

PONTIFICIA UNIVERSIDAD JAVERIANA CALI

Computación de Borde con FPGA para IoT

por

Manuel A. Valderrama

Dirigido por

Eugenio Tamura

Anteproyecto de grado presentado en cumplimiento parcial para
el título de Magister en Ingeniería

en la
Facultad de Ingeniería
Departamento de Electrónica y Ciencias de la Computación
Maestría en Ingeniería

20 de mayo de 2018

Índice General

Índice General	III
Índice de Figuras	V
Índice de Tablas	VII
Abreviaciones	IX
1 Introducción	5
2 Definición del Problema	7
2.1 Procesamiento	7
2.2 Costos	7
3 Objetivos	9
3.1 Objetivo General	9
3.2 Objetivos Específicos	9
3.3 Resultados Esperados	10
4 Justificación	11
4.1 Aceleración del Procesamiento	11
4.2 Reducción de Costos	11
4.3 Enriquecimiento de la Información	11
5 Alcances	13
6 Marco Teórico	15
6.1 Estado del Arte	15
6.1.1 Computación de Borde	15
6.1.2 Síntesis de Alto Nivel HLS	17
6.1.3 Computación Consciente del Contexto	18
7 Metodología	21
8 Recursos	23
8.1 Humanos	23
8.1.1 Director	23
8.1.2 Grupo de Investigación	23
8.2 Económicos	24
8.2.1 Presupuesto	24
9 Cronograma de Actividades	25
10 Análisis de Resultados	27
10.1 Benchmark	28
10.2 Análisis de tiempo y recursos	29
10.2.1 Tiempo de ejecución	29
10.2.2 Recursos utilizados	30

11 Conclusiones	33
---------------------------	----

Bibliografía	35
------------------------	----

Índice de Figuras

6.1	Evolución y aplicaciones de IoT[1].	15
6.2	Distribución del Internet de las Cosas (Operación)[2].	16
6.3	Escalas de tiempos de servicios de la computación de borde vs la nube[3].	17
6.4	Red de sensores estructurada por capas [4].	19
10.1	Benchmark resultados de la implementación	28
10.2	Rendimiento del acelerador en Hardware	29
10.3	Recursos utilizados por la FPGA	30
10.4	Factor de ganancia recursos contra tiempo	31

Índice de Tablas

8.1 Presupuesto para el desarrollo del proyecto	24
---	----

Abreviaciones

IoT	Internet of Things
IP	Intellectual Property
QoS	Quality of Semony
FPGA	Field Programmable Gate Array
HLS	High Level Synthesis
SoC	System on a Chip
OS	Operative System

Resumen

Muchas de las soluciones de software que se pueden encontrar en el mercado actualmente tienen algún grado de procesamiento en la nube, lo que proporciona hardware y software bajo demanda, brindando así al desarrollador herramientas que hace 10 años eran impensables y hoy están al alcance de un click.

Las virtudes de la computación en la nube, aunque amplias, implican altos costos operativos y en algunos casos limitan el público no por su presupuesto sino por la propia naturaleza de la tecnología involucrada en las comunicaciones (vía Internet), que adiciona dificultades de tipo operativas en aquellas aplicaciones en las que el tiempo de respuesta es un factor clave. En estos casos, la computación en la nube no se puede tomar ni siquiera como una opción.

En este trabajo se pretende hacer uso de la computación de borde y la aceleración por hardware usando FPGAs como una herramienta que optimiza el ancho de banda y la cuota de datos enviados a la nube para su posterior procesamiento, con el objetivo de incrementar el número de investigaciones y proyectos de investigación en el ramo del Internet de las Cosas en el país.

Abstract

Currently, many of the software solutions that can be found in the market have some degree of processing in the cloud, providing hardware and software on demand, bringing to the developer tools that 10 years ago were unthinkable but today can be reached with a single click.

The virtues of cloud computing, although broad, imply high operating costs and in some cases limit the public not by its budget, but by the very nature of the technology involved in communications (via the Internet), which adds operational difficulties in those applications where the response time is a key factor. In these cases, cloud computing cannot be even taken as an option.

In this work, we will use edge computing and hardware acceleration using FPGAs as a tool that optimizes the bandwidth and data rate sent to the cloud for further processing, with the aim of increasing the number of research projects in the field of Internet of Things in the country.

Capítulo 1

Introducción

La computación en la nube como medio de procesamiento y análisis se convirtió en una herramienta con recursos casi ilimitados para los desarrolladores desde su masificación hace unos pocos años. La computación en la nube cuenta con múltiples bondades; entre ellas encontramos el IoT como una de las aplicaciones más importantes dada la gran cantidad de datos y recursos necesarios para procesar los altos volúmenes de información generados por las redes de sensores a los que usualmente se encuentra acoplada. Hoy, encontramos el término computación de borde, que aunque es un esquema relativamente nuevo de computación, ha llamado la atención de múltiples investigadores dado que implica llevar parte de los servicios de la nube al lugar donde realmente se producen los datos [3]. Realizar parte del procesamiento en el borde le permite al desarrollador acceder a aplicaciones sensibles a la latencia incapaces de operar directamente sobre la nube, reducir costos en analítica, volumen de datos procesados en la nube, almacenamiento, entre otros.

Aparte de las diferentes virtudes que ofrece la computación de borde, también encontramos marcadas falencias como la reducida potencia de cálculo de algunos de los dispositivos, con esto la aceleración por hardware y la reconfiguración parcial dinámica juega un rol importante sobre este nivel, aportando potencia de cálculo en un espacio reducido en función de las necesidades de procesamiento requeridas por la aplicación.

Capítulo 2

Definición del Problema

2.1. Procesamiento

La necesidad de procesar grandes bloques de información en tiempo real sobre la nube, ha traído consigo múltiples retos que no pueden ser resueltos por el esquema de operación en que está basado el *Cloud Computing*, es decir, que estos problemas son inevitables a menos que la aplicación no sea sensible a ellos. Uno de los problemas más comunes es la latencia, enviar grandes cantidades de información a través de Internet por un ancho de banda determinado puede retrasar el procesamiento hasta que toda la información esté disponible en el servidor de la nube, este problema se suma al tiempo que tarda en responder el servidor, imponiendo unas restricciones en el tiempo de respuesta mínimo que define la IEEE [5, 6].

2.2. Costos

El procesamiento centralizado ofrecido por la mayoría de proveedores de computación en la nube es robusto en general, pero se ve limitado debido a los altos costos operativos para cierto tipo de aplicaciones que no requieren mucha potencia de cálculo. Por otra parte el almacenamiento en la nube también implica costos tanto de persistencia de los datos como de la cantidad de transacciones que se realicen al medio de almacenamiento [7–10].

Capítulo 3

Objetivos

3.1. Objetivo General

Proponer una plataforma de computación de borde para IoT que proporcione aceleración por hardware a través de la reconfiguración parcial de manera dinámica con el objetivo de permitir que la plataforma se ajuste de acuerdo al contexto de ejecución.

3.2. Objetivos Específicos

- Definir los requerimientos y los lineamientos requeridos por la plataforma.
- Desarrollar algoritmo de fusión de datos con filtro de Kalman como módulos IP usando HLS.
- Integrar módulos IP de fusión de datos como plataforma consciente del contexto usando reconfiguración parcial.
- Comparar el rendimiento de la plataforma desarrollada con una implementación diseñada completamente en software.

3.3. Resultados Esperados

Se espera tener una plataforma mixta hardware-software que sea capaz de procesar datos provenientes de múltiples sensores de variables físicas relacionadas entre sí para combinarlos en un estimado que posteriormente será enviado a la nube para su procesamiento y análisis.

Para efectos de desarrollo se dispondrá de una base de datos con la que la plataforma interactuará simulando la captura de datos a lo largo del tiempo.

- Los datos procesados contendrán información más rica y útil que los valores individuales.
- El tiempo de procesamiento de los datos usando aceleración por HW deberá ser menor que su contraparte SW.

Capítulo 4

Justificación

4.1. Aceleración del Procesamiento

Una plataforma de procesamiento concurrente como una FPGA provee aceleración del procesamiento según la configuración adoptada por el desarrollador. Dependiendo de la cantidad de hardware asociado y el diseño del algoritmo, una FPGA es capaz de acelerar varias veces los resultados por unidad de tiempo a diferencia de su contraparte software como por ejemplo casos como la ejecución de consultas a bases de datos, criptografía, simulaciones, entre otros [11–14].

4.2. Reducción de Costos

Llevar parte de los servicios de la nube al campo implica la reducción de costos operativos. Esta migración, conocida como computación de borde, implica para el desarrollador invertir más recursos en servidores locales de procesamiento y almacenamiento de datos que tienen un costo fijo de adquisición y un costo bajo de operación en contraste con los servicios de la nube [6, 7].

4.3. Enriquecimiento de la Información

La información en crudo de una o múltiples fuentes no aporta la suficiente información para realizar conclusiones en un modelo cuyo objetivo es realizar el análisis de los datos. El procesamiento de datos en el borde permite al desarrollador analizar muchos más detalles, tomar decisiones o incluso observar y predecir el comportamiento de las variables implicadas en la medición, con lo cual el desarrollador obtiene una visión más amplia de lo que ocurre en el medio observado [4, 15, 16].

Capítulo 5

Alcances

- 1. Definir los requerimientos y los lineamientos requeridos por la plataforma**
 - Definir las características de la plataforma según la tecnología de cómputo y fuentes de datos disponibles.
- 2. Desarrollar algoritmo de fusión de datos con filtros de Kalman y Lógica Difusa como módulos IP usando HLS**
 - Desarrollo de un algoritmo de fusión de datos en C o C++ que permita el procesamiento de los datos provenientes de múltiples sensores para luego ser sintetizado usando síntesis de alto nivel.
- 3. Integrar módulos IP de fusión de datos como plataforma consciente del contexto usando reconfiguración parcial**
 - Se espera desarrollar una plataforma que se reconfigure parcialmente basado en el tipo sensor según la variable, tiempo de muestreo y la cantidad de sensores sincronizados con el gateway.
 - La cantidad de señales en paralelo que se puedan analizar dependerá del alcance número 1 y los recursos de hardware (cantidad de bloques lógicos de la FPGA) con los que cuente la plataforma de desarrollo.
- 4. Comparar el rendimiento de la plataforma desarrollada con una implementación diseñada completamente en software**
 - Análisis de rendimiento de los módulos según el tiempo de procesamiento y los recursos de HW utilizados contra una implementación netamente SW.

Capítulo 6

Marco Teórico

6.1. Estado del Arte

6.1.1. Computación de Borde

El Internet de las Cosas definido como una red distribuida de dispositivos heterogéneos (sensores, actuadores y procesadores) que intercambian mensajes entre ellos sin intervención humana, ha cobrado particular interés por parte de los investigadores dado que miles de millones de dispositivos se estarán comunicando en un futuro cercano. La Figura 6.1 muestra la tendencia de la capacidad de procesamiento y la cantidad de servicios que el IoT puede ofrecer, así como también aumentan los retos de seguridad y administración de los datos [1, 17].

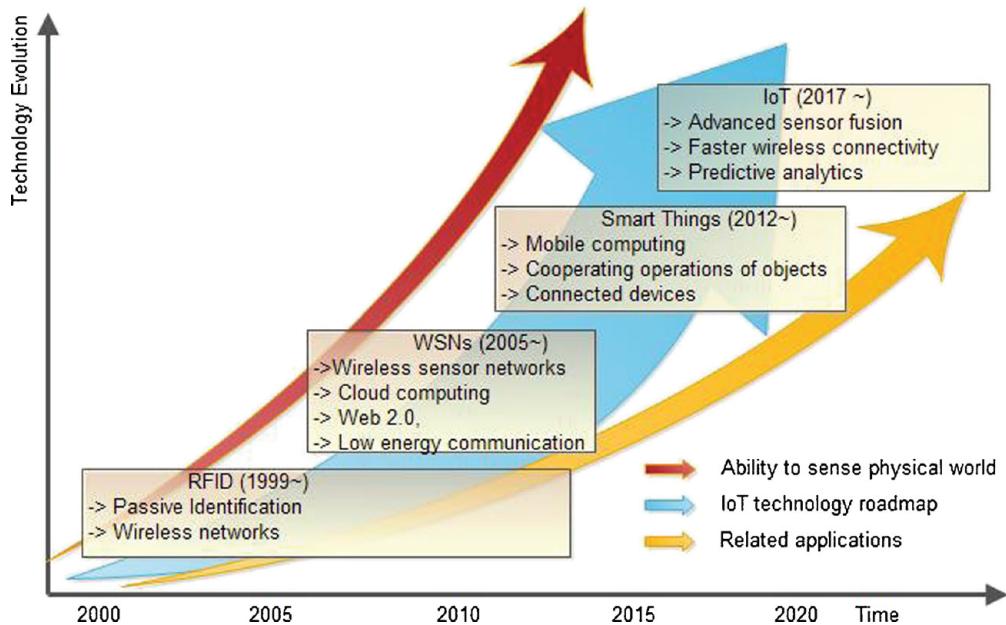


FIGURA 6.1: Evolución y aplicaciones de IoT[1].

Con el incremento en la cantidad de servicios también surgen otras necesidades; unas de ellas y quizás las más críticas en la arquitectura pensada para IoT es la latencia de los datos y el rendimiento del sistema [2]. Se propone llevar algunos de los servicios de la nube más cerca del nivel al que se encuentran los datos, se propone sub- dividir en múltiples niveles según las necesidades de las diferentes aplicaciones como lo muestra la Figura 6.2. En el caso del nivel *Extremo* ó *MIST* se tienen niveles de procesamiento bajos; en este nivel son los mismos nodos sensores y actuadores los que toman decisiones operativas a un nivel muy bajo dadas las limitaciones en los procesadores que requieren para funcionar. A nivel de *Borde* ó *FOG* se tiene mayor potencia de cálculo permitiendo realizar labores de almacenamiento, analítica, toma de decisiones, etc. Esto es posible dado que en algunas aplicaciones el dispositivo de borde, también conocido como *Gateway*, hace la labor de establecer un puente entre el extremo y la nube, contando con una radio potente y una conexión a Internet y corriente eléctrica estables; es por estas necesidades que el gateway dentro de su concepción debe contar con hardware más potente que los demás dispositivos. Finalmente, sobre la última capa se encuentra la *Nube* ó *CLOUD*, el cual es el pilar más importante del Internet de las Cosas, ofreciendo múltiples servicios, con recursos prácticamente ilimitados en procesamiento, capaz de escalar dinámicamente conforme las necesidades lo requieran [3].

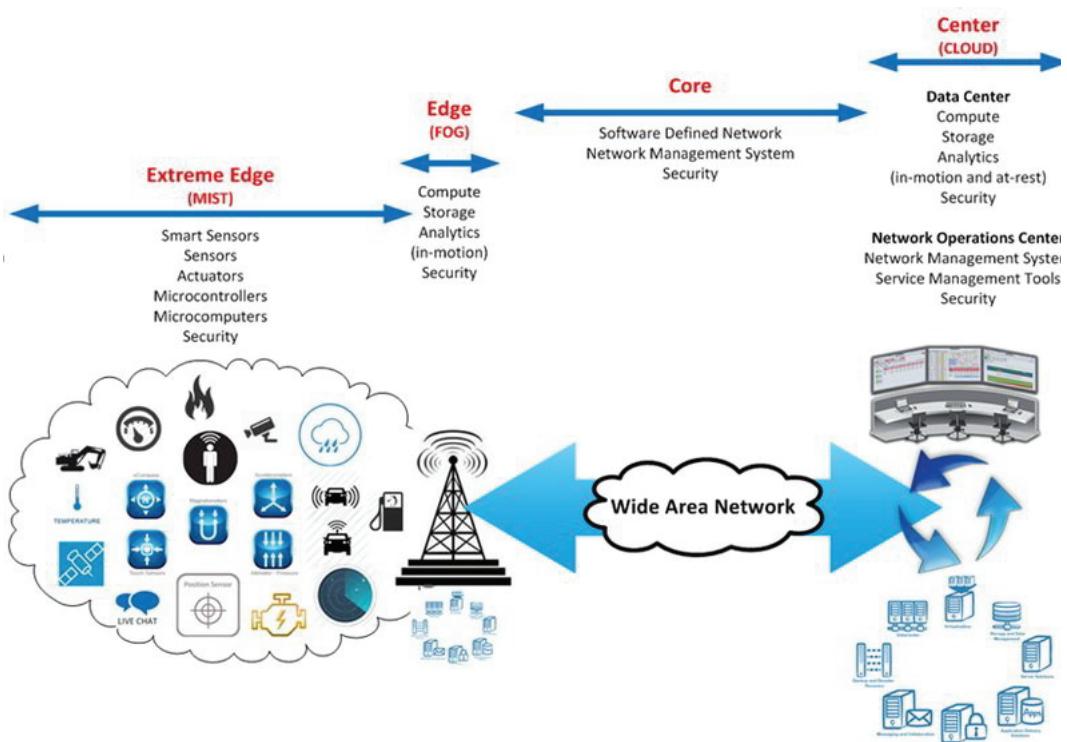


FIGURA 6.2: Distribución del Internet de las Cosas (Operación)[2].

Entonces como se sugiere, la computación de borde está orientada a la comunicación entre dispositivos, logrando así almacenamiento y toma de decisiones eficientes. Esto implica directamente reducción en tiempo de procesamiento, almacenamiento, visualización y manejo de los datos como se muestra en la Figura 6.3.

Al final la nube como el pilar fundamental ofrece potencia de cálculo y por ende más servicios [18] pero algunas de las aplicaciones únicamente requieren parte de estos servicios, que pueden ser resueltos en el borde por un gateway. Las aplicaciones que hacen uso de la computación de borde pueden ser de cualquier tipo; por ejemplo, encontramos modelos de programación para aplicaciones que requieran procesamiento de borde a gran escala [19], Gateways inteligentes capaces de enrutar de mejor manera los paquetes de red que tienen como destino la nube [20], modelos de *centros de datos* para el procesamiento de datos efectivos en el borde [21], QoS distribuido dentro de la red de IoT [22]. Como se ve, el concepto es ampliamente usado pero el terreno sobre la aceleración por hardware sigue sin ser explorado dadas las complicaciones que requiere su programación. Con sintetizadores de alto nivel apenas madurando, se espera avanzar hacia la creación de APIs ó middlewares que permitan a los desarrolladores incrementar la potencia de cálculo en el borde como lo sugieren las patentes [23, 24].

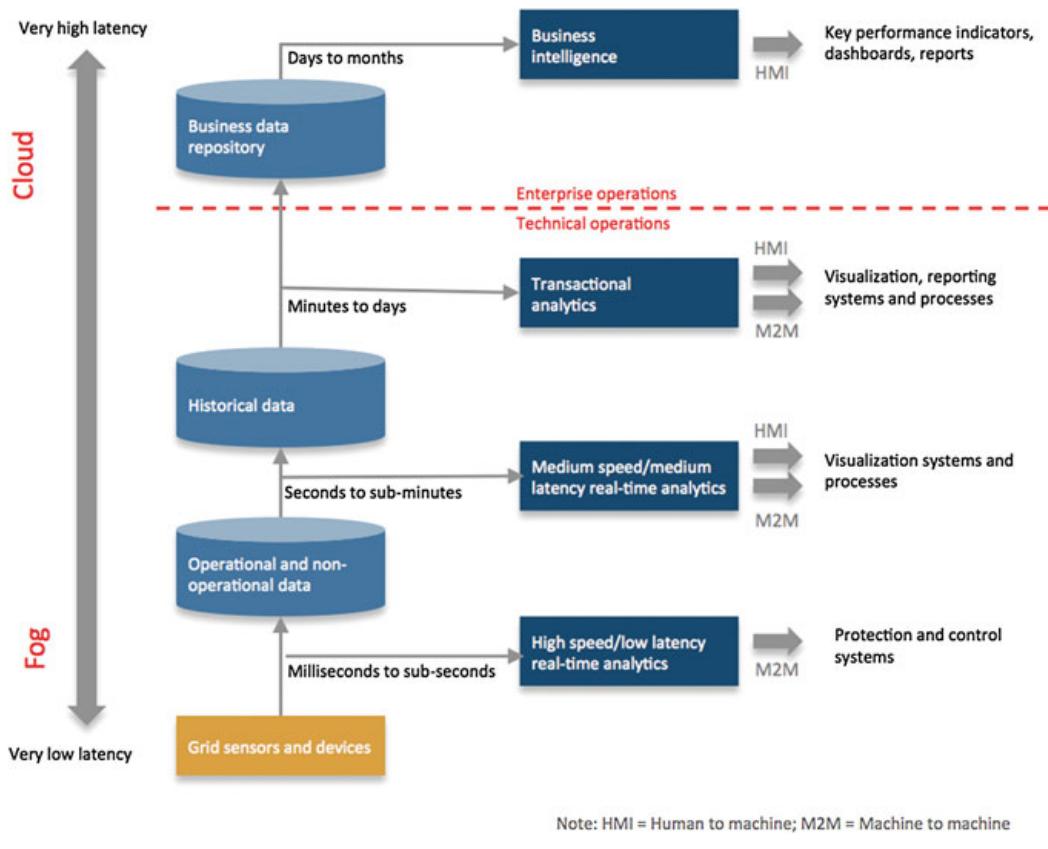


FIGURA 6.3: Escalas de tiempos de servicios de la computación de borde vs la nube[3].

6.1.2. Síntesis de Alto Nivel HLS

La síntesis de alto nivel HLS le permite al desarrollador programar hardware usando lenguajes con altos niveles de abstracción como lo son C/C++ sin la necesidad de hacer uso de lenguajes de descripción de hardware como VHDL o Verilog cuyo desarrollo y simulación es lento y complejo comparado con el tiempo que toma desarrollar en un lenguaje de alto nivel. Actualmente, la tercera generación de compiladores de HLS

ofrece a los desarrolladores ventajas significativas frente a las revisiones anteriores como la optimización del proceso de síntesis de hardware, la estandarización del uso de C, C++ y System C como lenguajes de entrada al software de síntesis y la posibilidad de adicionar directivas de optimización sin realizar modificaciones al código fuente, co-simulación entre los diferentes lenguajes de entrada y la RTL objetivo, entre otras [25].

Las bondades de HLS frente a los paradigmas de programación de hardware tradicionales son muchas como se mencionó anteriormente. Una de ellas y quizás de las más importantes, es la aplicación de optimizaciones de hardware, que según el contexto puede incrementar el rendimiento de un circuito modificando la cantidad de hardware dedicado al procesamiento de cada tarea [26, 27].

Por otra parte, HLS provee al desarrollador la opción de crear hardware reconfigurable de forma dinámica, esto es conocido como reconfiguración parcial o PR por sus siglas en inglés. La aplicación de la reconfiguración parcial le permite al desarrollador integrar múltiples módulos de hardware con diferentes algoritmos que pueden ser instanciados sobre la superficie de la FPGA de forma dinámica [28–30] con el objetivo de tener en un único dispositivo múltiples funciones sin la necesidad de reprogramar la FPGA completamente.

Algunos de los proyectos orientados a usar netamente HLS como medio de prototipado e implementación de sistemas hardware se pueden encontrar soluciones desde optimizar procesos de kernel de un S.O., pasando por acceder y modificar bases de datos de forma concurrente usando paralelismo y finalizando con la fusión de datos [31–38].

6.1.3. Computación Consciente del Contexto

Cuando se habla de grandes cantidades de información recolectada por miles de sensores dentro de un esquema de IoT, se pone sobre la mesa una pregunta, ¿Cómo interpretar los datos?. Es importante notar que por más información que se tenga disponible si no se analiza, interpreta y entiende dentro de un contexto dado, esta información no tiene valor alguno.

La computación consciente del contexto, vista desde una perspectiva de IoT propone una forma de mediar con la información desde diferentes flancos. Actualmente hablamos de la implementación de soluciones middleware capaces de añadir recursos que se encarguen de abstraer, fusionar y comprimir los datos, además de añadir servicios como QoS, paradigmas de programación, adaptabilidad, escalabilidad, recursos de hardware y lo más importante, seguridad de los datos [39].

La Figura 6.4 ofrece una vista general de la forma como se distribuye una red de sensores por capas, desde los elementos más básicos hasta los más complejos. En las primeras tres capas se observa el flujo de datos que va desde los dispositivos menos potentes hasta los más potentes en la capa 3, que muchas veces hacen las veces de enrutadores de borde de red. En las capas 4 y 5 se tienen dispositivos computacionalmente mucho más potentes; estos son conocidos como Gateways, capaces de llevar la información del campo hacia la nube. Finalmente, se tiene la nube en internet donde la información capturada puede ser transformada y procesada para su análisis. Cada una de las capas cumple

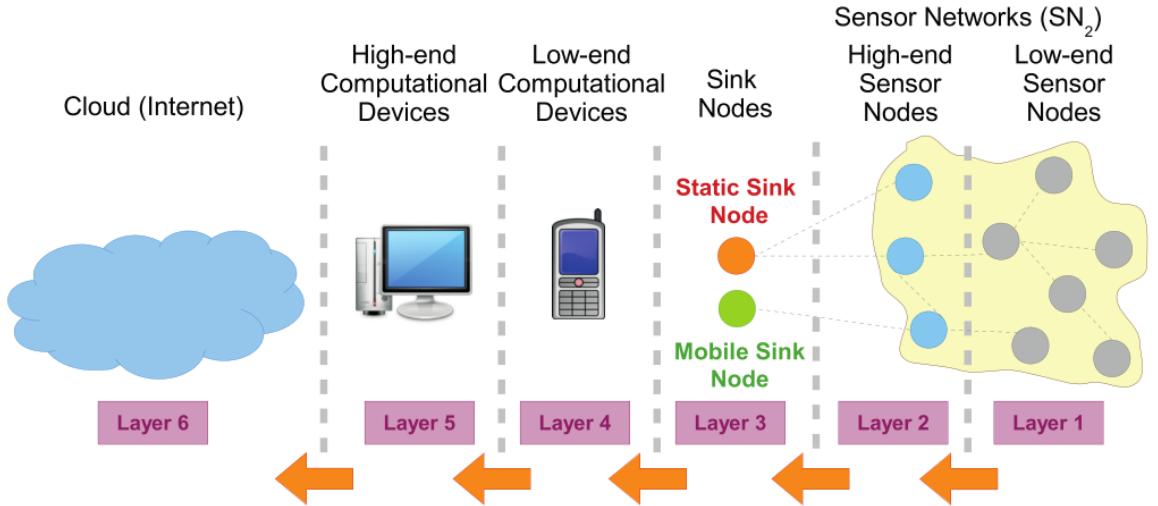


FIGURA 6.4: Red de sensores estructurada por capas [4].

un rol específico en el que la eficiencia es la clave[40]. Es decir, se podría realizar el procesamiento entre las dos primeras capas reduciendo considerablemente el costo de las comunicaciones, pero las limitaciones en capacidad de cómputo y energéticas impedirían realizar un análisis detallado de la información recolectada por la red.

En función de esta necesidad los investigadores definieron siete grandes características que permiten enmarcar una solución para el IoT dentro del contexto de la computación consciente del contexto que necesariamente, dadas las condiciones, deben ser realizadas entre las capas cuatro a seis [41]. Estos son: inteligencia, arquitectura, sistemas complejos, magnitud, tiempo, espacio y todo como un servicio. Algunas de estas cobran particular interés, como inteligencia, pues determina que para usar la información recolectada por diferentes fuentes en crudo se debe transformar para lograr lo que se denomina *información de alto nivel*, que lo que quiere decir es que esta información nueva enriquecida y puesta en un contexto, pueda ser usada como insumo para otro tipo de operaciones. Otra de las características que cobra particular interés es la de tiempo, pues dados los altos volúmenes de transacciones de datos se hacen necesarios mecanismos que permitan operar en tiempo real, siendo una restricción importante cuando se modela una solución de este tipo.

A partir de este punto se introduce el concepto de "*Principios Administrativos de Diseño Consciente del Contexto*", que básicamente se define como los requerimientos para la aplicación de un middleware consciente del contexto de operación, que se encuentre en capacidad de analizar, transformar e interactuar con el medio para darle sentido a la información [42–44]. Uno de los requerimientos más tangibles es el llamado *optimización de recursos*, el cual implica que cualquier mejora en los recursos de procesamiento por más pequeña que sea tendrá un gran impacto en el procesamiento dados los altos volúmenes de información que el IoT puede llegar a procesar.

Finalmente, tenemos la definición estricta de Schilit [45], en la que una aplicación consciente del contexto no es más que un dispositivo que cumple unas reglas básicas según

la operación que realiza, desde la toma simple de decisiones basado en eventos hasta la resolución de los mismos usando ventanas de oportunidad en las que las condiciones del evento establecen las fronteras.

Capítulo 7

Metodología

Para el desarrollo de los objetivos se seguirán los siguientes lineamientos.

1. Aplicación de la computación consciente del contexto

- *Aplicación de algoritmo de procesamiento de datos y escalamiento de hardware:* Una vez definidos los algoritmos a usar, se desarrollarán inicialmente en un lenguaje de programación como C/C++ el cual servirá de insumo para el producto final usando HLS.
- *Comparación de algoritmos:* Se realizará una comparación de los algoritmos según la cantidad de sensores a fusionar.
- *Escritura:* Documentar los hallazgos encontrados.

2. Elección de la plataforma de computación para IoT que permita recon��uraci配n parcial dinámica

- *Búsqueda tarjeta de desarrollo:* Se consultan las diferentes tarjetas de desarrollo hardware disponibles con programación en alto nivel.
- *Verificación:* Se realiza un análisis de compatibilidad entre lenguajes de programación y herramientas disponibles.
- *Escritura:* Se especifica la arquitectura y herramientas a usar.

3. Integración de los diferentes módulos dentro de la plataforma de procesamiento usando síntesis de alto nivel

- *Programación HLS:* Se programan los algoritmos de fusión usando síntesis de alto nivel HLS.
- *Programación RP:* Se realiza la configuración de los módulos que harán uso de la reconfiguración parcial.
- *Escritura:* Se reportan los recursos y métodos de hardware usados para la implementación.

4. Comparar el rendimiento de la plataforma desarrollada con una implementación diseñada completamente en software

- *Pruebas iniciales:* Se realiza la verificación de funcionamiento del sistema completo.
- *Correcciones:* Se corrigen posibles errores de implementación y diseño.
- *Pruebas finales:* Se realizan pruebas de rendimiento y funcionalidad del sistema en su última etapa.
- *Evaluación final:* Se evalúa el desempeño del sistema en su totalidad con lo cual se obtienen los resultados a reportar en el documento.
- *Escritura:* Se reportan los resultados obtenidos.

Capítulo 8

Recursos

8.1. Humanos

8.1.1. Director

- Nombre: [Dr. Eugenio Tamura](#).
- Dedicación: 2 Horas/Semana.
- Función: Asesoría investigativa.

8.1.2. Grupo de Investigación

- Grupo de Automática y Robótica GAR.
- Director: Luis Eduardo Tobón.
- Clasificación Colciencias: A1.
- Área de Conocimiento: Ingeniería y Tecnología Ingenierías Eléctrica, Electrónica e Informática.
- GrupLAC: <https://goo.gl/NvuWzE>.

8.2. Económicos

8.2.1. Presupuesto

ITEM	UNIDADES	COP	DESCRIPCIÓN
LIC. XILINX VIVADO SYSTEM EDITION	1	\$10'485.000	Licencia de software de desarrollo, incluye Vivado HLS (High Level Synthesis, Reconfiguración Parcial y Xilinx SDK) https://goo.gl/bSQiq9
TARJETA DE DESARROLLO FPGA	1	\$567.000	Tarjeta de desarrollo FPGA ZYBO https://goo.gl/z0OWdb
PC DE DESARROLLO	1	\$3'150.000	Computador de desarrollo con 16GB RAM, 1TB HDD, 256GB SSD, VGA NVIDIA 960M
TIEMPO DE DESARROLLO	1	\$11'200.000	Tiempo de desarrollo por hora por seis meses.
ASESORIAS	1	\$182.400	Costo de transporte durante seis meses de desarrollo.
PAPELERIA	1	\$30.000	Costo de papelería, impresiones, anillado, etc.
TOTAL	-	\$25'614.400	Costo total aproximado de desarrollo de la tesis de grado.

TABLA 8.1: Presupuesto para el desarrollo del proyecto

Capítulo 9

Cronograma de Actividades

Capítulo 10

Análisis de Resultados

En este capítulo se muestran los resultados obtenidos en la implementación del hardware. Para esto se sigue la siguiente metodología:

Se usa como banco de pruebas la tarjeta de desarrollo ZYBO-INDICAR CUAL— con la distribución de Petalinux 2017.4 con Kernel de Linux versión 4.9.0 y File System por defecto de Petalinux 2017.4.

- Se analizan los tiempos de ejecución del código a implementar en Hardware junto con la versión Software mediante un Benchmark. Este medirá el tiempo de ejecución a lo largo de tres iteraciones, en el caso del HW se medirá el rendimiento del mismo a las frecuencias de 100MHz y 150MHz, pudiendo así determinar la eficiencia del circuito.
- Se analizarán los tiempos de ejecución del Hardware a lo largo de 4 estrategias de optimización obtenidas mediante HLS y la cantidad de recursos usados para cada solución.
- Se analizará el circuito con mejor rendimiento contra él mismo, modificando la estrategia de síntesis a una de optimización de área buscando reducir la cantidad de recursos usados por la solución.
- Finalmente se determina la relación rendimiento contra uso de recursos buscando la eficiencia más alta.
- Se determinará la ganancia SW vs HW posteriormente se analizará

10.1. Benchmark

Dada la necesidad de medir el rendimiento del circuito implementado en HW contra la versión SW se determinó que la mejor métrica se obtendría realizando un Benchmark de tres etapas con ejecuciones de 100 y 300 millones de operaciones con la implementación en HW funcionando a 100MHz y 150MHz(Valores recomendados por el sintetizador de alto nivel). La prueba consta de cargar las neuronas de entrada y calcular el resultado.

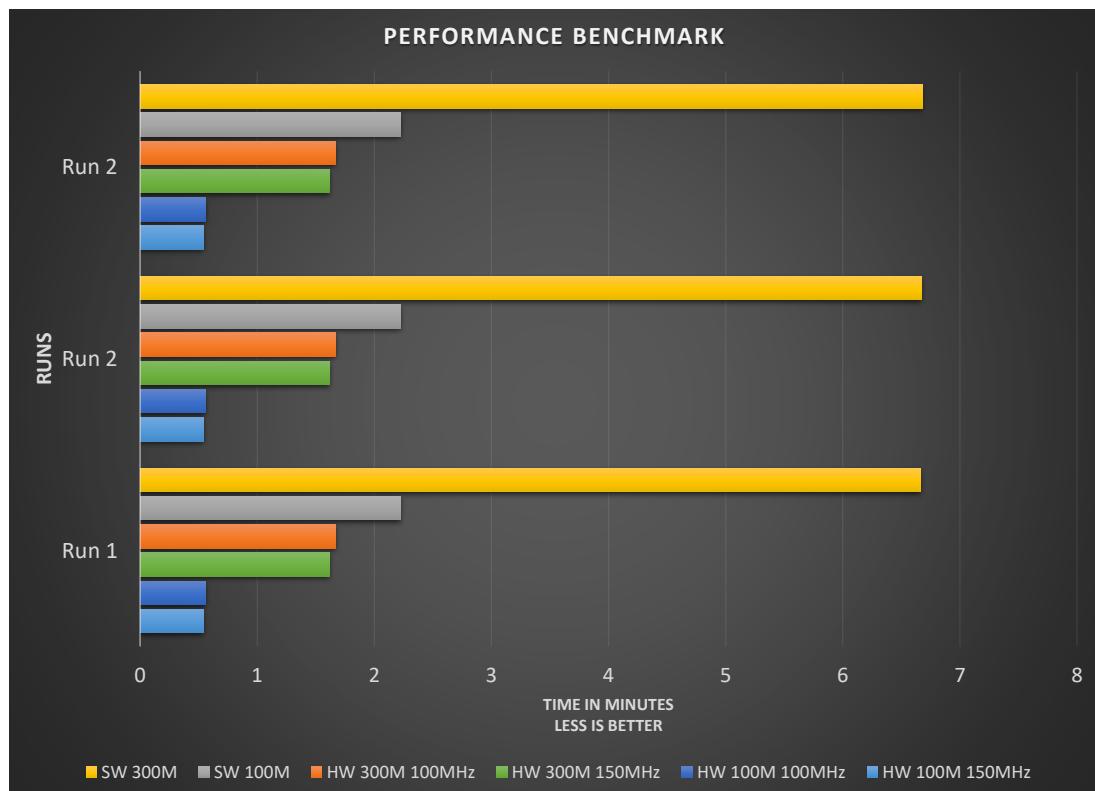


FIGURA 10.1: Benchmark resultados de la implementación

Como se observa en la figura 10.1, la implementación en HW es aproximadamente cuatro veces más eficiente, esta prueba lo que busca es saturar el bus AMBA cargando y exigiendo al máximo el pipeline del bus usando las optimizaciones burst de la interfaz AXI. A diferencia de la implementación HW, la implementación software realiza una carga de las neuronas de entrada de forma secuencial adicionando un tiempo de carga alto debido a los múltiples llamados al bus que se encarga de la RAM.

10.2. Análisis de tiempo y recursos

10.2.1. Tiempo de ejecución

Se observa que en la figura 10.2 la implementación de diferentes estrategias de optimización del hardware, en este caso tenemos:

- WO-OPT: Sin optimizaciones.
- P-UNROLL: Pipelining a nivel de bucles combinado con Unroll a los bucles.
- DATAFLOW: Pipelining a nivel de tareas.
- AO: Optimización de área.
- PO: Pipelining a nivel de tareas, bucles y unroll

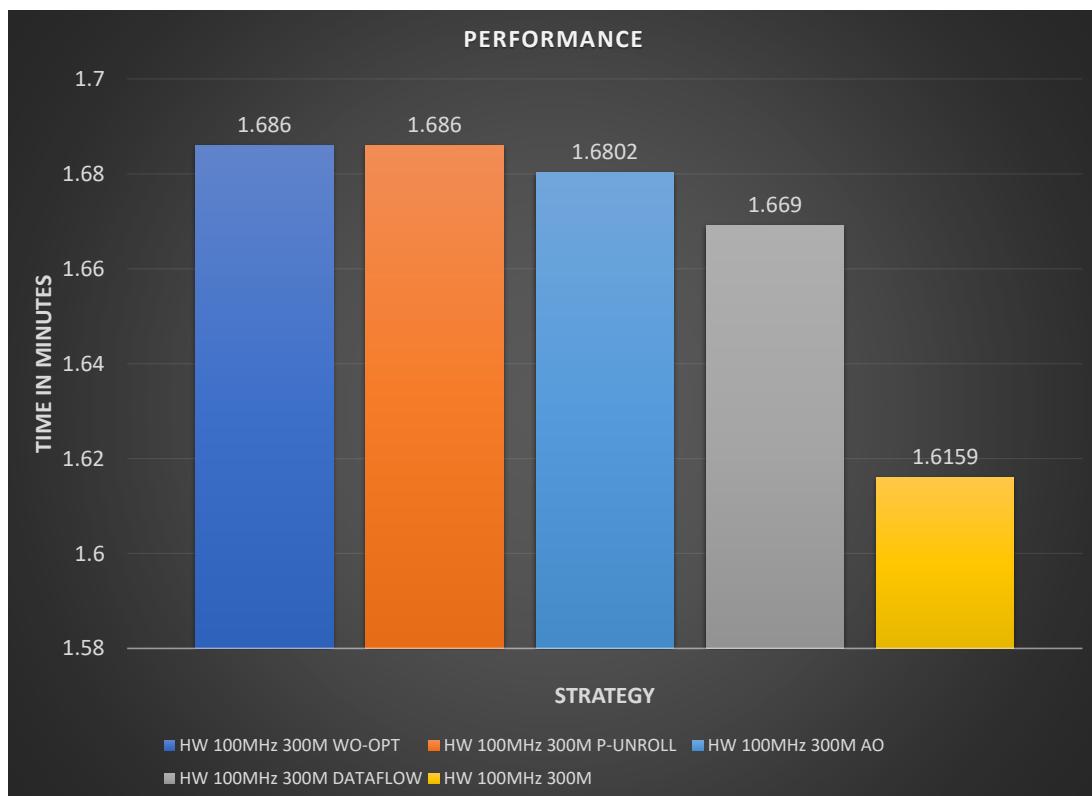


FIGURA 10.2: Rendimiento del acelerador en Hardware

La figura muestra claramente que la estrategia "PO." ofrece el mejor rendimiento en tiempo de ejecución de las tareas, siendo de primera mano la estrategia a implementar en producción.

10.2.2. Recursos utilizados

Una vez determinados los tiempos de ejecución se hace necesario tener en cuenta el área o la cantidad de recursos necesarios para cumplir las diferentes tareas de la solución. En este caso se analizó la cantidad de recursos para cada una teniendo en cuenta el reporte del sintetizador de hardware Vivado.

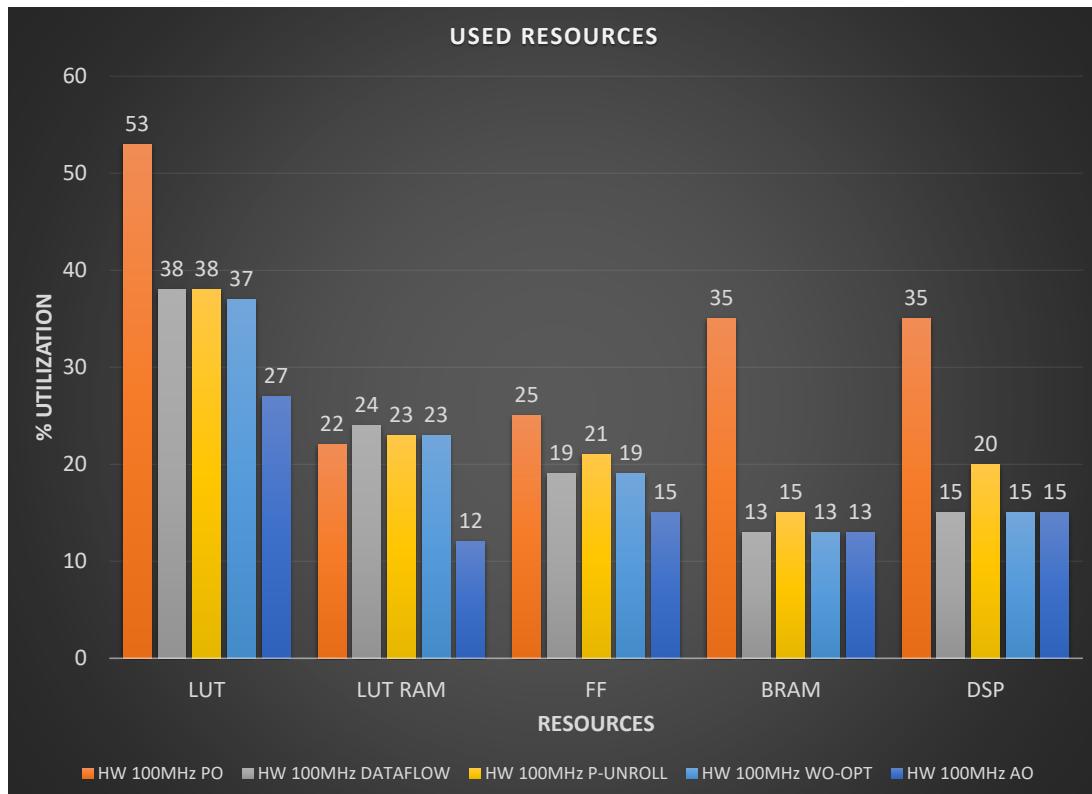


FIGURA 10.3: Recursos utilizados por la FPGA

La figura 10.3 muestra el porcentaje de recursos usados por cada una de las diferentes estrategias. Es importante notar que la estrategia ".AO" usa 51 % menos LUT, 42 % menos DSPs y 37 % menos BRAM que su contraparte "PO" para un rendimiento casi marginal en tiempo.

La figura 10.4 muestra claramente que la estrategia ".AO" es aproximadamente 46 % más eficiente que su contraparte "PO". De esta forma es posible determinar que al usar menos área vamos a obtener un mejor delta de eficiencia pues al usar menos recursos es posible adicionar mas módulos de hardware o en otros casos usar menos potencia en tiempo de ejecución.

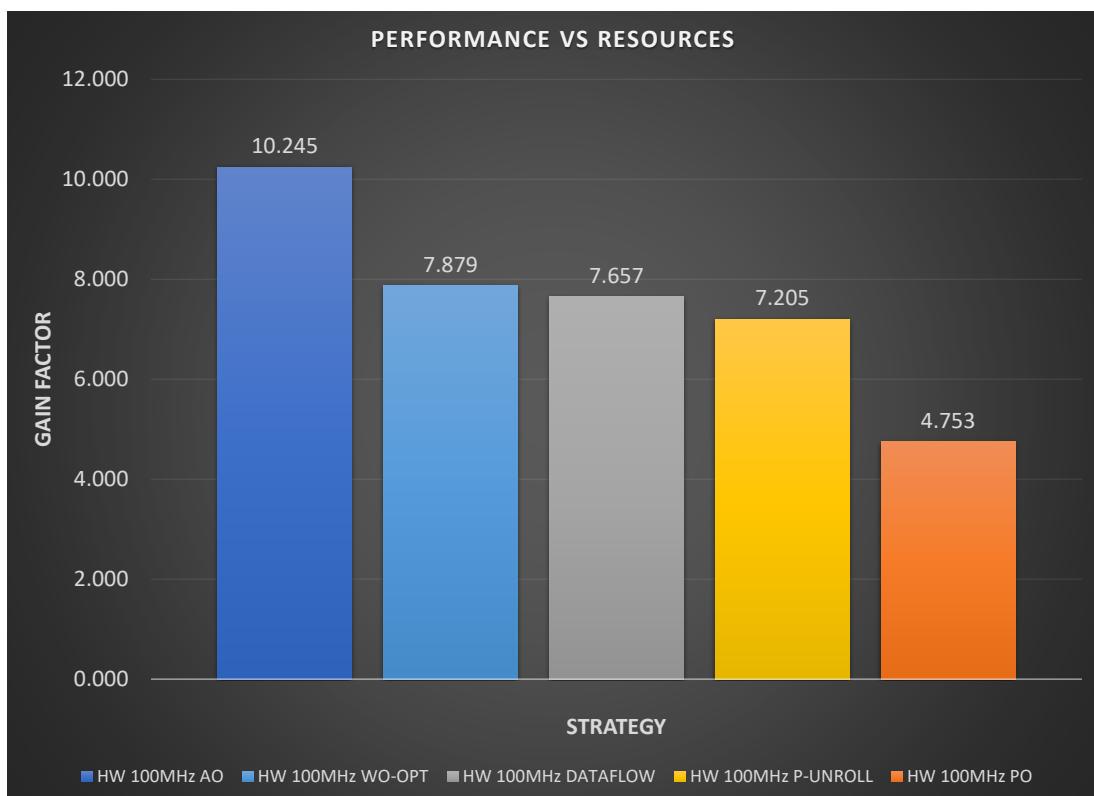


FIGURA 10.4: Factor de ganancia recursos contra tiempo

Capítulo 11

Conclusiones

La transformación de altos volúmenes de datos es viable usando dispositivos destinados orientados al Internet de las Cosas con una baja potencia computacional usando hardware. Con un diseño de hardware y las optimizaciones propuestas es posible realizar múltiples operaciones sobre los datos concurrentemente con un bajo footprint sobre el procesador, relegando únicamente la carga operativa a la FPGA y dejando el procesador atender los diferentes servicios de red que implican una conexión con la nube.

El diseño de hardware propuesto proporciona una ganancia de 4x la velocidad de cálculo sobre su contraparte software con el mismo diseño base, convirtiéndolo en una solución mucho más eficiente para el procesamiento de los datos en el borde de una red IoT.

Con las optimizaciones de área correctas es posible integrar múltiples módulos IP, que realicen diferentes operaciones y transformaciones sobre los datos sacrificando rendimiento que puede o no ser notable dependiendo de la complejidad y optimizaciones de hardware que requiera el circuito.

Durante la etapa de desarrollo inicial del hardware se determinó que el canal de datos de alta velocidad ofrecido por la interfaz AXI Full es el requerido para una aplicación donde el alto volumen de datos a procesar es un factor determinante para el rendimiento total de la solución. Esto se determinó luego de implementar correctamente el circuito con la interfaz AXI Lite, con la cual se obtuvo un rendimiento más bajo que la solución netamente software.

Se estableció que el ancho de bits del bus de datos del circuito hardware no influencia sobre la latencia total del circuito, con lo cual se pudo implementar un ancho de datos de 32 bits, garantizando una alta resolución para las operaciones sin perder rendimiento.

La implementación de una solución como la propuesta requiere de pasos muy precisos en la configuración del kernel, más aún si se desean características específicas como un File System de Ubuntu o módulos de hardware como una DMA para video o para mover grandes bloques de datos de una región a otra de memoria. Con una arquitectura como la propuesta es posible flashear la FPGA en tiempo de ejecución del Kernel de Linux y adicionar nuevos binarios y configuraciones al File System usando un servidor sftp y cliente como FileZilla.

Bibliografía

- [1] S. Li, L. Da Xu, and S. Zhao, “The internet of things: a survey,” *Information Systems Frontiers*, vol. 17, no. 2, pp. 243–259, 2015.
- [2] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, “Fog computing and its role in the internet of things,” in *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*, pp. 13–16, ACM, 2012.
- [3] F. Bonomi, R. Milito, P. Natarajan, and J. Zhu, “Fog computing: A platform for internet of things and analytics,” in *Big Data and Internet of Things: A Roadmap for Smart Environments*, pp. 169–186, Springer, 2014.
- [4] C. Perera, A. Zaslavsky, P. Christen, and D. Georgakopoulos, “Context aware computing for the internet of things: A survey,” *IEEE Communications Surveys & Tutorials*, vol. 16, no. 1, pp. 414–454, 2014.
- [5] J. Mattai and M. Joseph, *Real-Time Systems: specification, verification, and analysis*. Prentice Hall PTR, 1995.
- [6] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, “Fog computing and its role in the internet of things,” pp. 13–16, 2012.
- [7] Microsoft, “Pricing calculator — Microsoft Azure.” <https://goo.gl/rvv4Ql>, 2016. (Accessed on 11/01/2016).
- [8] Amazon, “Amazon web services pricing calculator.” <https://goo.gl/R5PwWq>, 2016. (Accessed on 11/02/2016).
- [9] Google, “Google cloud platform pricing calculator — google cloud platform.” <https://goo.gl/E4zCnk>, 2016. (Accessed on 11/02/2016).
- [10] Ubidots, “Ubidots pricing.” <https://goo.gl/a5vyym>, 2016. (Accessed on 11/02/2016).
- [11] B. Salami, G. MALAZGIRT, O. Arcas-Abella, A. Yurdakul, and N. Sonmez, “Axledb: A novel programmable query processing platform on FPGA,” *Microprocessors and Microsystems*, 2017.
- [12] Z. Liu, J. Großschädl, Z. Hu, K. Järvinen, H. Wang, and I. Verbauwhede, “Elliptic curve cryptography with efficiently computable endomorphisms and its hardware implementations for the internet of things,” *IEEE Transactions on Computers*, vol. 66, no. 5, pp. 773–785, 2017.

- [13] G. Mingas, L. Bottolo, and C.-S. Bouganis, “Particle mcmc algorithms and architectures for accelerating inference in state-space models,” *International Journal of Approximate Reasoning*, 2016.
- [14] F. Mehdipour, B. Javadi, and A. Mahanti, “Fog-engine: towards big data analytics in the fog,” pp. 640–646, 2016.
- [15] F. T. Mahmoudi, F. Samadzadegan, and P. Reinartz, “Object recognition based on the context aware decision-level fusion in multiviews imagery,” *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 8, no. 1, pp. 12–22, 2015.
- [16] A. Abrardo, M. Martalò, and G. Ferrari, “Information fusion for efficient target detection in large-scale surveillance wireless sensor networks,” *Information Fusion*, vol. 38, pp. 55–64, 2017.
- [17] I. Stojmenovic and S. Wen, “The fog computing paradigm: scenarios and security issues,” in *Computer Science and Information Systems (FedCSIS), 2014 Federated Conference on*, pp. 1–8, IEEE, 2014.
- [18] H. Madsen, B. Burtschy, G. Albeanu, and F. Popentiu-Vladicescu, “Reliability in the utility computing era: towards reliable fog computing,” in *2013 20th International Conference on Systems, Signals and Image Processing (IWSSIP)*, pp. 43–46, IEEE, 2013.
- [19] K. Hong, D. Lillethun, U. Ramachandran, B. Ottenwälder, and B. Koldehofe, “Mobile fog: A programming model for large-scale applications on the internet of things,” in *Proceedings of the second ACM SIGCOMM workshop on Mobile cloud computing*, pp. 15–20, ACM, 2013.
- [20] M. Aazam and E.-N. Huh, “Fog computing and smart gateway based communication for cloud of things,” in *Future Internet of Things and Cloud (FiCloud), 2014 International Conference on*, pp. 464–470, IEEE, 2014.
- [21] M. Aazam and E.-N. Huh, “Fog computing micro datacenter based dynamic resource estimation and pricing model for iot,” in *2015 IEEE 29th International Conference on Advanced Information Networking and Applications*, pp. 687–694, IEEE, 2015.
- [22] V. Cardellini, V. Grassi, F. Lo Presti, and M. Nardelli, “Distributed qos-aware scheduling in storm,” in *Proceedings of the 9th ACM International Conference on Distributed Event-Based Systems*, pp. 344–347, ACM, 2015.
- [23] V. Subramanian, D. Chan, A. K. Moghe, R. Pan, and F. Bonomi, “Dynamic coding for network traffic by fog computing node,” Jan. 5 2016. US Patent 9,232,433.
- [24] J. Zhu, D. Chan, M. M. S. PRABHU, P. NATARAJAN, and H. Hu, “Improving web sites performance using edge servers in fog computing architecture,” May 29 2013. US Patent App. 13/904,327.
- [25] G. Martin and G. Smith, “High-level synthesis: Past, present, and future,” *IEEE Design & Test of Computers*, no. 4, pp. 18–25, 2009.

- [26] Q. Huang, R. Lian, A. Canis, J. Choi, R. Xi, N. Calagar, S. Brown, and J. Anderson, “The effect of compiler optimizations on high-level synthesis-generated hardware,” *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, vol. 8, no. 3, p. 14, 2015.
- [27] A. Canis, J. Choi, M. Aldham, V. Zhang, A. Kammoona, T. Czajkowski, S. D. Brown, and J. H. Anderson, “Legup: An open-source high-level synthesis tool for fpga-based processor/accelerator systems,” *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 13, no. 2, p. 24, 2013.
- [28] C. Kao, “Benefits of partial reconfiguration,” *Xcell journal*, vol. 55, pp. 65–67, 2005.
- [29] P. Wehner, C. Piberger, and D. Göhringer, “Using json to manage communication between services in the internet of things,” in *Reconfigurable and Communication-Centric Systems-on-Chip (ReCoSoC), 2014 9th International Symposium on*, pp. 1–4, IEEE, 2014.
- [30] S. G. Owens, *Design modifications and platform implementation procedures for supporting dynamic partial reconfiguration of FPGA applications*. PhD thesis, Mississippi State University, 2013.
- [31] C. Liu, C. L. Yu, and H. K.-H. So, “A soft coarse-grained reconfigurable array based high-level synthesis methodology: promoting design productivity and exploring extreme fpga frequency,” pp. 228–228, 2013.
- [32] J. S. Monson and B. L. Hutchings, “Using source-level transformations to improve high-level synthesis debug and validation on FPGAs,” pp. 5–8, 2015.
- [33] D. Navarro, Ó. Lucí, L. A. Barragán, I. Urriza, Ó. Jiménez, *et al.*, “High-level synthesis for accelerating the fpga implementation of computationally demanding control algorithms for power converters,” *IEEE Transactions on Industrial Informatics*, vol. 9, no. 3, pp. 1371–1379, 2013.
- [34] G. A. Malazgirt, N. Sonmez, A. Yurdakul, A. Cristal, and O. Unsal, “High level synthesis based hardware accelerator design for processing sql queries,” pp. 27–32, 2015.
- [35] J. Choi, S. Brown, and J. Anderson, “From software threads to parallel hardware in high-level synthesis for fpgas,” pp. 270–277, 2013.
- [36] C. Alias, A. Darte, and A. Plesco, “Optimizing remote accesses for offloaded kernels: application to high-level synthesis for fpga,” pp. 575–580, 2013.
- [37] R. Zhao, M. Tan, S. Dai, and Z. Zhang, “Area-efficient pipelining for fpga-targeted high-level synthesis,” p. 157, 2015.
- [38] W. Liu, H. Chen, and L. Ma, “Moving object detection and tracking based on zynq fpga and arm soc,” in *IET International Radar Conference 2015*, pp. 1–4, IET, 2015.

- [39] M. M. Molla and S. I. Ahamed, "A survey of middleware for sensor network and challenges," in *Parallel Processing Workshops, 2006. ICPP 2006 Workshops. 2006 International Conference on*, pp. 6–pp, IEEE, 2006.
- [40] S.-H. Sho, K.-S. Kim, W.-r. Jun, J.-S. Kim, S.-H. Kim, and J.-D. Lee, "Ttcg: three-tier context gathering technique for mobile devices," in *Proceedings of the 5th international conference on Pervasive services*, pp. 157–162, ACM, 2008.
- [41] H. Sundmaeker, P. Guillemin, P. Friess, and S. Woelfflé, "Vision and challenges for realising the internet of things," *Cluster of European Research Projects on the Internet of Things, European Commision*, 2010.
- [42] D. Martin, C. Lamsfus, and A. Alzua, "Automatic context data life cycle management framework," in *Pervasive Computing and Applications (ICPCA), 2010 5th International Conference on*, pp. 330–335, IEEE, 2010.
- [43] F. Ramparany, R. Poortinga, M. Stikic, J. Schmalenstroer, and T. Prante, "An open context information management infrastructure-the ist-amigo project," 2007.
- [44] A. M. Bernardos, P. Tarrio, and J. R. Casar, "A data fusion framework for context-aware mobile services," in *Multisensor Fusion and Integration for Intelligent Systems, 2008. MFI 2008. IEEE International Conference on*, pp. 606–613, IEEE, 2008.
- [45] B. Schilit, N. Adams, and R. Want, "Context-aware computing applications," in *Mobile Computing Systems and Applications, 1994. WMCSA 1994. First Workshop on*, pp. 85–90, IEEE, 1994.