campo de ciencia.

El Instituto de Ingeniería en Computadores del KIT incluye grupos de trabajo que abarcan los diferentes niveles de abstracción de sistemas computacionales. En el *Chair for Embedded Systems* (CES) se investigan diversos aspectos relacionados con el diseño de sistemas embebidos, desde la confiabilidad de circuitos hasta el manejo de potencia en sistemas multinúcleos.

El presente proyecto será desarrollado en el CES bajo la dirección del M.Sc. Jorge Castro-Godínez, ingeniero en electrónica, investigador y estudiante de doctorado, quien es egresado del Tecnológico de Costa Rica y posee más de tres años como investigador en el Instituto Tecnológico de Karlsruhe.

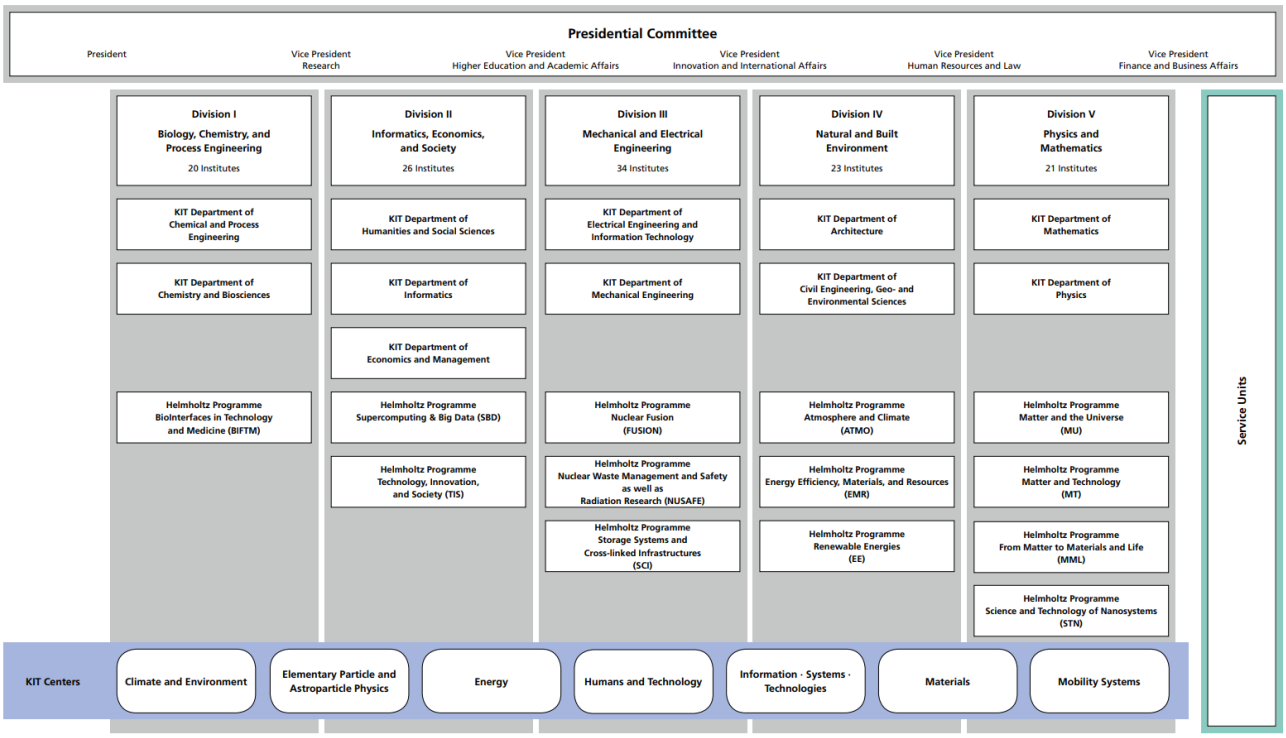

Figura 1: Organización científica en el KIT (Tomado de [2]).

3.2. Área de Conocimiento

El área de conocimiento principal del proyecto es computación aproximada (explicada brevemente en la sección de Contexto del Problema), la cual incluye aspectos tanto de software como de hardware, dado que involucra aspectos tanto de modificación del ISA como uso de ASIPs y aceleradores para la computación de ciertos algoritmos. Las áreas de conocimiento de ingeniería que se tratan son: conocimiento sobre aspectos de arquitectura y micro-arquitectura de procesadores (por lo mencionado anteriormente), desarrollo de compiladores, dado que se modificará el *frontend* de un compilador (Clang), finalmente, análisis de posibles aproximaciones Clang algoritmos en código escritos en el lenguaje de programación C, donde se espera agregar identificadores a las funciones que se busquen aproximar, para posteriormente ser procesadas de manera especial en la representación intermedia del código que se genere.

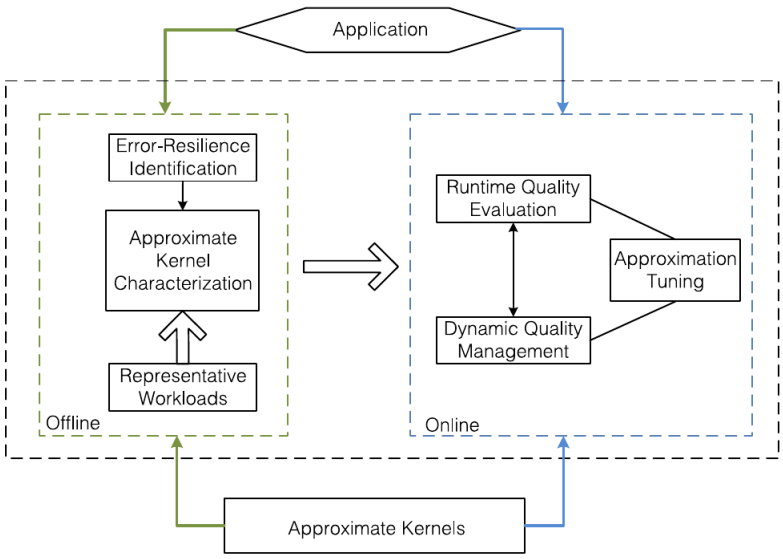

Figura 2: Un marco de trabajo para el uso de computación aproximada (Tomado de [3]).

4. Descripción de la Propuesta

4.1. Justificación y Definición del Problema

4.1.1. Contexto del Problema

En la actualidad, dada la gran cantidad de aplicaciones complejas (por ejemplo sistemas GPS, reconocimiento de voz, etc.), la computación aproximada ayuda a mantener una salida aceptable mientras se logra que ciertas métricas como tiempo de ejecución o eficiencia energética se mejoren. En general, la computación aproximada provee la libertad de escoger entre un cierto nivel de error o degradación de la calidad en la salida final de una aplicación (por ejemplo ruido en la señal de la salida) para mejorar el consumo de energía, el área o el tiempo de ejecución; esto sirve como herramienta a un investigador para que ajuste una aplicación dada a las necesidades reales y específicas de esta. En la Figura 2 se muestra un esquema que puede ser aplicado a sistemas tolerantes a errores para incluir en estos la computación aproximada [3].

Los elementos clave de la Figura 2 son *kernels* aproximados, los cuales representan la implementación (técnicas) de las funciones aproximadas, estas puede ser realizadas a nivel de hardware o de software; la identificación de la secciones tolerantes a errores y sus características particulares (análisis de impacto); y el manejo de la calidad, el cual implica una evaluación continua para determinar si la aplicación logra los requerimientos deseados.

Como se mencionó, la computación aproximada puede ser implementada tanto a nivel de software como de hardware. En software una implementación típica es a través de *Loop Perforation*, en la cual ciertos ciclos (usualmente con un patrón dado, como por ejemplo las pares) no

son computados, lo cual, por ejemplo en una aplicación de cálculo numérico, reduciría la precisión del valor final calculado. A nivel de hardware, se pueden utilizan módulos especializados, por ejemplo aceleradores para programas aproximados utilizando redes neuronales.

El graduado de la carrera Ingeniería en Computadores Juan Carlos Cruz, realizó un trabajo sobre la computación aproximada, donde él se dio a la tarea de caracterizar aplicaciones tolerantes a errores. Se busca partir de lo que Cruz trabajó y usar el conocmiento de qué versión aproximada es la correcta para entonces, a través de la modificación de Clang, generar un nuevo código C con las sustituciones correspondientes.

4.1.2. Especificación del Problema

Al contar con una aplicación que presenta una estructura en *pipeline*, es decir, que posee una serie de etapas donde cada etapa recibe su entrada de una etapa anterior y produce una salida para la etapa siguiente, y donde una o más etapas pueden ser aproximables con más de una versión aproximable (una versión se puede concentrar en mejorar el consumo de potencia, mientras que otra el tiempo de ejecución, por ejemplo) resulta complejo determinar qué combinación de versiones aproximadas utilizar de forma que no se sobrepase el error máximo permitido y a la vez se reduzca, de manera óptima, el uso de ciertos recursos. Dicho proceso podría tomar una cantidad considerable de tiempo si se decide probar todas las posibles combinaciones posibles de versiones aproximadas, por lo que es importante utilizar un esquema de trabajo diferente.

Una aplicación puede tener un comportamiento aproximado si alguna de sus etapas se puede aproximar, ya sea toda una sección o únicamente una función (dentro de una sección). La Figura 3 muestra como ejemplo una aplicación genérica donde ambas situaciones pueden ocurrir. En esta aplicación se tienen tres secciones, de las cuales la primera (por ejemplo, una etapa de preprocesamiento) puede ser completamente aproximada, la segunda no puede ser aproximada del todo (por ejemplo, una sección crítica de la aplicación) y, finalmente, la tercera tiene tres funciones específicas, de las cuales únicamente la segunda posee una versión aproximada.

En el caso hipotético de la Figura 3 donde se tenga más de una versión aproximada para la sección 1 y para la función 2 de la sección 3, es dificulta determinar qué combinación de versiones aproximadas produce la mejor aplicación aproximada final, debido a que, por ejemplo, un cambio en la sección 1 puede impactar severamente las secciones 2 y 3, pues dicha complejidad aumenta con la cantidad de versiones aproximadas; inclusive, puede que la versión aproximada de la función 2 determine qué tipo de versiones aproximadas son las más convenientes (según las especificaciones del usuario) en la sección 1 para no impactar en gran medida la calidad de la aplicación.

Actualmente, existe una manera premiliminar de hacer la exploración y determinar qué versión de una sección corresponde a ser utilizada según las restricciones del usuario; no obstante, no existe una herramienta que permita ingresar un código C con anotaciones propias y que, a partir de este, genere un nuevo código con las sustituciones correspondientes; es decir, una marco de trabajo más completo.

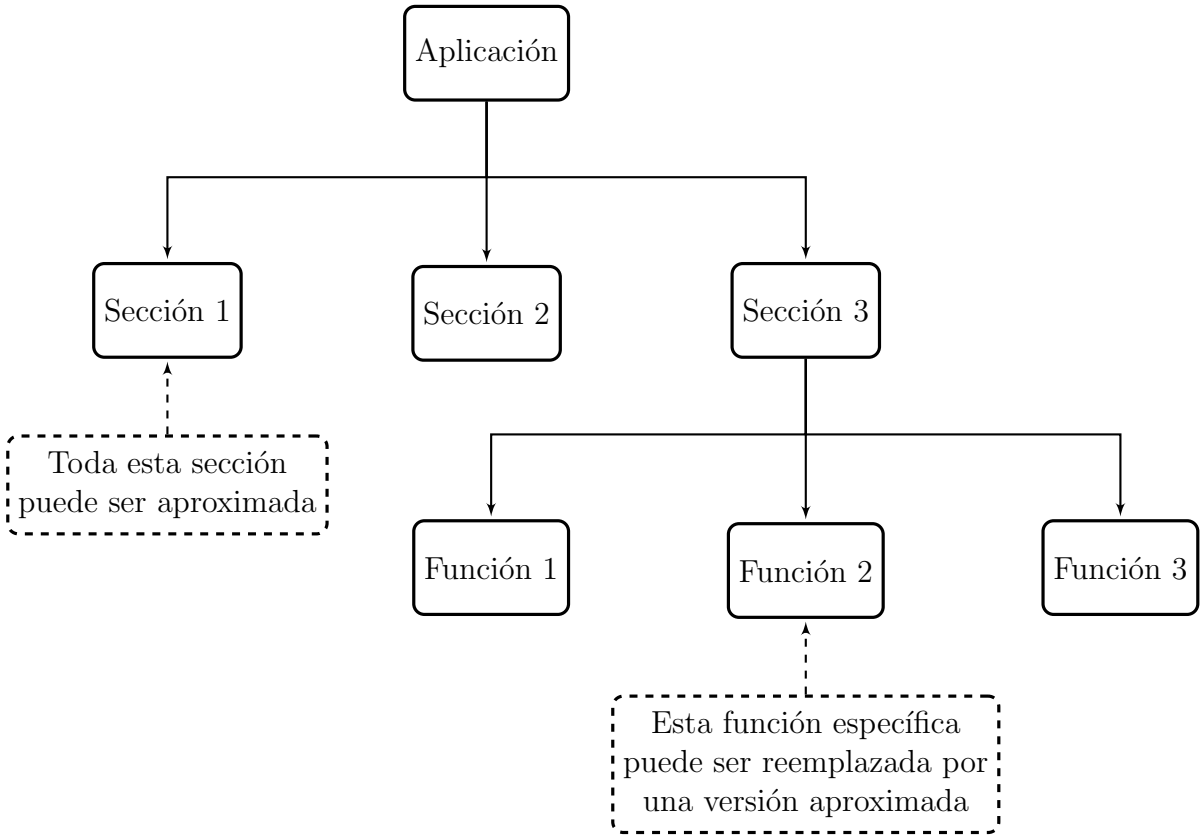

Figura 3: Una posible aplicación aproximada genérica analizada en este proyecto.

4.1.3. Justificación de la Necesidad

Debido a la creciente necesidad por un consumo eficiente de recursos, ya sea energía, área o tiempo de ejecución, la computación aproximada, como alternativa de solución a este problema, se considera considerablemente importante. El diseño de un marco de trabajo para la realización de aplicaciones personalizadas mediante el uso de computación aproximada puede traer nuevos conocimientos a esta área de investigación y potenciar la creación de más aplicaciones, especializadas según los requerimientos específicos de los usuarios.

Con este proyecto, el conocimiento generado en Alemania se puede traer para obtener grandes beneficios en Costa Rica, de tal forma que se incentive la investigación, e inclusive una posible inversión en esta área. Actualmente no existe una herramienta académica que realice lo que se plantea en esta propuesta, por lo que se espera que el proyecto incentive muchos otros proyectos en el área de generación de procesadores con funciones especializadas.

4.2. Enfoque de la solución

Se busca desarrollar una herramienta de software que pueda escoger entre diferentes versiones para una aplicación aproximada (cada versión dada por una combinación diferente de

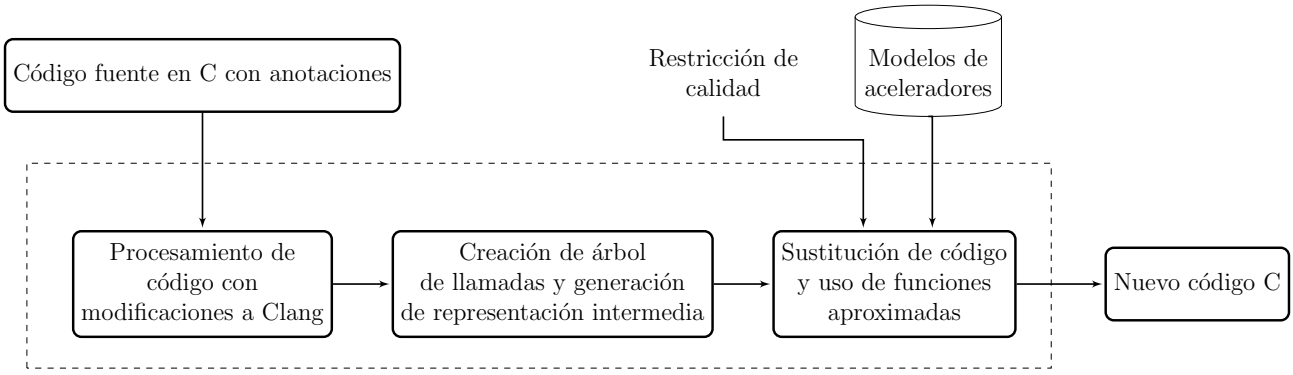

Figura 4: Esquema general de la solución propuesta.

versiones específicas para cada sección aproximable) y que como producto genere un nuevo código ejecutable, según el criterio de usuario que especifique cuáles recursos son críticos en la aplicación y cuál es la cantidad máxima de error permitido. La Figura 4 muestra una abstracción de la implementación de esta herramienta.

Como se muestra en la Figura 4, se espera recibir un código fuente de una aplicación aproximable, donde previamente el usuario ha indicado, a través de pragmas propios, qué funciones del código son tolerantes a errores; este es transformado a una representación intermedia, el cual consistirá de un árbol (o la estructura que se considere apropiada) con las llamadas de las funciones. Se toma el algoritmo desarrollado por Juan Carlos Cruz para poder escoger cuáles versiones aproximadas de las posibles que existen se ajustan para cumplir a cabalidad con el error permitido (restricción de ecalidad) y, de la mejor manera, con la priorización de recursos. Una vez identificadas las versiones que serán utilizadas, se procederá a sustituir el código específico que cuente con el soporte para las funciones aproximadas correspondientes, utilizando modelos funcionales de de aceleradores, los cuales son representaciones a nivel de software del hardware especialmente. Como producto final, se espera entregar una nuevo código fuente con todas las sustituciones correspondientes.

4.3. Especificación de Objetivos

4.3.1. Objetivo General

Desarrollar una herramienta que, a partir un algoritmo que determina cuáles versiones de secciones aproximadas corresponden en cierta aplicación, pueda generar una representación intermedia del código para finalmente generar una versión modificada del mismo.

4.3.2. Objetivos Específicos

1. Modificar el *frontend* (Clang) de un compilador de forma que se reconozca anotaciones propias que se agreguen en el código para establecer qué funciones son aproximables.

2. Abstraer el código de la representación intermedia a un árbol de llamadas o la estructura que se considere apropiada, que tome en cuenta las anotaciones dadas por el usuario.

3. Implementar el algoritmo propuesto para la exploración del espacio de diseño de posibles versiones de secciones aproximadas (aporte de Juan Carlos) en la herramienta a desarrollar.

4. Generar una nueva versión de código ejecutable donde se las anotaciones sean sustituidas por los aceleradores o versiones aproximadas correspondientes.

4.4. Beneficios y Beneficiarios con la Propuesta

1. Daniel Esteban Moya Sánchez: Se podrán fortalecer los conocimientos de arquitectura en computadores, compiladores, y aproximaciones a nivel de hardware y software, todo esto en un ambiente internacional, el cual aumentará el panorama cultural que se tiene, con los beneficios personales que esto conlleva. Se espera cunplir con los requerimientos del curso CE5600 Trabajo Final de Graduación, para así completar el plan de estudios 2100 de la carrera de Ingeniería en Computadores y poder graduarse para el año 2019.

2. Jorge Alberto Castro Godínez: Profesor tutor en el KIT, Alemania. Es investigador y estudiante de doctorado, a cargo de varios proyectos en el CES. Dentro de los beneficios está el conocimiento que se genere dado a el intercambio de ideas y resultados del proyecto, que ampliarán las habilidades técnicas tanto a nivel de hardware como de software.

3. Chair for Embedded Systems (CES): Corresponde al lugar específico en el KIT donde se estará llevando a cabo el proyecto. El proyecto propuesto podrá formar parte de la investigación en el CES, lo cual beneficia parte de las áreas de investigación que trata. Se espera que los resultados del proyecto incentiven nuevos proyectos y aumenten así el conocimiento en el área de computación aproximada.

4. Instituto Tecnológico de Costa Rica: Como parte de los principios de investigación y extensión, para el TEC es sumamente importante la presencia de estudiantes en el exterior. Los conocimientos que se generan a partir del proyecto propuesto podrán mejor la investigación en el TEC, e incentivar el área de computación aproximada.

4.5. Supuestos y Limitaciones

1. Limitación de tiempo: El proyecto se debe completar en un periodo menor a 5 meses, específicamente del 1 de Julio al 20 de Noviembre del 2018, dado que se necesita regresar al país para realizar la defensa presencial del proyecto en el TEC, además de realizar los trámites correspondientes para la graduación del 2019.

2. Disponibilidad de recursos: El proyecto se realizará utilizando herramientas de software libres. Para investigar se utilizará internet y el material disponible en el CES. Cualquier material físico del proyecto (como placa FPGA de desarrollo) será provisto por el CES de ser necesario.

3. Disponibilidad de versiones de funciones aproximadas y modelos de aceleradores: El proyecto parte de la existencia de diferentes versiones aproximadas de funciones en aplicaciones tolerantes a errores, que se utilizarán en conjunto con los distintos modelos de aceleradores como base para alimentar la herramienta propuesta.

4. Disponibilidad de algoritmo de exploración: El proyecto utilizará el algoritmo desarrollado por Juan Carlos Cruz para la exploración de posibles versiones de secciones (o funciones) aproximadas, el cual será implementado en la herramienta propuesta.

4.6. Análisis de Riesgos

En la Tabla 1 se resumen los riesgos que se consideran para el proyecto.

Tabla 1: Posibles riesgos del proyecto.

ID	Descripción	Probabilidad de ocurrencia	Impacto (horas)	Plan de Acción
1	Falta de disponibilidad de asesores para el proyecto	0.2	16	Aceptar
2	Carencia de materiales necesarios para el proyecto	0.2	8	Evitar
3	Errores a la hora de modificar Clang	0.3	12	Mitigar
4	Dificultades a la hora de implementar algoritmo de exploración	0.4	10	Mitigar

A continuación se explica cada riesgo detalladamente:

1. Falta de disponibilidad de asesores para el proyecto: Corresponde a momentos en que, ya sea el supervisor Jorge Castro o el director de tesis no puedan atender alguna duda o simplemente no se encuentren disponibles en una semana determinada, lo cual atrase la revisión del trabajo.

2. Carencia de materiales necesarias para el proyecto: El atraso de la entrega o no la presencia de alguno de los materiales (información o programas de software principalmente) que se necesitan para realizar el proyecto, siendo el más importante las versiones aproximadas y los modelos de los acelerados que se utilizarán.

3. Errores a la hora de modificar Clang: Problemas de dependencias o aceptación de las modificaciones realizadas para el *frontend* Clang.

4. Dificultades a la hora de implementar algoritmo de exploración: Problemas a la hora de entender y adaptar el algoritmo desarrollado por Juan Carlos Cruz en la herramienta propuesta.

Los planes de acción se detallan a continuación:

1. Aceptar: Asumir la responsabildidad correspondiente y continuar trabajando, con más cuidado inclusive y realizando un auto-análisis.
2. Evitar: Definir apenas comience el proyecto los materiales y sus fuentes con las que se contarán, de tal modo que se pueda contar con ellos (o al menos con una copia digitial) para evitar contratiempos de adquisición cuando se necesiten en el proyecto.
3. Mitigar: Se realizará una investigación previa del *frontend* Clang y se realizarán cambios iterativos, de tal forma que poco a poco se verifique las nuevas funcionalidades que se agreguen a Clang. Por otro lado, para la implementación del algoritmo de exploración, este será estudiado previo a que inicie el proyecto propuesto y se realizarán las consultas respectivas al supervisor Jorge Castro a lo largo del proyecto.

5. Propuesta Metodológica

5.1. Tipificación del Trabajo

El proyecto se clasifica como un trabajo de investigación aplicada, con alto porcentaje de experimentación. Esto se debe a que no existe ninguna herramienta que realice lo que se plantea y se busca incursionar en un campo relativamente nuevo (considerando la tecnología del país y la educación recibida hasta el momento), donde se espera que los resultados obtenidos inspiren nuevos proyectos en el área.

5.2. Descripción del Proceso

La Figura 5 resume el proceso que se realizará durante el proyecto. Como se puede observar, el proyecto iniciará con una breve investigación sobre maneras de modificar el compilador de *frontend* Clang para permitir el reconocimiento de pragmas o anotaciones personalizadas en el código.

Seguidamente, se analizará qué estructura es la más apropiada para abstraer el código recibido y generar una representación intermedia del mismo, de manera que sea fácilmente manipulable y considere las anotaciones dadas por el usuario. Para esta tarea se tomará información de una base de datos del KIT sobre aceleradores, para finalmente generar una nueva versión

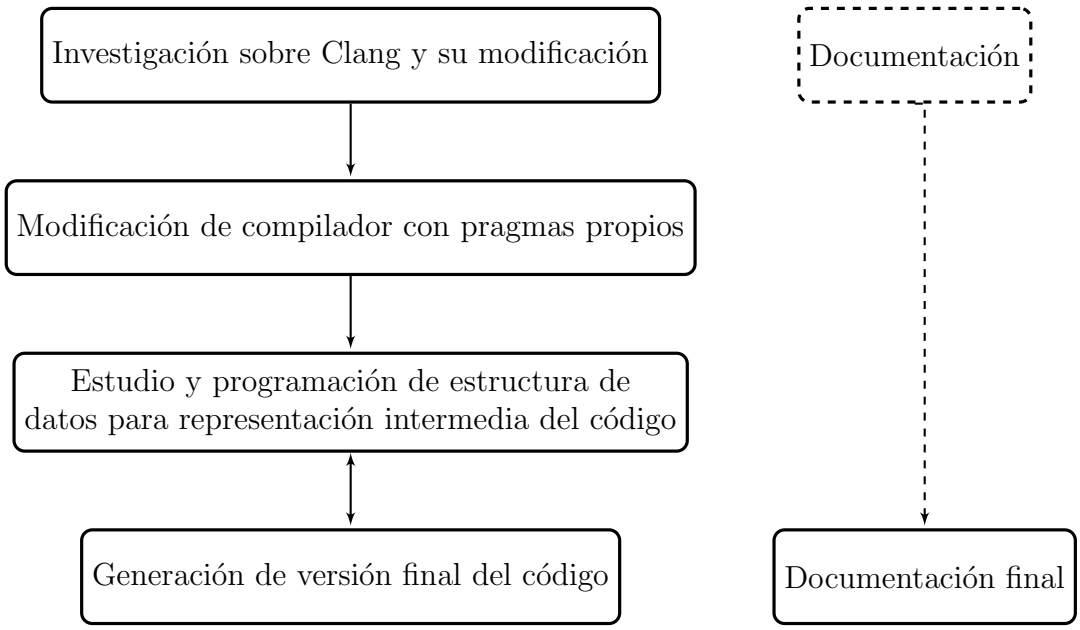

Figura 5: Proceso a realizar en este proyecto.

del código la cual contenga todos los detalles necesarios para simplemente ejecutarse con las versiones de secciones aproximadas seleccionadas.

Se parte de que el algoritmo (desarrollado por Cruz) que determinará qué versión de sección aproximada permite cumplir con los requerimientos del usuario con respecto al área, energía o calidad estará disponible y se realizará la adaptación de dicho algoritmo a la herramienta que se plantea desarrollar.

La documentación del proyecto se trabajará a lo largo de todo el proceso de desarrollo, de forma que al final se genere un artículo científico y demás documentos propios de un trabajo final de graduación.

5.3. Herramientas

1. Lenguaje de programación: El software desarrollado utilizará C/C++ como lenguaje de programación.

2. Sistema operativo: Se utilizará Ubuntu 17.10.

3. Compiladores: Se utilizarán los compiladores GCC/G++. Se modificará el compilador de *frontend* Clang para incluir las anotaciones propias.

4. Editor de texto: Para la documentación se utilizará LaTeX, con el ambiente de desarrollo Kile y compartido a través de Git. Para programar, se utilizará principalmente el editor en terminal Vim.

5.4. Descripción de Entregables

La tabla 2 presenta la asociación entre entregables y objetivos del proyectos.

Tabla 2: Entregables del proyecto

Objetivo	Entregable
Modificar el compilador de front-end Clang de forma que se reconozcan las anotaciones propias que se agreguen en el código para establecer qué funciones son aproximables.	Compilador Clang modificado con pragmas propios
Abstraer el código de la representación intermedia a un árbol de llamadas o la estructura que se considere apropiada, que tome en cuenta las anotaciones dadas por el usuario	Código en representación de árbol de llamadas u otro tipo de representación según se determine
Implementar el algoritmo propuesto para la exploración del espacio de diseño de posibles versiones de secciones aproximadas (aporte de Juan Carlos) en la herramienta a desarrollar	Adaptación del algoritmo desarrollado en la herramienta propuesta
Generar una nueva versión de código ejecutable donde se las anotaciones sean sustituidas por los aceleradores o versiones aproximadas correspondientes	Código generado con anotaciones (pragmas) sustituidas

5.5. Estrategias de Verificación y Validación

Dado que se trata de un proyecto de investigación, cada uno de los entregables mencionados anteriormente tendrá un conjunto de pruebas asociado, las cuales corresponden a:

- Pruebas para compilador Clang: se empezará por verificar la correcta funcionalidad con un código sin anotaciones y se irán realizando pruebas unitarias para cada pragma nuevo que se agregue.
- Pruebas para el código en representación intermedia: se revisará manualmente que el código en esta interpretación sea correcto.
- Pruebas de integración del algoritmo de exploración: el supervisor realizará una revisión de la funcionalidad del algoritmo una vez que se encuentre integrado con la herramienta propuesta.
- Pruebas para el código final generado (con pragmas sustituidos): el supervisor probará el código y determinará si cumple con los diferentes tipos de posibles requerimientos de un usuario.

Todas las pruebas mencionadas anteriormente serán llevadas a cabo mediante la supervisión de Jorge Castro y la contraparte en el TEC que se asigne.

5.6. Cronograma de Trabajo Propuesto

Considerando los 4 meses (16 semanas) del semestre, el cronograma de trabajo para el proyecto propuesto se muestra en la tabla 3.

Tabla 3: Cronograma para el proyecto propuesto

	Semana															
Actividad	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Modificación de Clang	■	■	■	■												
Generación de representación intermedia					■	■	■	■								
Generación de código con pragmas sustituidas (adaptación de algoritmo de Juan Carlos)									■	■	■	■				
Validación de la herramienta											■	■	■	■		
Documentación final								■	■	■	■	■	■	■	■	
Elaboración de presentación para la defensa del Trabajo Final de Graduación															■	■

Referencias

[1] Karlsruhe Institute of Technology. Data and facts, 2018.

[2] Karlsruhe Institute of Technology. Tasks and structure, 2018.

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

Ingeniería en Computadores
Ficha de contactos del proyecto

Datos del estudiante

Nombre	Daniel Esteban Moya Sánchez
Correo electrónico	danielmscr1994@gmail.com
Teléfonos	(+506) 8325 9730

Datos del proyecto

Nombre	Diseño asistido de aplicaciones aproximadas para sistemas computacionales personalizables
Breve descripción	Se desarrollará una herramienta que, a partir un algoritmo que determina cuáles versiones de secciones aproximadas corresponden en cierta aplicación, se pueda generar una representación intermedia del código para finalmente generar una versión modificada del código.
Fecha de inicio	Lunes 2 de Julio del 2018

Datos de la empresa u organización

Nombre	Chair for Embedded Systems (CES), Instituto Tecnológico de Karlsruhe (KIT), Alemania
Nombre contacto	Jorge Alberto Castro Godínez, M.Sc.
Correo electrónico	jocastro@itcr.ac.cr
Teléfonos	+49 721 608 48780