Diseño basado en modelos para la generación de *IP-Cores* enfocados en la industria 4.0

Tapia Puga, Luis Israel

Vicerrectorado de Investigación, Innovación y Transferencia de Tecnología

Centro de Posgrados

Maestría de Investigación en Electrónica

Trabajo de titulación, previo a la obtención del título de Magíster en Investigación en Electrónica, mención: Automática

Ing. Urbina Gamboa, Wilmer Marcelo, PhD.

21 de Octubre del 2021

Informe de originalidad

NOMBRE DEL CURSO
Revisión_Tesis

NOMBRE DEL ALUMNO
LUIS ISRAEL TAPIA PUGA

NOMBRE DEL ARCHIVO
LUIS ISRAEL TAPIA PUGA - Tesis_Luis_Tapia

CREACIÓN DEL INFORME
19 oct. 2021

Resumen

Pasajes marcados	0	0 %
Pasajes citados/entrecomillados	0	0 %

Firmado electrónicamente por:
WILMER MARCELO
URBINA GAMBOA

.....

Ing. Urbina Gamboa, Wilmer Marcelo, PhD.

C.C.: 1803293818

DIRECTOR

VICERRECTORADO DE INVESTIGACIÓN, INNOVACIÓN Y TRANSFERENCIA DE TECNOLOGÍA

CENTRO DE POSGRADOS

CERTIFICACIÓN

Certifico que el trabajo de titulación, "**Diseño basado en modelos para la generación de *IP-Cores* enfocados en la industria 4.0**" fue realizado por el señor **Tapia Puga, Luis Israel** el mismo que ha sido revisado y analizado en su totalidad, por la herramienta de verificación de similitud de contenido; por lo tanto cumple con los requisitos legales, teóricos, científicos, técnicos y metodológicos establecidos por la Universidad de las Fuerzas Armadas ESPE, razón por la cual me permito acreditar y autorizar para que lo sustente públicamente.

Sangolquí, 21 de Octubre del 2021

Firmado electrónicamente por:
WILMER MARCELO
URBINA GAMBOA

……

Ing. Urbina Gamboa, Wilmer Marcelo, PhD.

C.C.: 1803293818

ESPE
UNIVERSIDAD DE LAS FUERZAS ARMADAS
INNOVACIÓN PARA LA EXCELENCIA

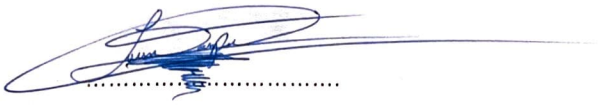
VICERRECTORADO DE INVESTIGACIÓN, INNOVACIÓN Y TRANSFERENCIA DE TECNOLOGÍA

CENTRO DE POSGRADOS

RESPONSABILIDAD DE AUTORÍA

Yo **Tapia Puga, Luis Israel**, con cédula de ciudadanía n° 1720966421 declaro que el contenido, ideas y criterios del trabajo de titulación: **Diseño basado en modelos para la generación de *IP-Cores* enfocados en la industria 4.0** es de mí autoría y responsabilidad, cumpliendo con los requisitos legales, teóricos, científicos, técnicos y metodológicos establecidos por la Universidad de las Fuerzas Armadas ESPE, respetando los derechos intelectuales de terceros y referenciando las citas bibliográficas.

Sangolquí, 21 de Octubre del 2021



Tapia Puga, Luis Israel

C.C.: 1720966421

VICERRECTORADO DE INVESTIGACIÓN, INNOVACIÓN Y TRANSFERENCIA DE TECNOLOGÍA

CENTRO DE POSGRADOS

AUTORIZACIÓN DE PUBLICACIÓN

Yo **Tapia Puga, Luis Israel,** con cédula de ciudadanía n° 17209966421, **autorizo** a la **Universidad de las Fuerzas Armadas ESPE** publicar el trabajo de titulación: **Diseño basado en modelos para la generación de *IP-Cores* enfocados en la industria 4.0** en el Repositorio Institucional, cuyo contenido, ideas y criterios son de mi responsabilidad.

Sangolquí, 21 de Octubre del 2021

.....

Tapia Puga, Luis Israel

C.C.: 1720966421

VICERRECTORADO DE INVESTIGACIÓN, INNOVACIÓN Y TRANSFERENCIA DE TECNOLOGÍA

CENTRO DE POSGRADOS

DEDICATORIA

Dedico esta tesis a mis padres, que me han animado y apoyado para conseguir mis objetivos académicos, a mi novia Paula, que siempre me alentó a no rendirme en ninguna circunstancia y me brindo apoyo moral durante el transcurso de mis estudios. Por último, quiero dedicar esta tesis a todos mis amigos, por apoyarme y extenderme su mano en todo momento. Les doy mi más sincero y cordial agradecimiento.

Sangolquí, 21 de Octubre del 2021

VICERRECTORADO DE INVESTIGACIÓN, INNOVACIÓN Y TRANSFERENCIA DE TECNOLOGÍA

CENTRO DE POSGRADOS

AGRADECIMIENTOS

Agradezco a la Universidad de las Fuerzas Armadas ESPE, y a su centro de estudios de posgrado, por permitirme ser parte del programa de becas y financiar mis estudios de posgrado. Agradezco al Dr. Marcelo Urbina principal colaborador durante todo este proceso, quien, con su paciencia, consejos, sabiduría y perseverancia, me permitió culminar con éxito este trabajo de tesis. También agradezco a mis demás profesores, quienes con su perseverancia transmitieron sus conocimientos de manera profesional, y forjaron en mí las bases de sus conocimientos, donde el fruto de su esfuerzo se ve reflejado en este trabajo de grado. Agradezco a mis padres, quienes han sido el motor que ha impulsado mis sueños y esperanzas, quienes estuvieron conmigo a lo largo de mi proceso estudiantil, dedico este logro a mis queridos padres como una meta más conquistada. Por último, pero no menos importante, agradezco a Dios por guiarme siempre en su camino, y por darme tranquilidad en muchas situaciones difíciles en las que he salido adelante y me ha permitido terminar mis estudios de manera satisfactoria**.**

Sangolquí, 21 de Octubre del 2021

Tabla de Contenido

Índice de Tablas

Índice de Figuras

Resumen

Desde el punto de vista tecnológico, la Industria 4.0 ha tomado gran relevancia en la actualidad, donde se pretende crear Sistemas Ciberfísicos *(CPS)* cada vez más sofisticados y eficientes. Por ello, el uso de *System On Chips (SoCs)* basados en *FPGAs* es una solución viable para generar aceleradores de *Hardware* (*IP-Cores*) compatibles con la Industria 4.0. El uso de *IP-Cores* constituye en la actualidad la forma más recomendada para diseñar *SoCs*, sin embargo, existe la problemática de que la mayoría de los *IP-Cores* disponibles en el mercado están enfocados a sistemas computacionales como *CPUs*, memorias y controladores de periféricos.

En este sentido, la propuesta de investigación presentada en esta tesis busca realizar contribuciones en el campo de los sistemas de control y de los sistemas embebidos, planteando una metodología de diseño basada en modelos que permita generar *IP-Cores* orientados a satisfacer las necesidades de la Industria 4.0, de forma que se puedan crear sistemas *CPS* embebidos capaces de contener en un único encapsulado, tanto módulos *Hardware* de sistemas de control automático, así como otros componentes básicos de un dispositivo computacional como controladores de periféricos, memorias y *CPUs*. Los IP-Cores modelados contendrán las estrategias de control automático, y estarán diseñados a partir de los controladores PID de uno y dos grados de libertad; útiles para gobernar CPS de brazos robóticos, utilizados en los procesos altamente automatizados.

Palabras Claves:

- **METODOLOGÍAS DE DISEÑO.**
- **SISTEMAS CIBERFÍSICOS.**
- **SISTEMAS Y APLICACIONES DE CONTROL INFORMÁTICO EMBEBIDO.**
- **IMPLEMENTACIÓN DE SISTEMAS INFORMÁTICOS EMBEBIDOS.**
- **CO-DISEÑO DE *HARDWARE*/*SOFTWARE*.**

Abstract

From the technological point of view, Industry 4.0 has taken great relevance nowadays, where the aim is to create increasingly sophisticated and efficient Cyber-Physical Systems (CPS). Therefore, the use of System On Chips (SoCs) based on FPGAs is a viable solution to generate Hardware accelerators (IP-Cores) compatible with Industry 4.0. The use of IP-Cores is currently the most recommended way to design SoCs, however, there is the problem that most of the IP-Cores available in the market are focused on computational systems such as CPUs, memories and peripheral controllers.

In this sense, the research proposal presented in this thesis seeks to make contributions in the field of control systems and embedded systems, proposing a model-based design methodology that allows generating IP-Cores oriented to meet the needs of Industry 4.0, so that embedded CPS systems can be created capable of containing in a single encapsulation, both Hardware modules of automatic control systems, as well as other basic components of a computational device such as peripheral controllers, memories and CPUs. The modeled IP-Cores will contain the automatic control strategies, and will be designed from PID controllers of one and two degrees of freedom; useful to govern CPS of robotic arms, used in highly automated processes.

Keywords:

- **DESIGN METHODOLOGIES.**
- **CYBER-PHYSICAL SYSTEMS.**
- **EMBEDDED COMPUTER CONTROL SYSTEMS AND APPLICATIONS.**
- **IMPLEMENTATION OF EMBEDDED COMPUTER SYSTEMS.**
- ***HARDWARE/SOFTWARE* CO-DESIGN.**

Lista De Acrónimos

$\%\mathbf{MP}$	Porcentaje de sobre impulso
1DoF-PID	One-Degree-of-Freedom PID
2DoF-PID	Two-Degree-of-Freedom PID
AGV	Automatic Guided Vehicle
AHB	Advanced High-performance Bus
AI	Artificial Intelligence
AMBA	Advanced Microcontroller Bus Architecture
APB	Advanced Peripheral Bus
AVR	Augmented and Virtual Reality
AXI	Advanced eXtrensible Interface
BLE	Bluetooth Low Energy
CAD	Computer-Aided-Design
CAM	Computer-Aided-Manufacturing
CAPP	Computer Aided Process Planning
CIM	Computer-Integrated-Manufacturing
CLB	configurable logic blocks
CPS	Cyber Physical System
DSP	Digital Signal Processing
$\mathbf{e_{ss}}$	Error de estado estable
FPGA	Field Programmable Gate Arrays
HIL	Hardware in The Loop
HPC	High Performance computing
IIoT	Industrial Internet of Things

IoS	Internet of services,
IoT	Internet of Things
IP-Core	Intellectual Property Cores
IWN	Industrial Wireless Network
LoRaWAN	Long-Range Wide-Area Network
LUTs	Look Pp Tables
MpSoC	Multiprocessor System-on-Chip
NoCs	Networks On Chips
NRE	Non-Recurring-Engineering
PID	Proportional-Integral-Derivative
PWM	Pulse-width modulation
RAM	Random Access Memory
RISC	Reduced Instruction Set Computing
ROM	Read Only Memory
RPM	Revoluciones Por Minuto
RTL	Register-Transfer Level
SoC	System on Chip
SPI	Serial Peripherical Interface
SXG	Xilinx System Generator
$\mathbf{t_{ss}}$	Tiempo de establecimiento
$\mathbf{TV_u}$	Esfuerzo de control
UART	Universal Asynchronous Receiver-Transmitter
USB	Universal Serial Bus
VHDL	Lenguaje de Descripción de Hardware para circuitos integrados de muy alta velocidad

Introducción

La iniciativa de la industria 4.0 impulsa sistemas distribuidos, altamente automatizados y redes de producción dinámicas con tecnología del internet de las cosas (*Internet of Thing*, *IoT)*, como la robótica, las interfaces hombre-máquina, la inteligencia artificial y el *Software* de código abierto (Pereira & Romero, 2017). La automatización se logra mediante la utilización de los sistemas ciber-físicos (*Cyber Phisical Systems*, *CPS*) interconectados, es decir, las entidades de estos sistemas deben tener un comportamiento inteligente y deben comunicarse entre sí para lograr un objetivo en común (Alladi et al., 2019). Por otra parte, el desarrollo acelerado de la tecnología eléctrica, electrónica y los métodos de producción altamente modernizados, ha permitido que los sistemas evolucionen y mejoren su capacidad e inteligencia. Es así que hoy en día se ha logrado desarrollar y condensar sistemas completos en un solo encapsulado, conocidos como *System on Chip* (*SoC*), los cuales integran una serie de bloques funcionales en un espacio físico reducido.

Los *SoCs* son sistemas rentables y enérgicamente eficientes, características importantes en una era de inteligencia computacional integrada (Yudi et al., 2017). Esto permite que sean candidatos ideales para generar sistemas *CPS* (Lee et al., 2015), además, por su versatilidad pueden ser compatibles con la tecnología *IoT*, computación en la nube, y en conjunto son clave en la habilitación de la industria 4.0 (Chen et al., 2017; O'Donovan et al., 2018). Los principales bloques que componen un *SoC*, son los núcleos de propiedad intelectual (*Intelectrual Property Cores, IP-Cores*) (Mohamed, 2015). Los *IP-Cores* son tejidos de *Hardware* pre-verificados que cumplen con una función específica y están listos para ser integrados en el diseño de un *SoC*, esto los convierte en un instrumento obligatorio para m*axi*mizar la productividad a la hora de diseñar un *SoC*, reduciendo el desarrollo y el tiempo de comercialización (Riahi et al., 2005; Sengupta &

Kachave, 2018). La capacidad del *IP-Cores* de tener un estándar de comunicación es lo que lo diferencia de otros tejidos de *Hardware* desarrollados en *FPGA,* que funcionan de forma aislada y autónoma (*Stand-Alone*). En la actualidad, el desafío está en desarrollar *IP-Cores* que permitan la construcción de nuevos *SoC* especializados que cubran los requerimientos de la industria 4.0, es decir que permitan una gran escalabilidad de los sistemas, procesamiento masivo de datos, sistemas comunicados, controlados y automatizados (Aheleroff et al., 2020; Xu et al., 2018a). Por tal motivo existe la necesidad de modelar *IP-Cores* enfocados en áreas de la manufactura, robótica, sistemas de comunicación, inteligencia artificial y sistemas de control avanzados, que aporten con el desarrollo de la cuarta revolución industrial.

Cabe destacar que los sistemas de comunicación, los vehículos aéreos no tripulados, el control automotriz y las centrales eléctricas dependen cada vez más de sistemas embebidos, debido en gran medida a su reducido tamaño, coste y versátil programación. La mayoría de estos sistemas embebidos están diseñados a partir de *SoCs*, los cuales para reducir el costo de desarrollo y de Ingeniería-No-Recurrente (*Non-Recurring-Engineering, NRE*), hacen uso de los núcleos de propiedad intelectual. Esto abre un abanico a la hora de ofrecer nuevos componentes a los diseñadores *IP-Cores* (Hategekimana et al., 2018). Los *SoCs* modernos son capaces albergar en un solo encapsulado: módems, procesadores de señales digitales, núcleos heterogéneos y aceleradores de aplicaciones específicas para incrementar la eficiencia a un sistema. Es así que se podría generar *CPS* a partir de *SoCs* creados por *IP-Cores* enfocados en la industria 4.0, capaces de cubrir necesidades de control automáticos para los dispositivos que forman parte de un sistema altamente controlado y automatizado.

El incremento del desempeño de sistemas que hacen uso de la tecnología de *SoCs*, se debe a que el conjunto de *IP-Cores* que trabajan como procesadores de *Hardware* se ejecutan de manera concurrente e independientes entre sí; de modo que se

puede conseguir construir plataformas completas que como se menciona anteriormente, albergan procesadores, drivers de periféricos y un número determinado de controladores que gobiernan diferentes plantas de manera paralela. Esto permite que los procesadores liberen carga computacional, de modo que, los procesadores se encarguen de realizar la iniciación y calibración de parámetros de controlador; sin embargo, su prioridad es atender rutinas que necesitan una gran cantidad de recursos de procesamiento, como lo es la ejecución de algoritmos de inteligencia y visión artificial, dejando el trabajo de control automático de otros *CPS* a tejido de *Hardware* dedicado. Es así que, se puede aprovechar el espacio de *Hardware* disponible para generar *IP-Cores* que cumplan la función de gobernar sistemas con estrategia de control industriales robustas implementadas como *IP-Cores,* modelados a partir de estrategias de control clásicas como los controladores de acción Proporcional-Integral-Derivativa *(Proportional-Integral-Derivative, PID),* que pueden ser de uno o dos grados de libertad; como resultado se obtendrá un sistema *SoC* que no solo contenga elementos computacionales, sino que posea un tejido de *Hardware* especializado para gobernar *CPS* que formarán parte de la industria 4.0

Antecedentes

Para hacer realidad los retos planteados por la nueva revolución industrial, conocida como industria 4.0, se exige la incorporación de *CPS* altamente inteligentes, con un *Hardware* robusto y capaz de soportar gran capacidad de procesamiento; es así que los aceleradores de *Hardware* son la base ideal para construir tales sistemas. El desafío actual de diseños complejos y la creciente presión del mercado ha provocado que el uso de *IP-Cores* reutilizables, sea una solución sostenible que involucre un nivel de síntesis de alto nivel, la misma que puede ser protegido para asegurar la propiedad intelectual de los diseños (Sengupta et al., 2016). El mercado de *IP-Cores* resulta un interesante tema de investigación, poco explorado, pero que a su vez cumple un papel

vital en el desarrollo de la industria de los semiconductores. Actualmente, el mercado de *IP-Cores* está mayoritariamente enfocado a la tecnología computacional, y existe un gran crecimiento del uso de *IP-Cores*; es así que desde el 2017 todos los chips de computadoras contienen al menos un *IP-Cores* de terceros, mientras que en el 2013 apenas el 50% de los chips contenía esta tecnología (Distel, 2017). Por tal motivo, es necesario afrontar esta tendencia para que los ingenieros diseñadores de sistemas electrónicos dispongan de una metodología para diseñar e implementar *IP-Cores*, pero enfocados a cubrir los requerimientos de la industria 4.0, que satisfagan las necesidades de producción y de manufactura, a través de sistemas de control y de comunicación acelerados por *Hardware*.

En la literatura, se pueden encontrar trabajos que abordan el diseño de *SoC*, los cuales cubren algunas aplicaciones en la industria; destacando su alta flexibilidad, capacidad de adaptarse a distintos requerimientos y restricciones ante fallas (Baklouti et al., 2015). Por tal motivo, esta tecnología puede ser aprovechada para: crear dispositivos, módulos de producción y controladores, que puedan intercambiar información de forma independiente y que permitan crear un entorno de fabricación inteligente. La posibilidad de implementar múltiples núcleos de propiedad intelectual en un mismo encapsulado, ha logrado acelerar el procesamiento de algoritmos de inteligencia artificial; lo que contribuye a impulsar otro pilar fundamental para el desarrollo de la industria 4.0. Un ejemplo es la plataforma *SoC* Alpha, presentado en (Perkovic et al., 2017), la cuál es un *SoC-Multi-core* que implementa algoritmos de visión artificial, utilizada en aplicaciones de conducción autónoma en el sector automotriz. La plataforma Alpha al estar basada en un acelerador por *Hardware,* ejecuta múltiples tareas en paralelo, convirtiéndola en un dispositivo de tiempo real, robusto y tolerante a fallos; aspectos muy importantes para un sistema crítico de conducción autónoma y segura.

De igual manera, los sistemas de comunicación presentan una gran relevancia en los sistemas automatizados; en la literatura se puede hallar diseños de *SoC* a partir de *IP-Cores* que pueden integrar funciones, que gestionan interfaces de comunicación. Existen interfaces que administran la comunicación para que exista un intercambio de información entre dispositivos *IoT,* e industriales. Como resultado de la implementación de un *Gateway* de protocolos I2C a POWERLINK (Valente et al., 2019), se determinó que el diseño utilizó aproximadamente un 17% de los recursos totales de una *FPGA Intel Cyclone IV EP4CE115F29C7*.

Otras propuestas se centran en comunicar *CPS* industriales, a través de protocolos de alta disponibilidad y redundancia. En el trabajo presentado por (Urbina et al., 2017), donde muestra que al implementar un *Gateway* para administrar *Profibus, Modbus* y *Profinet*, se utilizaron 3640 *LUTs*, 4196 registros y 36 bloques de memoria *RAM*, los cuales representan menos del 25% de los recursos totales en una *FPGA Xilinx Zynq XC7Z020*, esto permite reservar recursos para otros diseños que se pueden sintetizar en el *Hardware* restante. El uso de *IP-Cores* simplifica drásticamente el diseño de sistemas complejos, esto permite que los aceleradores de *Hardware* crezcan dos veces más rá*PID*o en el mercado global de semiconductores. En años pasados, para la creación de *SoCs* se utilizaba una lógica nueva en cada diseño, sin embargo, esta tendencia disminuyó de un 45% a un 42% entre el año 2012 y 2014; mientras que el uso de *IP-Cores* para la formación de nuevos *SoCs* incrementó de un 39% a un 59% (Rodriguez-Andina et al., 2015). En esta nueva tendencia se busca crear aceleradores de *Hardware* de tipo *Soft-Core* estandarizados para la creación de sistemas computacionales embebidos en un solo encapsulado, dejando atrás la metodología de realizar un diseño nuevo en cada proceso de creación de un *SoC.*

Las fábricas inteligentes garantizan que los procesos sean completamente controlados y automatizados; se pueden encontrar una variedad de trabajos donde

abordan el diseño de moduladores por ancho de pulso *(Pulse-width modulation, PWM*) sintetizadas en *FPGAS*, los cuales pueden ser utilizados en aplicaciones de control para convertidores de potencia y *drivers* de motores. Sin embargo, son pocas las investigaciones que abordan sus diseños dentro de un *IP-Core,* el cual pueda ser reutilizable para satisfacer los requerimientos de un *SoC*. En el trabajo presentado por (Diao et al., 2018), se implementó un módulo *PWM* sintetizado en *Hardware* de tipo *Stand-Alone*, el diseño carece de un mecanismo para complementarse de manera sencilla con otros módulos de *Hardware*; este trabajo es relevante debido a que es necesario comprender, que implementar un diseño en *Hardware* de tipo *stand-alone* no siempre es la mejor opción; puesto que, utilizar sistemas de este tipo podrían ocasionar que se tenga una *FPGA* por cada diseño, consecuentemente se incrementaría el tamaño de una placa electrónica. Por otro lado, si se desea implementar los diseños en una sola *FPGA*, que, al no poseer un estándar de comunicación entre los bloques de *Hardware*, dificultaría su integración en un solo *System On Chip*. En definitiva, el uso de *IP-Cores* permite facilitar la construcción de un *SoC* que mantenga todo el sistema en un solo encapsulado. El trabajo (Corna et al., 2019) se destaca, porque presenta una contribución de un *IP-Core* que modela un *PWM,* que a diferencia de los diseños *Stand-Alone*, el producto final es un acelerador de *Hardware* que contempla las interfaces *ARM* Advanced eXtensible Interface (*AXI*); las cuales permiten integrarse a otros *IP-Cores*, incluso se puede integrar hasta 64 canales del mismo *IP-Core* con una frecuencia máxima de 133Mhz. Otra ventaja que trae consigo esta tecnología es la capacidad de formar *SoCs* industriales de manera más eficiente y productiva, con la capacidad de intercambiar información con más dispositivos ciber-físicos, por tal motivo, la tendencia actual es evitar crear sistemas embebidos que sean totalmente aislados.

Algo similar sucede en los trabajos que involucran sistemas de control, puesto que existe una gran cantidad de estudios que abordan controladores para aplicaciones de

automatización industrial basados en *FPGA*. Los estudios expuestos por (Abdelati, 2016) y (Indira & Swapna, 2017) mostraron las ventajas de implementar un controladores *PID* de un grado de libertad sintetizados en *Hardware,* debido a que poseen una mayor velocidad de procesamiento de las señales. Sin embargo, estas propuestas continúan sin ser la mejor opción para utilizarlo en una metodología de diseño *SoC* moderno, ya que los resultados presentados en estos trabajos son diseños de tipo *Stand-Alone*; es decir que, dichos diseños, aunque pueden volverse a reutilizar, no cuentan con un sistema de intercomunicación con otros *IP-Cores* para formar un sistema computacional más avanzado. Esto provocaría que el tiempo de desarrollo y el costo de ingeniería no recursiva para construir un sistema embebido sea mucho más alto. La solución presentada por (Youness et al., 2014) , muestra una propuesta interesante de controladores *PID* de un grado de libertad embebidos para motores DC utilizando tecnología de *Multi-Procesor System on Chip* (*MPSOC*) y procesadores multinúcleos basados en *FPGAS*; la investigación expone las ventajas de *AMBA*s tecnologías, por un lado, el sistema MP*SOC* es eficiente debido a que el *Hardware* es capaz de adaptarse al sistema de control y se aprovecha tanto los recursos de *Hardware* como de *Software*, aunque esto podría generar un incremento del esfuerzo y tiempo de diseño. Por otro lado, la tecnología de procesadores multinúcleo puede garantizar un buen performance con esfuerzo y tiempo de diseño más corto, pero el *Hardware* no puede ser personalizado para sacar alguna ventaja adicional; como resultado de la experimentación se determinó que, las tecnologías *MPSOC* y procesadores multinúcleos permiten que los tiempos de ejecución de tareas se acelere hasta casi un 75% en comparación a una implementación tradicional de monoprocesador con ejecución secuencial de las tareas. Cabe mencionar que el trabajo presenta una validación del sistema modelado a través de la simulación *Hardware In Loop (HIL*), con la finalidad de aprovechar los recursos de *Hardware* para realizar las verificaciones y observaciones del comportamiento del sistema diseñado. Algo

similar se puede observar en los trabajos (Yang et al., 2018) y (Alvarez-Gonzalez et al., 2017), los cuales realizaron una validación en sus estudios a través de esta metodología *HIL*, que fue utilizada para simular fallas en tiempo real en un sistema de control de tracción y en un sistema de motores de inducción implementados en *FPGAS* respectivamente. El resultado expuesto por (Alvarez-Gonzalez et al., 2017), indica que el máximo *Slack Time* no superaró los $0.104\,ns$; además, mediante las distintas simulaciones se analizaron diferentes fallas de rotor y pérdida de fase, el error entre la precisión de punto flotante en la simulación fuera de línea frente a la simulación *HIL* con punto fijo resultó de 0.1415 km/h en la velocidad del rotor. Se concluyó que el sistema poseía un comportamiento parecido en ambos escenarios de simulación, sin embargo, no realizaron un análisis del efecto que se tendría al utilizar una precisión de punto flotante en *Hardware* y cuál sería el costo de recursos que podría causar utilizar este tipo de precisión. Las propuestas de estos dos últimos trabajos no consideran el beneficio de crear un *IP-Core* que sea reutilizable en diseños más avanzados de tipo *SoC*, pero muestran los beneficios que poseen los controladores sintetizados en *Hardware* mediante la simulación *HIL*.

Es conveniente aclarar, que a pesar de que existe una gran cantidad de trabajos basados en diseños *Stand-Alone*, se pueden encontrar otras propuestas, como la presentada por (Huang et al., 2017); que expone un diseño de *IP-Core* capaz de controlar la posición y velocidad de un motor para aplicaciones de robótica humanoide o industrial, a partir de un controlador PI y un módulo *PWM* implementados en conjunto. Se determinó que el tiempo en que el controlador alcanza la velocidad deseada en el motor de 1000 RPM es de 1300 ms; sin embargo, existe un tiempo muerto de 750 ms debido al intercambio de información del computador host al *SoC*, con dicha velocidad de 1000 RPM el eslabón puede alcanzar una posición de 10000 pulsaciones en 1800 ms que no afecta en gran medida el funcionamiento del sistema. El objetivo final de este trabajo fue

crear un *SoC* compuesto por varios de los *IP-Cores* que controlen los diferentes eslabones de un brazo articulado, y que a su vez estén comunicados entre sí a través de una interfaz *AXI*, lo que facilita implementar de manera modular y sencilla al *SoC*. El diseño propuesto permite el control de manera concurrente de los motores, ubicados en los ejes de los eslabones del robot; sin embargo, los autores comentan que el trabajó exhibió algunos problemas relacionados con el control de velocidad, que presentaba una fluctuación antes de estabilizarse, debido a la ganancia integral del compensador; como trabajo futuro se plantearon crear un algoritmo de control más preciso. El desempeño del *IP-Core* podría mejorar al utilizar un *Hardware* capaz de manejar precisiones de punto flotante o punto fijo con mayor resolución, adicionalmente se podría considerar algoritmos de control más robustos, como los controladores *PID* con filtros derivativos y de dos grados de libertad que puedan mejorar el desempeño y robustez del sistema. Resulta interesante observar que en esta propuesta el uso de *IP-Cores* puede implicar en un *Hardware* mejor aprovechado, dejando recursos de *Software* que se pueden dedicar a realizar otras funciones relacionadas a la industria 4.0, como procesar algoritmos de inteligencia artificial a través de su procesador de doble núcleo. Es preciso mencionar que si bien el modelo de *IP-Core* facilita su integración en el *SoC* creado, se podría mejorar el diseño otorgándole una mayor flexibilidad al acelerador de *Hardware*, produciendo que las ganancias proporcionales e integrales del controlador puedan ser adaptadas para que el *IP-Core* sea utilizado en un diseño *SoC* de terceros.

En cuanto al desarrollo tecnológico de los dispositivos enfocados a la industria 4.0 y al internet de las cosas industriales (*Industrial Internet of Thing, IIoT*), se abordan dispositivos como sensores y actuadores que tengan conectividad, y un *Software* capaz de incorporar: aprendizaje automático, nube informática, tecnología de *big data* y tecnología de automatización (Cigánek et al., 2018). Existe un modelo de componentes para la industria 4.0 denominada *I4.0 component*, desarrollado por *BITCOM, VDMA y*

ZWEI (Zezulka et al., 2016), el cual está destinado a ayudar a los desarrolladores e integradores de sistemas para crear *Hardware* y *Software* enfocada a la industria 4.0. La pieza más importante que este modelo aborda es la capacidad de comunicación que debe existir entre procesos virtuales y los dispositivos o procesos de producción físicos. El estudio presentando por (Garrido-Hidalgo et al., 2019), aborda las formas más comunes de comunicación entre dispositivos *IoT* y muestra una serie de mejoras sobre las tecnologías *Long-Range Wide-Area Network (LoRaWAN)* y *Bluetooth Low Energy (BLE),* que facilitan la gestión de la información entre los dispositivos que conforman la industrial 4.0, y puede llegar a tener latencias máximas de 185 ms en un sistema de identificación de productos y clasificación de piezas, lo que no representa ningún problema en el rendimiento general del sistema.

Actualmente la industria 4.0 se centra en la creación de productos y procesos inteligentes, esto hace que los diseñadores tengan que afrontar un desarrollo tecnológico más eficaz. El trabajo (Nunes et al., 2017) aborda un enfoque para mejorar los dispositivos inteligentes dirigidos a la industria 4.0; el objetivo del desarrollo de estos nuevos productos es integrar los requisitos de ingeniería y diseño industrial a través de un proceso estructurado, que permita la construcción de dispositivos a un menor costo, mayor calidad y menor tiempo de ingeniería, esto incluye la optimización de recursos y eliminación de desperdicios. Por otro lado, se menciona que los dispositivos orientados a la industria 4.0, deben ser independientes y a su vez que cooperen con otros sistemas; a través de una red integrada por diferentes equipos inteligentes, los cuales en conjunto pueden realizar acciones en tiempo real y permiten una intercomunicación entre el mundo físico y el informático. Los *SoC* basados en *FPGA* pueden ser candidatos ideales para formar este tipo de dispositivos, debido a que pueden integrar múltiples *IP-Cores* que cumplan con estos requisitos; adicionalmente esta tecnología se ha convertido en

tendencia en el desarrollo de nuevos productos, debido a los avances tecnológicos que permiten aumentar el número de transistores dentro de un encapsulado.

En base a la revisión del estado del arte, la mayoría de los trabajos se centran en el desarrollo de drivers de motores (Controlador y driver *PWM*), que resalta demostrar la superioridad de desempeño que tiene el uso de las *FPGAS* frente a los procesadores; lamentablemente la mayoría de estas aplicaciones son de tipo *Stand-Alone*, por lo que carecen de mecanismos que permitan su integración en un diseño de *SoC* moderno. El tipo de controlador implementado en las investigaciones son *PI*/*PID* de un solo grado de libertad, que a pesar de ser uno de los controladores industriales más utilizados, no es la mejor opción para ser utilizados en sistemas servo-controlados. Esto provoca que la aplicación de estos diseños sea reducida, elevando el costo *NRE* en el proceso de crear *SoCs* modernos que cubren otras aplicaciones industriales. Esto motiva a la presente investigación a desarrollar una metodología que permita modelar, verificar y sintetizar *IP-Cores* de tipo *Soft-Core;* generados a partir de modelos relacionados a la ingeniería de control automático, que puedan ser parte de un *CPS,* pieza clave para la habilitación de la industria 4.0.

Por consiguiente, el estudio está centrado en desarrollar una metodología de diseño de *IP-Cores* basados en modelos enfocados a la industria 4.0, como caso aplicativo de la metodología, el *IP-Core* estará construido a partir de modelos de controladores *PID* de uno y dos grados de libertad, puesto que un estudio reciente expone la premisa de que el controlador *PID* de uno y dos grados de libertad, juega un rol significativo en los procesos de automatización, y es utilizado en una amplia gama de aplicaciones industriales (Bauer et al., 2016); lo que los convierte en buenos candidatos para integrarse a un entorno de control en la industria 4.0. Como se ha revisado en la literatura, la mayoría de trabajos relacionados a controladores abordan estrategias *PI*/*PID* de un grado de libertad (*One-Degree-of-Freedom, 1DoF-PID*), ignorando las ventajas que

tiene un controlador *PI*/*PID* de dos grados de libertad (*Two-Degree-of-Freedom*, *2DoF-PID*) modelado como un *IP-Core*. Hay que resaltar que la estrategia *2DoF-PID* permite mantener un buen comportamiento transitorio ante cambios de consigna en sistemas servo-controlados, esto debido a que el controlador de dos grados de libertad, considera parámetros adicionales que permite mejorar el comportamiento y robustez en lazo cerrado de un sistema servo-controlado (Deshmukh & Kadu, 2017); el cual bien podría ser utilizado para gobernar de manera concurrente sistemas ciber-físicos compuestos por brazos robóticos cuyos eslabones son accionados por servo-motores.

La planta en la que se va aplicar el estudio del modelo de controlador sintetizado como *IP-Core* será un motor DC o servo-motor, modelado a través de principios físicos; en dicho mecanismo servo-controlado se evaluará el comportamiento de lazo cerrado de estas estrategias de control modeladas en *Hardware* (*1DoF-PID* y *2DoF-PID*). El uso de estos sistemas se los puede encontrar en dispositivos *CPS* que gobiernan brazos robóticos; usualmente, se presentan en procesos de automatización de la industria 4.0. Al tratarse de un tejido de *Hardware* que se utilizará para realizar procesamiento de señales, como lo efectúa el controlador *PID* en un sistema de lazo cerrado, se debe tomar en cuenta el tipo de datos que se va a manipular en *Hardware* (booleanos, enteros, reales), y el número de bits utilizados en la precisión de las operaciones computacionales; puesto que, estos factores afectan directamente al costo de recursos de *Hardware* y podrían influir en el desempeño del controlador en lazo cerrado ($t_{ss}, \%M, e_{ss}\ y\ TV_u$, robustez ante cambios de consigna). Es así que, como parte de la investigación, se analizará el desempeño del controlador sintetizado como *IP-Core* con distintos tipos de precisión (punto fijo o punto flotante) y cómo afecta en el uso de recursos de *Hardware* (*Registers, LUTs, BRAMs, DSPs, Slack Path Time, Delay Path Time y el Routing Delay Time*). Como resultado, los modelos de controladores en forma de *IP-Core*, tendrán la ventaja de tener la capacidad de complementarse y comunicarse con otros *IP-Cores* a

través de un estándar de comunicación *AXI*, el cual permite que exista un intercambio de información entre diferentes núcleos de propiedad intelectual; además de ser lo suficientemente flexible para modificar ciertos parámetros del *IP-Core* como las ganancias del controlador. En otras palabras, el *IP-Core* modelado no sólo será útil como parte de una solución en la aplicación industrial, si no que dispondrá de la capacidad para ser parte de un sistema computacional embebido más grande, compuesto de otros *IP-Cores*, como: drivers de comunicación industrial, procesadores, controladores de periféricos, elementos que forman parte de un diseño *SoC* y que a su vez cumplan con los requerimientos de los dispositivos ciber-físicos orientados a la industria 4.0. La contribución del presente trabajo será:

- Ayudar a los diseñadores de *SoCs* modernos, generar plataformas industriales que puedan albergar tejido de *Hardware* dedicado, capaz de realizar procesamientos específicos que cumplan con las necesidades de la industria 4.0.
- Ayudar en la ingeniería de control a afrontar los nuevos retos tecnológicos, para desarrollar modelos de controladores robustos, tolerantes a fallas y con capacidad de operación en tiempo real enfocados a procesos industriales acelerados por *Hardware* y fáciles de ser implementados en un diseño *SoC*.
- Mostrar las ventajas de desempeño de los controladores *PID* de uno y dos grados de libertad sintetizado en *Hardware* en un sistema de lazo cerrado servo-controlado frente a los algoritmos tradicionales ejecutados por *Software*.
- Generar un recurso de *Hardware* modelado como un *IP-Core*, capaz de poderse complementar con otro tipo de S*oft-Cores* (procesadores, drives de comunicación, drivers de periféricos, memorias) con el fin de crear sistemas computacionales embebidos que puedan formar parte de los sistemas ciber-físicos de la industria 4.0.

Formulación del problema

Al revisar la línea base y estado del arte se ha observado falta de diseños flexibles de *IP-Cores* orientados a la industria 4.0, basadas en una metodología de modelamiento. A su vez, es necesario realizar una evaluación del sistema diseñado, esto se lo puede realizar a través de la metodología *TOP-DOWN*.

La metodología *TOP-DOWN* involucra un alto nivel de abstracción para realizar el modelamiento, pero al descender a niveles más bajos de abstracción se puede realizar la simulación, verificación y síntesis del modelo en *Hardware*, la propuesta se encuentra descrita con mayor detalle en la sección de Metodología.

Hasta ahora, la industria 4.0 se ha compuesto de varias tecnologías sofisticadas, la mayoría son actuadores y sensores con características especiales de cómputo y comunicación. A su vez, los controladores que gobiernan estas tecnologías están basadas en procesadores; sin embargo, los beneficios proporcionados por la tecnología basada en *FPGAS,* pueden obtener las mismas funciones que los sistemas basados en procesadores, y acelerados por *Hardware*. Los *System on Chips* son una solución tecnológica funcional y sofisticada que pueden contribuir al desarrollo de la industria 4.0; esta tecnología puede integrar un sistema computacional completo dentro de un solo encapsulado y la mejor manera de crear *SoCs* es mediante el uso de *IP-Cores*. Por esta razón es necesario mantener un catálogo de *IP-Cores* que permita que los diseñadores de sistemas embebidos, cuenten con las herramientas necesarias para crear *SoCs* en menor tiempo y garantizando su funcionamiento. A su vez, los nuevos *SoCs* orientados a la industria 4.0 serán los que se conviertan en *CPS,* componentes principales para la habilitación de la industria 4.0.

Justificación e Importancia

En las investigaciones evidenciadas en los antecedentes y estado del arte, se observa que el mercado actual de los *IP-Cores* está orientado en su mayoría a los sistemas computacionales, y una minoría está orientada a la industria. Esta minoría a su vez está limitado a drivers de motores, comunicaciones industriales y otras aplicaciones enfocadas en la industria automotriz; de manera similar, los controladores contemplados suelen ser solo de un grado de libertad, esto provoca que controladores *2DoF-PID* sean prácticamente ignorados para ser incorporados como *IP-Cores*. La falta de una metodología de diseño para generar *IP-Cores* que cubran aplicaciones de plantas servo-controladas, provoca que exista una escasez de recursos de *Hardware* que im*PID*e a los diseñadores de sistemas embebidos generar *SoCs* que satisfagan a toda una gama de aplicaciones industriales, tales como: la alimenticia, farmacéutica, química entre otras; las cuales están pasando o pasarán por la transición de la tecnología de industria 3.0 a 4.0. Por otro lado, si se considera que desde el año 2014 más del 55% de los diseños *SoC* hacen uso de *IP-Cores* y desde el 2017, todos los dispositivos *SoCs* contienen al menos un *IP-Core*; por lo que es necesario seguir trabajando para mantener esta tendencia y generar *IP-Cores* que satisfagan las necesidades de la nueva era de manufactura industrial.

EL propósito consiste en tener las herramientas y metodologías de diseño para la construcción de *IP-Cores* orientadas a crear *CPS* con controladores acelerados por *Hardware*, lo que permitiría mejorar las respuestas transitorias de los sistemas servo-controlados, los cuales pueden funcionar correctamente en lazo cerrado con un controlador *PID* de dos grados de libertad.

La presente propuesta de investigación plantea entregar una estrategia o metodología para el diseño y pre-verificación de *IP-Cores,* que responda a las

necesidades de tener un sistema de control robusto, para satisfacer las necesidades de la industria 4.0. Es decir, como resultado final se desea obtener un *IP-Core* flexible, genérico y capaz de ser utilizado en múltiples aplicaciones de control automático, como lo son las aplicaciones de sistemas servo-controladas. El modelo estará validado a través del desempeño del controlador sintetizado en *Hardware* a través de parámetros de diseño como tiempo de establecimiento (t_{ss}), el porcentaje de sobre impulso ($\%MP$), error de estado estable (e_{ss}), la variación total del esfuerzo de control (TV_u) y el costo de recursos que se generar al crear el *IP-Core (LUTs, Brams, DSP y Registros, Slack Path Time, Delay Path Time y el Routing Delay Time*); se incluyen las interfaces de comunicación *AXI* para poderse combinar con otros *IP-Cores*. Para la pre-verificación del *IP-Core*, parte fundamental del ciclo de diseño de *IP-Cores*, se utilizará una simulación tanto en *Software* como en *Hardware* utilizando la metodología *HIL*, para esto se utilizará plataformas de *Hardware* reconfigurable (Plataformas de desarrollo basadas en *FPGAS*) utilizadas en trabajos de investigación.

Alcance

El resultado esperado consiste en entregar un modelo de *IP-Core* que cumpla con la función de ser un controlador acelerado por *Hardware*; afrontando los nuevos desafíos de la ingeniería de control en el campo de la industria 4.0. Se busca que las estrategias de control como el *PID* de un grado y dos grados de libertad puedan formar un *IP-Core*, que tenga la posibilidad de combinarse con otros aceleradores de *Hardware* a través de un sistema de comunicación estándar. La combinación de los *IP-Cores* facilita la generación de *SoCs,* que a su vez puedan formar *CPS*, componentes habilitadores en la industria 4.0. Mediante, el uso de estrategias de control industriales, robustas como el controlador *PID* de uno o dos grados de libertad, se puede gobernar plantas complejas como los sistemas servo-controlados. Por otro lado, las estrategias de control estarán

aceleradas por *Hardware* con uso de la tecnología de *FPGAS*, lo que permite mantener un elevado desempeño en el funcionamiento del sistema; como resultado se espera obtener un *IP-Core* capaz de mantener el control con una dinámica aceptable, en un sistema de lazo cerrado, donde se debe mantener un adecuado seguimiento de referencia, robustez ante perturbaciones y cambios de consigna, con un porcentaje de sobre impulso menor al 5%, y un error de estado estable menor al 1%. El prototipo final mantendrá un uso aceptable de recursos de *Hardware* (*LUTs, BRAMs, DSP, I/O*), además, el *IP-Core* debe ser factible de ser implementado; debido a que el tejido de *Hardware* diseñado no debe ocupar más del 25% de recursos disponibles en un *SoC*, con el propósito de reservar recursos que serán ocupados por otros módulos, como procesadores, memorias, buses, y drivers de periféricos.

Descripción del proyecto de investigación.

La presente investigación propone una metodología de diseño *TOP-DOWN* (Lorandel et al., 2015; Macko et al., 2018; Sengupta & Roy, 2017), que es la metodología más utilizada para desarrollar sistemas basados en *FPGAS*. El proceso de diseño Top-Down contempla múltiples niveles de abstracción, el nivel más alto, llamado también nivel de sistema y modelamiento, usualmente utiliza lenguajes como *C/C++, Matlab o SystemC*; para este caso se utilizará las herramientas de Matlab para modelar, sintonizar y simular el sistema de control.

La simulación en *Software* permite analizar el comportamiento de los controladores sintonizados e implementados de manera tradicional, a través de un proceso secuencial realizado por el procesador, y servirá de base para realizar un análisis comparativo entre los controladores implementados por *Software* y *Hardware*. Las métricas para evaluar el desempeño del controlador son: el tiempo de establecimiento (t_{ss}), el porcentaje de sobre impulso ($\%M$), error de estado estable (e_{ss}), la variación total

del esfuerzo de control (TV_u), la robustez ante cambio de consigna, y ante perturbaciones internas mediante la función sensibilidad (M_s) (Meng et al., 2020; Verma & Padhy, 2020).

Una vez obtenida las estructuras de los controladores, modelado y simulado con herramientas de *Software,* se puede proceder a modelar el *Hardware* del controlador con la combinación de las herramientas del *Software Xilinx System Generator (SXG)* y de *Matlab/Simulink*. Es importante realizar una verificación del funcionamiento del *Hardware,* para esto se utilizará la metodología de *Hardware HIL*; la cual permite realizar simulaciones en tiempo real, con altas tasas de muestreo y ejecución. Esta estrategia permite simular sistemas formados por componentes virtuales (modelos matemáticos de plantas) y componentes físicos, los cuales corresponden a los controladores sintetizados en *Hardware*. El equipo físico que se utilizará para sintetizar el *Hardware* es la tarjeta Basys-3, la cual es una plataforma de *Hardware* reconfigurable (*FPGA*), que permitirá sintetizar el *Hardware* y realizar la simulación *HIL* junto a la herramienta de *Simulink.*

Para realizar la simulación *HIL*, es necesario realizar la síntesis del controlador en *Hardware*, por tal motivo es inevitable descender a los niveles más bajos de abstracción de la metodología *TOP-DOWN*; que involucra procesos de síntesis, ruteo y generación del archivo *Bitstream*; este proceso se lo realizará mediante las herramientas de *Matlab y Vivado IDE,* para realizar las pruebas de simulación correspondientes. Las métricas para evaluar el desempeño del *IP-Core* estarán divididas en dos partes; la primera parte, evalúa el desempeño del controlador en *Hardware,* a través de las métricas mencionadas para el proceso de simulación con recursos puramente de *Software*, es decir con la evaluación de $t_{ss}, \%M, e_{ss}, TV_u$, robustez ante cambio de consigna y perturbaciones internas del sistema. En el análisis se observará la variación y error que existe en la salida del controlador sintetizado en *Hardware* y la respuesta del sistema con precisión de punto fijo y punto flotante, versus la respuesta obtenida de la implementación del controlador de manera tradicional, ejecutada por un procesador utilizando precisión de punto flotante. El

uso de distintos sistemas de precisión en el diseño genera un impacto directo en el uso de recursos de *Hardware* y en el análisis de tiempos de síntesis. Aquí comienza la segunda parte de la valoración del *IP-Core*, donde se evalúa el uso de recursos de *Hardware* en la síntesis, los parámetros medibles son: el porcentaje de *LUTs, BRams, DSP* y *Registros* necesarios para generar el *IP-Core*; mientras que para el análisis de tiempos de síntesis se evaluán métricas de *Slack Path Time* (SPT), *Delay Path Time* (DPT) y el *Routing Delay Time* (RDT).

Se pretende realizar un estudio que permite combinar la ingeniería electrónica y de control para formar *CPS* usados en la industria 4.0, con el cuál se pueda obtener un sistema de control robusto, como la estrategia del controlador *PI/PID* de uno y dos grados de libertad disponible como un recurso de *Hardware* listo para ser utilizados en sistemas como brazos robóticos u otros procesos. Se debe recordar que estos tipos de controladores poseen un papel importante en la actualidad, ya que permiten gobernar sistemas con filosofías de control complejas; a su vez pretende, que las estrategias de control clásicas y nuevas, estén presentes en los nuevos avances tecnológicos relacionados a la electrónica y al *IoT*.

Objetivos

Objetivos General

Proponer una metodología de diseño basado en modelos de controladores clásicos para la generación de *IP-Cores* enfocados en la Industria 4.0.

Objetivos Específicos

- Definir la arquitectura y modelo de controlador en el dominio de tiempo discreto factible de ser modelado e implementado en *Hardware,* y que sea de utilidad para gobernar sistemas ciber-físicos de la industria 4.0.

- Diseñar el *IP-Cores* a través del ciclo de diseño del *IP-Core* basado en el modelamiento de *Hardware* del controlador seleccionado.
- Determinar el desempeño del *IP-Core* modelado y generado, para satisfacer una función requerida en un entorno altamente automatizado de la industria 4.0
- Identificar los parámetros de síntesis que posibilitan la implementación e integración del IP-Core dentro de un sistema embebido, para la formación de un *SoC* básico orientado a la industria 4.0.

Organización

El presente documento consta de seis capítulos agrupados en tres partes, además de un capítulo introductorio, el contenido del documento está dividido de la siguiente forma:

- **Estado del Arte:** El capítulo 2 brinda una visión general de la Industria 4.0 y los *IP-Cores*, se describe la evolución y situación actual de la manufactura, y se detallan los requisitos de operación de los dispositivos que componen la Industria 4.0. Además, se presenta una introducción a los núcleos de propiedad intelectual, se describe algunas de las ventajas de utilizar estos tejidos de *Hardware* preverificados, y cómo facilitan el diseño de los más recientes y avanzados *SoC*; posteriormente se aborda el ciclo de diseño de un *IP-Core,* y se resume la manera de cómo se integran varios *IP-Cores* en un solo encapsulado. Finalmente se proporciona un resumen de trabajos previos que abordan arquitecturas de controladores *PID* implementados en *Hardware*.
- **Contribución:** Basado en la metodología de modelos y sistemas de control en plataformas embebidas, se desarrolla el capítulo 3, que propone una estrategia de diseño fundamentada en modelos para la generación de *IP-Cores* que puedan formar parte de la Industria 4.0, que permitan satisfacer las necesidades relativas

introducidas en el capítulo 2, como son: la modularidad, la flexibilidad, la interoperabilidad, el modelado de sistemas que cubren aplicaciones complejas, entre otras. Por otro lado, el capítulo presenta una propuesta de aplicación para validar la metodología basada en modelos, así como la aplicabilidad de la arquitectura en la industria. El capítulo 4 presenta el diseño y las estrategias utilizadas para generar el *Hardware* de los *IP-Core*, sobre la base de la metodología y los modelos presentados en el capítulo 3.

- **Validación:** El capítulo 5 expone la viabilidad de implementar el sistema propuesto como un *IP-Core*, en la primera parte del capítulo, se describe el *Hardware* y *Software* utilizado para realizar la experimentación; el resto del capítulo se centra en la descripción y verificación del funcionamiento de las arquitecturas sintetizadas en *Hardware*. En este sentido se evaluó dos escenarios con la finalidad de validar el desempeño y uso de recursos necesarios para ser implementados. El primer escenario contempla una implementación cuyas operaciones en el procesamiento digital de señales (*DSP*) utiliza una aproximación de punto flotante, el segundo escenario presenta una implementación de las operaciones *DSP* con aproximación de punto fijo; para cada escenario se evaluá al sistema embebido implementado ante perturbaciones, y se valoraron algunas métricas como el error que existe entre el comportamiento simulado en *Software* y el implementado en *Hardware*.

La Industria 4.0 y Los *IP-Cores*

Introducción

El Término de industria 4.0 apareció en el año 2011 y desde entonces ha sido el centro de atención en la manufactura industrial en varias partes del mundo. La industria 4.0 en inicio se conceptualizó como la cuarta revolución industrial, pero durante los últimos años esta conceptualización ha evolucionado involucrando una transformación digital y fabricación inteligente (Ghobakhloo, 2020), de modo que, componentes como los sistemas de control, maquinaria, materiales, productos, sistemas de decisión y recursos humanos puedan interactuar entre sí de manera coordinada y en tiempo real. La transformación digital bajo la Industria 4.0 se basa en la implementación e integración de una variedad de tecnologías basadas en *Cyber Physical Systems*. Estos sistemas están formados por un conjunto de dispositivos digitales, de operación simples y avanzadas, como sensores industriales, controladores industriales, vehículos guiados automatizados (*Automatic Guided Vehicle, AGV*), robots, realidad aumentada y realidad virtual (*Augmented and Virtual Reality*, *AVR*). Así como análisis de datos, computación en la nube e internet de los servicios (*Internet of services*, IoS), los que cuentan con ayuda tecnologíca de diseño y fabricación de computación de alto rendimiento (*High Performance computing - Computer-Aided-Design and Manufacturing*, *HPC-CADM*) y la Inteligencia Artificial (*Artificial Intelligence*, *AI*) (Chen et al., 2017; Hofmann & Rüsch, 2017). Sin duda los *CPS* son el elemento más representativo de la cuarta revolución industrial, pero adicionalmente se compone de tecnologías *IoT*, el *Big Data*, la nube y algoritmos de inteligencia artificial (Onik et al., 2019).

En este capítulo se describen brevemente aspectos relativos a la Industria 4.0: su definición, evolución, características, junto a los requisitos de operación de los dispositivos y tecnologías que se integran en la cuarta revolución industrial; se realiza la

descripción de los *IP-Cores*, técnicas de diseño de esta tecnología y la arquitectura de los buses de alto rendimiento utilizados en los *IP-Cores*. Finalmente, la última parte del capítulo describe arquitecturas de controladores implementados como *IP-Core* o simplemente como un diseño *Stand-Alone*, donde se podría identificar que al no tener un estándar de comunicación tiene el inconveniente de no poder integrarse con otros *IP-Core*.

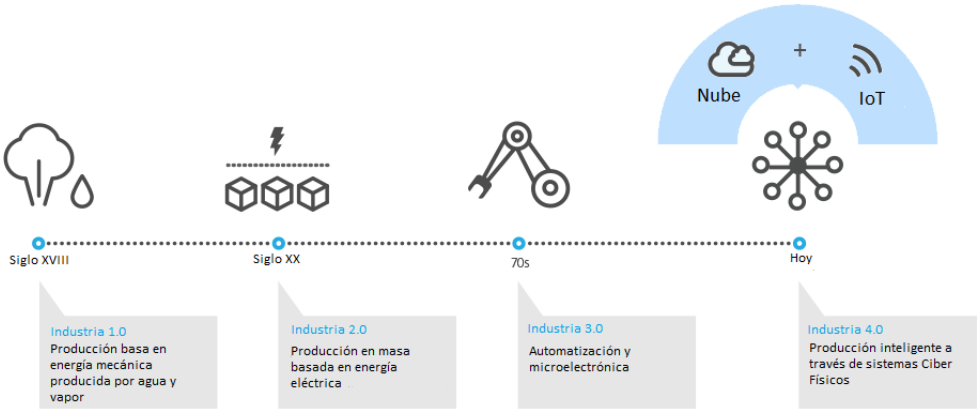
Evolución de la Industria 1.0 a la Industria 4.0

La evolución del proceso de la manufactura industrial, ha pasado por cuatro etapas o revoluciones, véase la Figura 1. La Primera Revolución Industrial inició a finales del siglo XVIII y principios del XIX, la energía de los sistemas de fabricación mecánicos se basaba en agua y vapor. La segunda Revolución Industrial comenzó a finales del siglo XIX, y se caracteriza por la producción en masa mediante el uso de energía eléctrica; la Tercera Revolución Industrial inició a mediados del siglo XX e introdujo la automatización y la tecnología microelectrónica en la fabricación. Estos avances en la manufactura estaban estrechamente relacionados con las TICs; por otra parte aparecen tecnologías para el diseño asistido por computadora (*Computer-Aided-Design, CAD*), la fabricación asistida por computadora (*Computer-Aided-Manufacturing*, *CAM*) y la planificación del procesamiento asistido por computadora (*Computer Aided Process Planning, CAPP*), que hicieron posible la fabricación integrada por computador (*Computer-Integrated-Manufacturing , CIM*) (Xu et al., 2018b). Para la cuarta revolución industrial aparece el término de Industria 4.0, que fue una frase acuñada por el gobierno alemán en la feria de *Hannover* en 2011 que promovió el uso de la digitalización en la manufactura (Griffiths & Ooi, 2018). En la industria 4.0 la fabricación funciona de manera integrada con el uso de los *CPS* para formar un entorno automatizado inteligente. Los *CPS* son los encargados de monitorear los procesos físicos, en palabras simples son los responsables de crear

una copia virtual del mundo físico, para tomar decisiones descentralizadas e integrando tecnologías *IoT*, como resultado se obtiene un conjunto de *CPS* capaces de poderse comunicar y cooperar, entre sí y con los humanos en tiempo real (Carvalho et al., 2018).

Figura 1

Línea de tiempo de las revoluciones industriales.



Requisitos de Operación de los Dispositivos de la Industria 4.0

La Industria 4.0 se enfrenta a un descubrimiento de nuevas tecnologías, para las modernas líneas de montaje, altamente automatizadas con el fin de responder y adaptarse a los requisitos y demandas dinámicas del mercado actual. Esto conlleva retos de integración, previsibilidad, flexibilidad y robustez que permiten afrontar condiciones inesperadas en los procesos de fabricación (Vaidya et al., 2018). Los dispositivos o elementos que componen la Industria 4.0 deben cumplir con una serie de requisitos que satisfagan los siguientes puntos:

- Los dispositivos físicos deben ser flexibles, modulares e inteligentes, capaces de agruparse y trabajar juntos para la toma de decisiones distribuida (Vaidya et al., 2018).

- El modelado de un sistema debe concluir en un modelo de control adecuado, esto provoca que aun continúen habiendo investigaciones para cubrir aplicaciones complejas (Vaidya et al., 2018).
- Se debe garantizar la alta calidad e integridad de los datos que intervendrán en la manufactura de la industria 4.0 (Vaidya et al., 2018).
- Los dispositivos deben operar protocolos de red inalámbricos industriales (*Industrial Wireless Network, IWN)* de alta velocidad, para la transferencia de un alto volumen de datos (Vaidya et al., 2018).
- Los dispositivos ciber-físicos deben contar con una estructura de 3 capas; la primera, la capa red de conexión, es la encargada de la conectividad, seguridad y gestión de los datos. La segunda, la capa de red cibernética, responsable de la conversión de datos del mundo físico y del mundo virtual. Finalmente tenemos la capa de red de gestión, que es el nivel de apoyo a la toma de decisiones basados en datos. (Lee et al., 2015).
- Deben ser capaces de producir datos masivos, transferibles en redes de banda ancha, de modo que la nube pueda procesar los macrodatos (Wang et al., 2016).
- Deben integrar un gestor de comunicaciones, soportar transferencias de datos entre dispositivos IoT, e industriales en tiempo real (Lins & Oliveira, 2020).

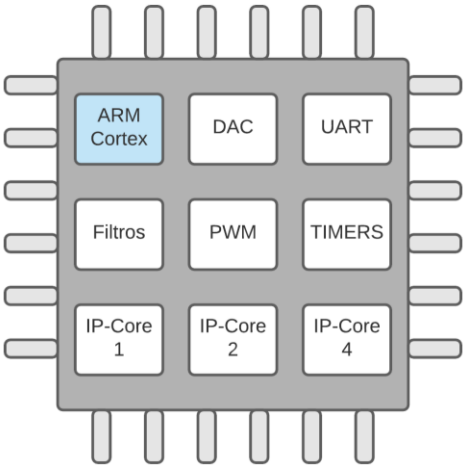
IP-Cores

El acelerado incremento de la complejidad del diseño y fabricación de circuitos integrados ha provocado que, los diseñadores busquen una forma de desarrollar y condensar sistemas completos en un solo encapsulado, conocidos como *System on Chip*; los cuales integran una serie de bloques funcionales en un espacio físico reducido, ver Figura 2. Los principales bloques que componen un *SoC,* son *IP-Cores*, esta tecnología es parte de la solución fundamentada en una metodología basada en reutilización de

diseños. Los *IP-Cores* facilitan al desarrollador de sistemas embebidos, m*AXI*mizan la productividad del diseño y minimizan el tiempo de desarrollo del circuito integrado, mientrás reducen el tiempo de comercialización del mismo (He et al., 2019; Sengupta & Kachave, 2018). Por lo tanto, los diseñadores de *SoCs* modernos pueden hoy en día utilizar e integrar rá*PID*amente los *IP-Cores* para controlar periféricos *UART, USB* o *SPI,* de forma totalmente segura, ya que los *IP-Cores* están pre-verificados y validados (Chakravarthi, 2020). Como consecuencia desde el 2012 al menos el 90% de los circuitos integrados son diseñados e implementados a partir de diferentes tipos de *IP-Cores* como por ejemplo los *Hard-Cores, Soft-Cores* y *Firm-Cores* (Palma et al., 2002).

Figura 2

Representación de un System On Chip compuesto por IP-Cores

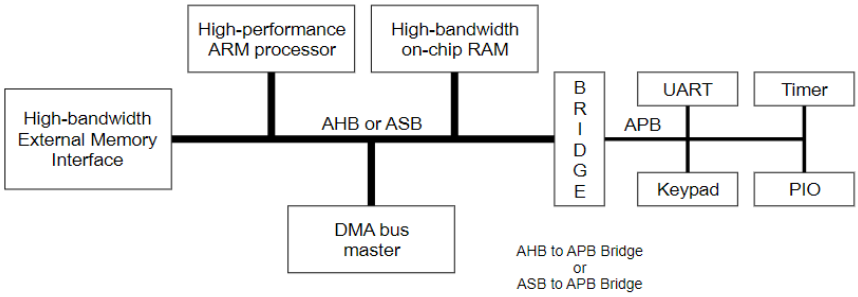


La integración y coordinación de diferentes núcleos *IP dentro* de un mismo encapsulado puede servir para una amplia gama de aplicaciones, como los sistemas de automoción, los dispositivos portátiles y los dispositivos *IoT*. Para que los diferentes bloques funcionales o *IP-Cores* interactúen entre sí dentro de un *SoC*, se suelen utilizar protocolos (bus) para la transferencia de datos y el control. Es muy común que los *IP-Cores* utilicen el estándar abierto *AMBA* (ARM, n.d.-b), que proporciona una interfaz de alto rendimiento para la reutilización de *nucleos de propiedad intelectual*, este ayuda a

reducir los riesgos de fallo, así como los costes de desarrollo en los diseños con muchos *IP-Cores* y periféricos. Otras formas en que se pueden comunicar los *IP-Cores* es a través de estructuras de *Networks On Chips* (*NOCs*) (Kornaros, 2018), sin embargo aún poseen ciertos errores y vulnerabilidades que suelen ser temas de investigación para ser corregidos (Prasad et al., n.d.). Por tal motivo es muy común observar una gran cantidad de diseños *SoCs* que utilizan una arquitectura de bus *AMBA,* desarrollada por *ARM* como se muestra en la Figura 3, el protocolo utilizado en esta arquitectura es *AMBA-AXI* (ARM, n.d.-a) soporta la comunicación con dispositivos de alto rendimiento.

Figura 3

Arquitectura típica de bus AMBA. ARM. A typical AMBA system



Nota. EL gráfico representa la arquitectura de bus *AMBA*. Recuperado de: https://developer.arm.com/documentation/ihi0011/a/Introduction-to-the-*AMBA*-Buses/A-typical-*AMBA*- based-microcontroller?lang=en

La arquitectura hace uso principalmente de dos tipos de buses:

- ***El bus AHB (Advanced High-performance Bus)****:* Es un bus de alto rendimiento gobernado por un reloj de alta frecuencia, capaz de soportar transferencias tipo ráfagas.
- ***El bus APB (Advanced Peripheral Bus):*** Es un bus utilizado para la interacción de diferentes periféricos, con un ancho de banda bajo y que no

requiera un alto rendimiento, optimizado para tener un consumo mínimo de energía y con una reducida complejidad en su interfaz.

- ***BRIDGE:*** Se encarga de la conversión de las tramas de datos entre los buses *AHB* y *APB*, funciona como un moderador para que la información pueda fluir entre estos dos buses.

Normalmente se considera a los *IP-Cores* como sistemas de *caja negra,* es decir se conoce la función que realiza el componente pero no es necesario saber el diseño interno del *IP-Core*, de modo que se pueden considerar parte de los elementos *Commercial-Off-The-Shelf (COTS)* (Di Mascio et al., 2021). Esto significa que están configurados y listos para ser utilizados por un tercero que no tuvo participación en la generación del *IP-Core*, de modo que el tiempo en la etapa de diseño se reduzca para futuros proyectos. Dentro de los *IP-Cores* más utilizados se puede encontrar a los procesadores de tipo *Sof-Core* como: *Microblaze, Leon* y *NIOS*.

Técnica de Diseño y Modelado para los *IP-Cores*

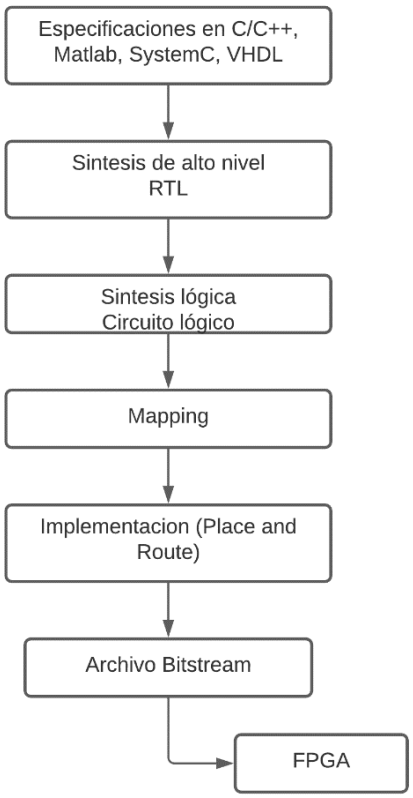
Normalmente, el ciclo de desarrollo de un *IP-Core* utilizado las técnicas de diseño se compone de cuatro pasos fundamentales: Modelado, verificación, optimización y protección del núcleo de propiedad intelectual (Mohamed, 2016). Los diseñadores, pueden utilizar lenguajes como *C/C++, Matlab, SystemC* o *VHDL* para generar y modelar los aceleradores de *Hardware (IP-Core)* (Bazeghi, 2016) (Deschamps et al., 2019).

El flujo de diseño para un circuito inicia con las especificaciones descritas por un lenguaje de programación (*C/C++, Matlab, SystemC*) o un lenguaje de descripción de *Hardware* (*VHDL*), hay que destacar que el lenguaje Matlab se puede utilizar directamente para evaluar el rendimiento y verificación funcional de los diseños HDL (Jain et al., 2016). Una vez definido el comportamiento del sistema mediante lenguajes de alto nivel, se lleva a cabo un proceso de conversión de estas especificaciones, como resultado

de esta conversión, se obtiene un archivo que define el funcionamiento del circuito a nivel de transferencia de registros (*Register-Transfer Level, RTL*). Posteriormente, se realiza una síntesis lógica del archivo *RTL*, dando como resultado un circuito lógico denominado *Netlist*, compuesto por puertas lógicas, registros y bloques de memoria y de redes de interconexión. A partir de ahí, el circuito lógico se mapea en un circuito equivalente, compuesto por elementos de una librería formada por *Standard Cells, Gate Arrays, LUTs, MUX y CLBs*; en la fase final de diseño, se implementa el circuito (*Place and Route*); finalmente, se genera el archivo llamado *Bitstream* que es descargado a la *FPGA* (Deschamps et al., 2019). La Figura 4 muestra el diagrama de flujo que resume el proceso de diseño y generación del *Hardware* de un circuito.

Figura 4

Flujo de diseño para la generación de circuitos basados en FPGAS



Nota. Adaptado de Complex Digital Circuits (p. 146), por J. Deschamps, 2019, Springer

Arquitecturas de Controladores PI/*PID* de un Grado de Libertad Implementados en *Hardware*

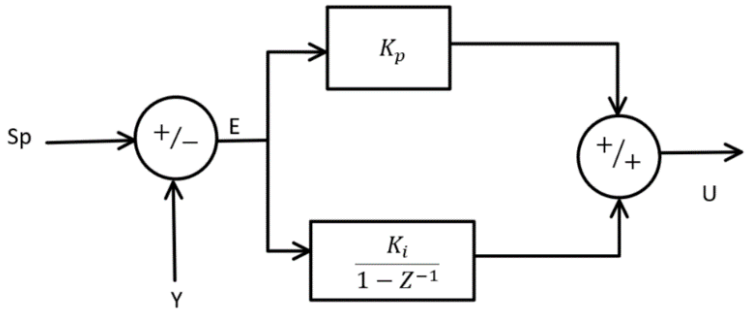
En el estudio del estado del arte se encontraron contribuciones de *IP-Cores* relacionadas a la industria, de los cuales se hallaron controladores *PI/PID* de un grado de libertad. Los controladores *PID* son controladores robustos, sin embargo, existen modelos más sofisticados que este controlador, como el controlador *PID* de dos grados de libertad. El cuál es utilizado para mantener un buen comportamiento transitorio ante cambios de consigna en sistemas servo-controlados; presentes en robot articulados que forman parte de la automatización de la industria 4.0. En la revisión de arquitecturas de trabajos previos se pueden hallar controladores *PI/PID* de un grado de libertad, que fueron utilizadas para *IP-Cores* (Huang et al., 2017) o que fueron implementadas en *FPGA* como un diseño *Stand-Alone* (Abdelati, 2016; Indira & Swapna, 2017; Youness et al., 2014). En (Huang et al., 2017) utilizan una arquitectura estándar PI de la forma *Euler en atraso* para el integrador digital, la arquitectura está descrita en la ecuación (1).

$$U = \left(K_p + \frac{K_i}{1 - Z^{-1}}\right) E \tag{1}$$

Donde K_p es la ganancia proporcional y K_i es la ganancia del integrador del controlador *PI*, el diagrama de bloques del controlador se presenta en la Figura 5. Este trabajo se destaca por el uso del diseño del controlador *PID* para la construcción del *IP-Cores*, de este modo, fue posible integrar varios *IP-Cores* para gobernar las articulaciones de un brazo robótico; cuyo perfil de posición está controlado por un procesador *ARM*, como resultado se obtiene un *SoC* útil para la industria.

Figura 5

Diagrama de bloques del controlador PI digital con Euler en atrasado



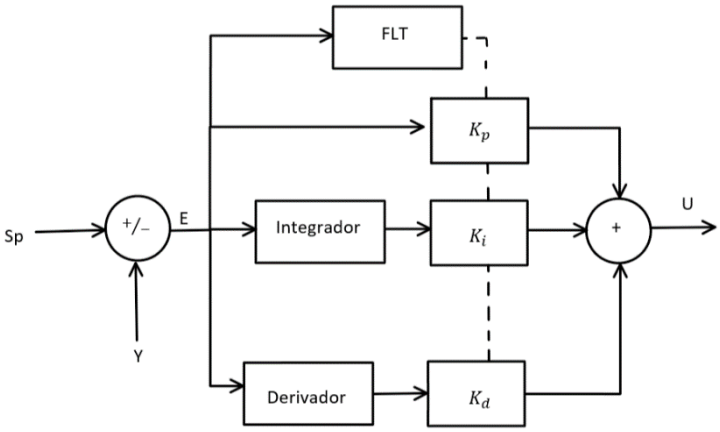
Nota. Adaptado de Noble Implementation of Motor Driver with All Programmable *SoC* for Humanoid Robot or Industrial Device (p. 17), por L. Huang, 2017, International Journal of Humanoid Robotics.

(Youness et al., 2014) propone un controlador embebido *PID* de un grado de libertad digital como se muestra en el diagrama de bloques de la Figura 6, cuyos parámetros de controlador son sintonizados por el bloque *FLT* que contiene Lógica Difusa (*Fuzzy Logic, FL*). La implementación se realizó en una plataforma de *Hardware* reconfigurable y mediante la simulación *HIL*, se validaron dos propuestas. La primera consiste en implementar el controlador a través de sistemas de multiprocesamiento basados en *FPGAS* (*FPGA-based Multiprocessing Systems),* en donde la implementación resulta ser eficiente y los recursos de *Hardware* se adaptan a las necesidades de control, sin embargo, el esfuerzo de diseño puede ser mayor debido a que hay que considerar el diseño del sistema de comunicación entre los *IP-Cores*. La segunda propuesta se basa en sistemas multiprocesadores basados en MCU *(Multicore Microcontrollers, MCUs),* también llamados ordenadores embebidos en tiempo real, que pueden realizar múltiples tareas en paralelo (*Parallel-Multi-tasking, PMT*), el esfuerzo de diseño es menor y se centra en el *Software*, sin embargo, el *Hardware* es permanente y

no flexible. En general, la implementación basada en multiprocesadores aumenta la velocidad en tiempo real, en casi tres veces más que las implementaciones tradicionales del controlador *PID* ejecutadas secuencialmente con un solo procesador.

Figura 6

Diagrama de bloques del controlador digital PID



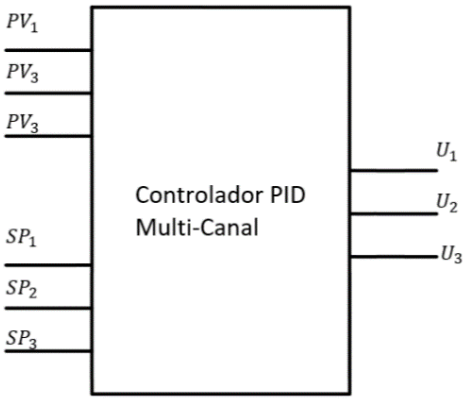
Nota. Adaptado de MP*SoCs* and multicore microcontrollers for embedded *PID* control: A detailed study (p. 9), por H. Youness, 2014, IEEE Transactions on Industrial Informatics.

Otra propuesta interesante implementada en *FPGA* es la arquitectura de un controlador *PID* multicanal de un grado de libertad propuesta por (Indira & Swapna, 2017), donde utilizaron para controlar un sistema de velocidad de un ventilador. La Figura 7 muestra la arquitectura del controlador multicanal propuesto, donde PV es la variable de proceso, SP es la consigna y U la acción de control. La implementación en el *Hardware* de la *FPGA* permite el control de múltiples bucles de manera concurrente; esta implementación se caracteriza por obtener una estrategia de control de procesamiento de alta velocidad y resolución, configurable en términos de potencia y paralelismo. Si bien se destacan las ventajas de contar con una estrategia de control implementada por

Hardware, esta propuesta carece de cualquier mecanismo de comunicación con otros *IP-Cores* para formar parte de un diseño de *SoC* moderno.

Figura 7

Arquitectura de controlador PID Multi-Canal



Nota. Adaptado de *FPGA* Implementation of low Power Multi Channel Generalized *PID* Controller for Industrial Automation Applications (p. 2), por M. Indira, 2017, International Journal of Engineering Trends and Applications.

En la literatura se puede encontrar muchos trabajos vinculados a la arquitectura de controladores implementados en *Hardware*, sin embargo, en muchos de los casos se tratan de diseños *Stand-Alone*; por esta razón se ha destacado en mayor parte los trabajos que implementan sus diseños utilizando algún tipo de tecnología relacionada a los *IP-Cores* en la industria. Es evidente que el uso de un *Hardware* reconfigurable permite obtener implementaciones diferenciadas con bajas latencias, alta disponibilidad y paralelismo entre otros. Así, el objetivo es presentar una metodología de diseño que permita mantener las ventajas de los modelos implementados en *Hardware*, pero con la capacidad de disponer de un mecanismo de integración con varios *IP-Cores,* que permitan formar un sistema *SoC* capaz de integrarse a la Industria 4.0.

Resumen

En este capítulo se ha presentado una visión general sobre la Industria 4.0, los *CPS* y los *IP-Cores.* Durante la revisión bibliográfica se ha encontrado diversas definiciones, de donde se deduce que los *CPS* son la pieza clave que componen la Industria 4.0. Los *CPS* se componen de una parte física (sensores y actuadores) y una parte cibernética, que se encarga de trasladar el mundo real a un mundo virtual, con capacidad para realizar análisis de datos masivos, gestión de comunicaciones, etc. La interacción de diferentes sistemas *CPS*, que de la mano con las tecnologías como *BigData*, *IoT e* Inteligencia Artificial constituyen la base de un entorno de fabricación inteligente.

Por otro lado, se han identificado algunas características que deben tener los dispositivos que conforman la industria 4.0 entre las que destacan: la flexibilidad, modularidad e inteligencia de los sistemas para cooperar en conjunto. Es importante que los dispositivos puedan garantizar una alta calidad e integridad en el tratamiento de los datos relacionados con la fabricación inteligente, del mismo modo, los dispositivos deben ser capaces de producir datos masivos, que puedan transferir gran cantidad de información a través de redes de banda ancha, para que el procesamiento de B*ig Data* pueda realizarse en la nube; por último, se debe integrar un gestor de comunicaciones industriales, pero que al mismo tiempo soporte la transferencia de información con los dispositivos *IoT*.

Se ha señalado cómo los *IP-Cores* son la pieza fundamental que facilita la creación de los *SoCs* modernos, se han mencionado algunos estándares de comunicación que utilizan los *IP-Cores* para interactuar entre sí, de los cuales cabe resaltar los buses de alto rendimiento como el *AMBA* y los *NoC*. La investigación de los *IP-Cores* nos ha conducido a explicar la técnica de diseño de los núcleos de propiedad

intelectual, que se encuentra basada en un ciclo de cuatro pasos fundamentales: el modelado, la verificación, la optimización y la protección intelectual del *IP-Core*; para el ámbito del diseño del *Hardware* de los *IP-Cores*, se ha mencionado un diagrama de flujo utilizado para sintetizar circuitos basado en una metodología *TOP-DOWN*, la cual parte de un alto nivel de abstracción para realizar el diseño mediante las especificaciones, hasta un bajo nivel de abstracción donde se obtiene un sistema compuesto por compuertas lógicas, registros y bloques de memoria.

Finalmente, se han revisado algunos trabajos de diseño de controladores *PID* utilizando tecnología relacionada con *IP-Cores* o en *FPGA*. El uso de controladores acelerados por *Hardware* permite tener una velocidad de procesamiento casi tres veces superior a la de los sistemas tradicionales implementados en monoprocesadores, esta diferencia se nota aún más cuando es necesario implementar varias veces la misma estrategia de control en un mismo dispositivo electrónico; ya que los algoritmos de control por *Hardware* se pueden ejecutar de forma concurrente, mientras que en un procesador tradicional se ejecutan en secuencia. En este ámbito es fundamental desarrollar una estrategia que permita a los diseñadores generar *IP-Cores* que puedan cumplir con los criterios de monitoreo, control y comunicación. Para este estudio, el sistema de control que se propone modelar en *IP-Cores* pretende ser lo suficientemente flexible para dar solución a algunos requerimientos de la Industria 4.0., donde se pretende incluir en el estudio, el uso de controladores *PID* de dos grados de libertad como *IP-Cores*. Según la revisión literaria, esta estrategia de control presenta un mejor desempeño en el dominio de los servosistemas, de este modo se podría utilizarlos para generar *CPS,* que gobiernen en paralelo las articulaciones de brazos robóticos presentes en las fábricas inteligentes.

Propuesta de la Metodología de Diseño Basado en Modelos

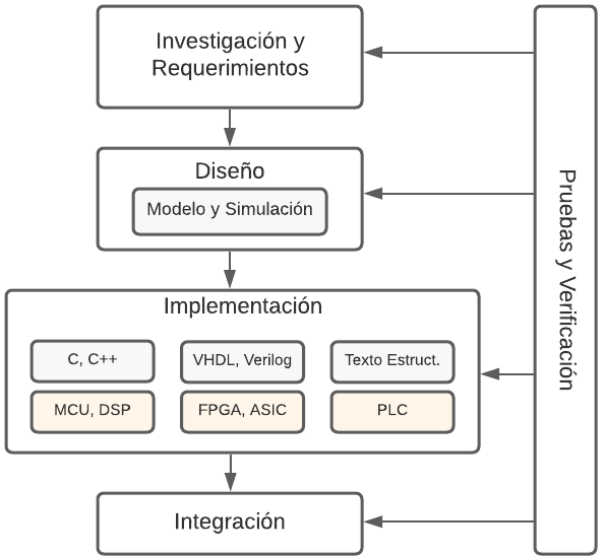
Introducción

Hace un tiempo atrás en un flujo de diseño tradicional, frecuentemente no se podía probar y validar los diseños de los sistemas de control hasta una etapa muy avanzada del periodo de desarrollo; es decir cuando los sensores, actuadores y otros elementos de *Hardware* del sistema estaban disponibles, es así que algunos problemas aparecían en el momento de la implementación, lo que conllevaba a la corrección de dichos problemas. Debido a este inconveniente surge un nuevo enfoque para evaluar rápidamente las estrategias de control y detección temprana de errores, de modo que el enfoque del diseño basado en modelos toma importancia, a través del cual se tiene un seguimiento del diseño siendo protagonista en todo el proceso del desarrollo; se inicia con la recolección de los requisitos del sistema, posteriormente se realiza el diseño del modelo de simulación y creación del modelo detallado del sistema, para finalizar con la implementación, prueba y verificación.

Una cualidad importante de esta metodología consiste en la posibilidad de simular el modelo en cualquier fase del proyecto, de forma que se pueda observar el comportamiento del sistema en diferentes escenarios, sin correr ningún peligro, sin sufrir retrasos importantes y sin tener que recurrir a un costoso *Hardware*. Incluso si el sistema es grande y complejo, los diseñadores pueden probar componentes de manera individual e independiente para ponerlos a prueba en un sistema completamente simulado (MathWorks, 2020). La Figura 8 muestra el flujo de diseño basado en modelos; posteriormente en este capítulo se presentan los beneficios de utilizar esta metodología en la actualidad y se describe la metodología propuesta para generar los *IP-Cores* basada en modelos.

Figura 8

Flujo de trabajo del diseño basado en modelos.



Nota. Adaptado de Model-Based Design for Embedded Control Systems (p. 3), por MathWorks, 2020. Recuperado de: https://www.mathworks.com/content/dam/mathworks/white-paper/gated/model-based-design-with-simulation-white-paper.pdf.

Beneficios del Uso de la Metodología de Diseño Basada en Modelos

El uso de una metodología de diseño basada en modelos ha transformado la forma de trabajar de los ingenieros y científicos, mejorando los resultados obtenidos en proyectos de laboratorio y de campo. Esta metodología gestiona el uso de recursos de *Software*, *Hardware* y tiempo, lo que permite la creación de bancos de pruebas para la verificación de sistemas, de este modo, se optimiza los tiempos de desarrollo y se evita la introducción de errores de forma manual (MatWorks, 2020). Como resultado, se obtiene un diseño favorable para ser llevado al campo, aunque se pueden utilizar diferentes dispositivos para la implementación, este estudio se centra en el uso de *FPGAS* para sintetizar el *Hardware* requerido.

Metodología Propuesta para el Modelado de *IP-Cores B*asado en Modelos

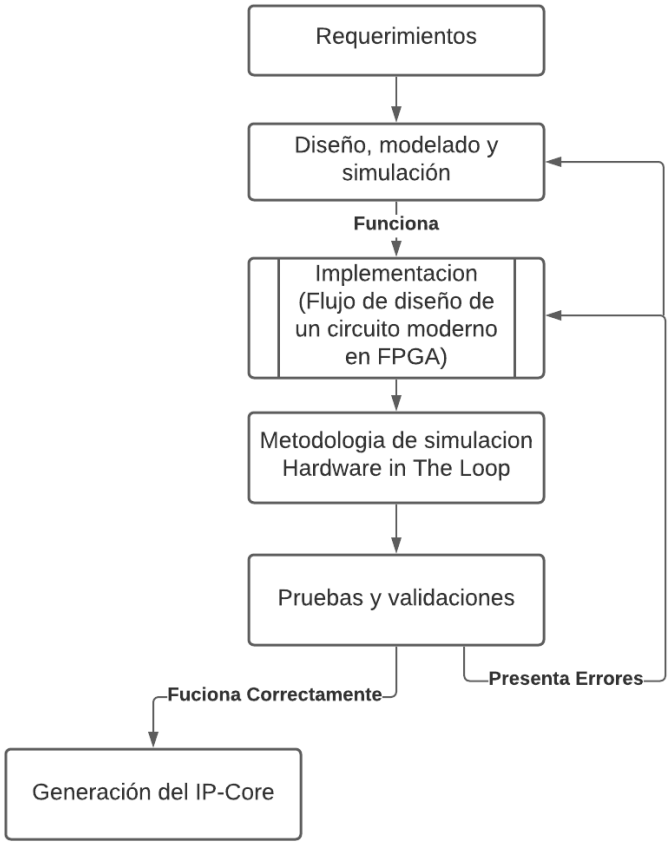
La metodología propuesta para el diseño del *IP-Core* basado en modelos consiste en combinar la estrategia de la Figura 8 y la filosofía *Top-Down* mostrada en la Figura 4; se debe resaltar, que el proceso de implementación estará enfocada al *Hardware* utilizando la *FPGA*. Una aclaración importante que surge, es que, al combinar estas dos metodologías, ya no es estrictamente necesario utilizar directamente *VHDL* o *Verilog* para el diseño de *Hardware*. Actualmente, existen flujos de diseño de circuitos modernos de *Hardware* para *FPGA*, que utilizan una metodología Top-Down donde se pueden integrar otros lenguajes como *C/C++, SystemC* o en *Matlab* para diseñar *Hardware* (Deschamps et al., 2019). Para la sección de pruebas y verificación se ha optado por utilizar la simulación *HIL*. La Figura 9 muestra el diagrama de flujo de la metodología propuesta, la cual permite mantenerse en el ciclo de diseño del *IP-Core* presentado en (Mohamed, 2016); para este estudio, se aplicará la metodología basada en modelos para generar *núcleos de propiedad intelectual* enfocados a la Industria 4.0. Se ha optado por modelar un sistema servo-controlado, basado en una planta de motor de corriente continua, que estará gobernada por un controlador *PID* de un grado de libertad y por un controlador *PID* de dos grados de libertad.

Los escenarios seleccionados pretenden mostrar el uso de recursos necesarios para implementar este tipo de controladores diseñados como *IP-Cores*, además de presentar el desempeño que poseen estos acelerados de *Hardware* cumpliendo su función. Posteriormente el uso de estos *IP-Cores* tienen como finalidad facilitar el diseño de plataformas *SoCs* para controlar sistemas de servo-motores, encargados de manipular las articulaciones de los brazos robóticos utilizados en los procesos de automatización industrial de nueva generación. (Indri et al., 2018; Tang & Veelenturf, 2019). El modelado se basará en principios físicos y la teoría de sistemas de control, con el objeto de obtener

las especificaciones necesarias para crear el modelo de *IP-Cores* utilizando un lenguaje de programación moderno, posteriormente se procederá a simular, sintetizar, implementar y verificar el funcionamiento correcto del *IP-Core*.

Figura 9

Flujo de diseño para la generación de IP-Cores basada en modelos



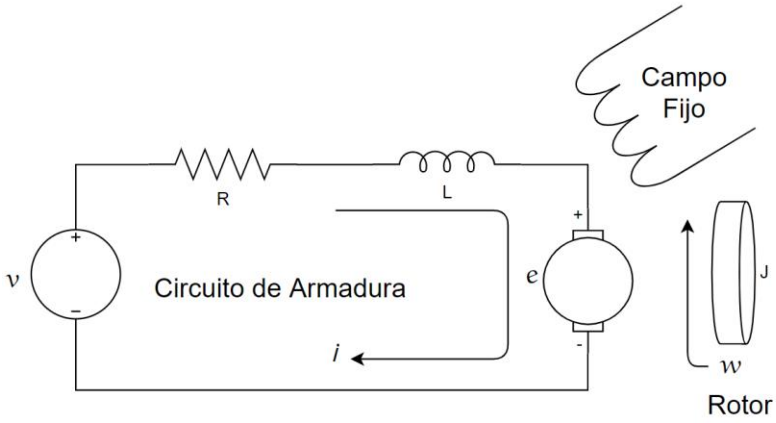
Diseño y Modelamiento del Sistema Servo-Motor DC

Una vez obtenida la motivación y requerimientos para diseñar controladores acelerados por *Hardware* que puedan gobernar sistemas ciberfísicos de la industria 4.0, es necesario pasar a la fase del diseño, modelado y simulación de los sistemas, como se indica en el paso 2 de diagrama de flujo presentado en la Figura 9. Para el diseño del

sistema servo-motor es necesario obtener las ecuaciones diferenciales que rigen el comportamiento de la dinámica del motor de corriente continua. La Figura 10 muestra el circuito equivalente con control de tensión del inducido y el modelo de un sistema mecánico general, el cual incorpora los parámetros mecánicos del motor y su mecanismo acoplado; los efectos de las reacciones del inducido se pueden ignorar y la corriente de campo se establece en un valor constante (Almatheel & Abdelrahman, 2017). El modelo lineal del motor de DC consta de una ecuación diferencial mecánica y una eléctrica, que describen el comportamiento del motor (Suman & Giri, 2016), a través del uso de un espacio de estados se puede especificar la conducta global del sistema.

Figura 10

Circuito eléctrico de armadura equivalente del motor de corriente continua



Nota. Adaptado de Modeling, Simulation and Implementation of Brushed DC Motor Speed Control Using Optical Incremental Encoder Feedback DC Motor Model Simulink Modeling, Simulation and Parameter estimation (p. 498), por B. Joshi, 2014, Proceeding of IOE Graduate Conference.

Utilizando la Figura 10 y la ley de voltajes de Kirchhoff aplicado en el circuito eléctrico se obtiene la ecuación (2).

$$v = R \cdot i + L \cdot \frac{di}{dt} + e \quad (2)$$

Donde: v es el voltaje de la armadura [V], i es la corriente del inducido [A], R es la resistencia del inducido [Ω], L es la inductancia de la armadura [H], e es la Fuerza Electromotriz (FEM) [V].

Por la segunda ley de Newton, para un funcionamiento normal, el par desarrollado debe ser igual al par de carga más la fricción y la inercia.

$$\tau_m = J \frac{d\omega}{dt} + B \cdot \omega + \tau_L \quad (3)$$

Donde: τ_m es el par motor [Nm], J es la inercia del rotor [$kg \cdot m^2$], ω es la velocidad angular [$rad\ /\ s$], B es el coeficiente de fricción viscosa [$Nm \cdot s\ /\ rad$], $\tau_L$es el par de carga [Nm].

En general, el par generado por un motor τ_m de DC es proporcional a la corriente del inducido y la fuerza del campo magnético. Se supone que el campo magnético es constante, por lo tanto, el par del motor es proporcional sólo a la corriente de la armadura i, multiplicada por una constante de par del motor K_t (Motor torque constant), como se muestra en la siguiente ecuación.

$$\tau_m = K_t \cdot i \quad (4)$$

La fem e, es proporcional a la velocidad angular ω del eje, multiplicado por un factor constante K_e (Back emf constan)

$$e = K_e \cdot \omega \quad (5)$$

Empleando la ecuación (4) en (3), y la ecuación (5) en (2) obtenemos las siguientes relaciones:

$$\frac{d\omega}{dt} = \frac{1}{J} \cdot K_t \cdot i - \frac{B_m}{J} - \frac{1}{J} \cdot \tau_L \tag{6}$$

$$\frac{di}{dt} = \frac{1}{L} \cdot v - \frac{R}{L} \cdot i - \frac{1}{L} \cdot K_e \cdot \omega \tag{7}$$

A través del espacio de estados se puede representar la dinámica completa del sistema, donde las variables de estado son $x_1 = \theta$, $x_2 = \omega = \dot{\theta} = \dot{x_1}$ y $x_3 = i$, la variable de control es $u = v$, donde θ es el ángulo desplazado del rotor, además podemos considerar a τ_L como una pequeña perturbación en el sistema, el cual permitirá analizar la robustez del sistema para pequeñas variaciones de este parámetro. Al considerar a τ_L como una perturbación externa del sistema, para el modelo ideal de espacio se puede considerar a $\tau_L = 0$, dando como resultado el modelo descrito en la ecuación (8), la salida del sistema se puede calcular empleando la ecuación (9). Como resultado se obtiene un sistema *SIMO* (*Single Input-Multiple outputs*).

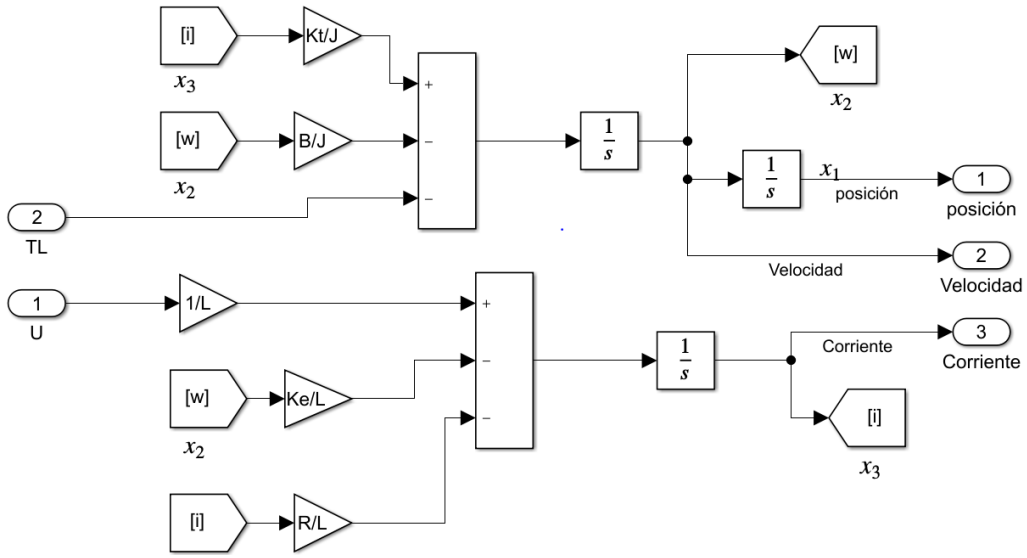
$$\begin{bmatrix} \dot{x_1} \\ \dot{x_2} \\ \dot{x_3} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & -\frac{B_m}{J} & \frac{K_t}{J} \\ 0 & -\frac{K_e}{L} & -\frac{R}{L} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \frac{1}{L} \end{bmatrix} u \tag{8}$$

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \tag{9}$$

La Figura 11 muestra el modelo del controlador construido empleando (6) y (7), donde $\tau_L$se considerará como una perturbación de carga en el sistema.

Figura 11

Modelo del motor DC



Nota. Modelo del motor DC construido en Simulink.

De acuerdo a la teoría de control para la representación de las funciones de transferencia a partir del espacio de estado se puede utilizar la ecuación (10).

$$\boldsymbol{G}(s) = \boldsymbol{C}[s\boldsymbol{I} - \boldsymbol{A}]^{-1}\boldsymbol{B} \tag{10}$$

Donde $\boldsymbol{G}(s)$ es una matriz de q funciones de transferencia, **I** es una matriz identidad. Para el sistema de motor DC propuesto, se tiene tres salidas, por consecuencia se tiene un total de tres funciones de transferencia que relaciona cada salida con la variable de entrada. $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}$ son las matrices de estado dadas por las ecuaciones (11), (12) y (13) respectivamente.

$$A = \begin{bmatrix} 0 & 1 & 0 \\ 0 & -\frac{B_m}{J} & \frac{K_t}{J} \\ 0 & -\frac{K_e}{L} & -\frac{R}{L} \end{bmatrix} \tag{11}$$

$$B = \begin{bmatrix} 0 \\ 0 \\ \frac{1}{L} \end{bmatrix} \tag{12}$$

$$C = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{13}$$

Utilizando la ecuación (10) tenemos como resultado la matriz de funciones de transferencia $\boldsymbol{G(s)}$ descrita en la ecuación (14)**.**

$$\boldsymbol{G(s)} = \begin{bmatrix} \frac{X_1(s)}{U(s)} \\ \frac{X_2(s)}{U(s)} \\ \frac{X_3(s)}{U(s)} \end{bmatrix} = \begin{bmatrix} \frac{\theta(s)}{V(s)} \\ \frac{\omega(s)}{V(s)} \\ \frac{I(s)}{V(s)} \end{bmatrix} = \begin{bmatrix} \frac{K_t}{(B_mR + K_eK_t)s + (B_mL + JR)s^2 + JLs^3} \\ \frac{K_t}{B_mR + K_eK_t + (B_mL + JR)s + JLs^2} \\ \frac{B_m + J \cdot s}{B_mR + K_eK_t + (B_mL + JR)s + JLs^2} \end{bmatrix} \tag{14}$$

Los motores DC están presentes en aplicaciones para controlar manipuladores robóticos en los procesos de automatización de la industria 4.0. La meta consiste en diseñar un control de posición del motor con el objetivo de que pueda ser utilizado para gobernar la posición y orientación de un brazo robótico, a través de la manipulación de los motores que se suelen colocar en las articulaciones del autómata. El control de posición del motor estará gobernado por un controlador de posición acelerado por *Hardware (IP-Core);* para realizar el estudio se hará énfasis en la función de transferencia que relaciona la posición del eje del rotor y el voltaje de entrada $\frac{\theta(s)}{V(s)}$, ya que es la variable

que se desea controlar, las otras funciones de transferencia no serán tomadas en cuenta en el estudio.

$$\frac{\theta(s)}{V(s)} = \frac{K_t}{(B_m \cdot R + K_e \cdot K_t)s + (B_m \cdot L + J \cdot R)s^2 + J \cdot L \cdot s^3} \quad (15)$$

Los parámetros del motor que se utilizaran para este estudio están mostrados en la Tabla 1, los cuales fueron obtenidos de dos estudios previos presentados en (Almatheel & Abdelrahman, 2017) y (Suman & Giri, 2016).

Tabla 1

Parámetros del motor DC

Parámetro	Valor
Inductancia de armadura ($\boldsymbol{L}$)	0.1215 H
Resistencia de Armadura ($\boldsymbol{R}$)	11.2 Ω
Inercia del Rotor ($\boldsymbol{J}$)	$0.02215\ kgm^2$
Voltaje de armadura Nominal ($\boldsymbol{V_a}$)	$240\ V$
Coeficiente de fricción viscoso ($\boldsymbol{B_m}$)	$0.002953\ Nm\frac{s}{rad}$
Constante de torque del motor ($\boldsymbol{K_t}$)	$1.28\ N\frac{m}{A}$
Cantante Back emf ($\boldsymbol{K_e}$)	$1.28\ V\frac{s}{rad}$

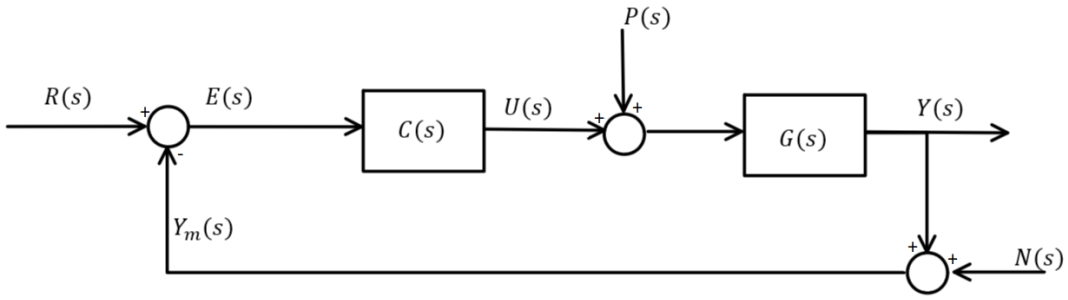
Descripción del Problema de Control

En el desarrollo y evolución de la industria no se ha pasado por alto la importancia de los controladores en los sistemas de manufactura, puesto que permiten mantener las variables del proceso en los valores deseados, para conservar una elevada calidad de los procesos industriales. Desde 1940, se han introducido los controladores comerciales *PID* (Vilanova, 2016), y pese a que han transcurrido varios años siguen siendo uno de los algoritmos de control más común utilizado en la industria de procesos debido a su sencillez y robustez. La estructura básica de un sistema de control realimentado lineal

analógico es la mostrada en la Figura 12. Donde R(s) es la referencia, es decir el valor deseado en la salida $Y(s)$, $P(s)$ es una perturbación que afecta al proceso y que se modela. En este caso de estudio, se considerará al torque en la carga (τ_L) como una perturbación al sistema del motor; el modelo lineal del proceso $G(s)$, incluye el sensor y el actuador. El controlador es un sistema lineal definido por su función de transferencia $C(s)$ y $N(s)$ puede considerarse como una perturbación de ruido provocado por el sensor en la medición de la señal de salida. Por lo tanto, la señal medida del sensor es $Y_m(s)$.

Figura 12

Modelo de un sistema de control lineal retroalimentado analógico



Nota. Adaptado de *PID* control: New identification and design methods (p. 57), por M. Johnson, M. Moradi, 2005

Para gobernar el sistema de posición del motor, se van a realizar dos escenarios con dos variantes del controlador *PID*. El primer escenario, estará regido por el controlador *PID* de un grado de libertad, para el segundo escenario se utilizará un controlador *PID* de dos grados de libertad.

Descripción Controlador PID de un grado de libertad

Para el controlador *PID* de un grado de libertad, se puede proporcionar ciertas notaciones y parámetros que permiten distinguir la forma de implementar el algoritmo de control, de modo que posteriormente se suministren las relaciones de conversión entre

las variantes de los algoritmos. Se puede construir el controlador *1DoF-PID* de dos maneras, con una estructura paralela y una estándar (Johnson et al., 2005), la estructura paralela del controlador en el dominio del tiempo está dada por la ecuación (16).

$$u(t) = K_p \cdot e(t) + K_i \int_0^t e(\tau)d\tau + K_d \frac{de(t)}{dt} \tag{16}$$

Donde K_p es la ganancia proporcional, K_i es la ganancia integral, K_d es la ganancia derivativa del controlador, $e(t)$ es el error del proceso y $u(t)$ es la acción de control. El modelo del controlador *PID* en su estructura paralela está determinada por (17) en el dominio de Laplace.

$$U(s) = \left[K_p + K_i \frac{1}{s} + K_d \cdot s\right] E(s) \tag{17}$$

En la forma estándar, el controlador *1DoF-PID* utiliza constantes de tiempo; la estructura del controlador *PID* estándar está representada por la ecuación (18).

$$u(t) = K_p \left[e(t) + \frac{1}{\tau_i}\int_0^t e(\tau)d\tau + \tau_d \frac{de(t)}{dt}\right] \tag{18}$$

El modelo del controlador *1DoF-PID* en su estructura estándar está dado por (19) en el dominio de Laplace.

$$U(s) = K_p \left[1 + \frac{1}{\tau_i \cdot s} + \tau_d \cdot s\right] E(s) \tag{19}$$

Donde τ_i es la constante de tiempo integral, τ_d es la constante de tiempo derivativa, la señal de error $e(t)$ está dada por la expresión (20) o su equivalente en el dominio de Laplace por la expresión (21).

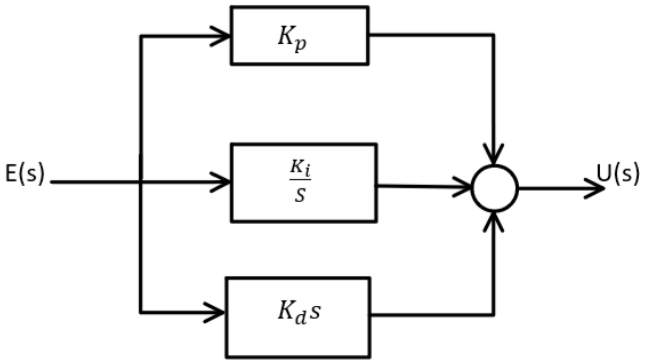
$$e(t) = r(t) - y(t) \tag{20}$$

$$E(s) = R(s) - Y(s) \tag{21}$$

Donde $R(s)$ es la referencia o valor deseado del sistema y $Y(s)$ es la señal de salida medida de la planta. La Figura 13 muestra el diagrama de bloques de un controlador *PID* con estructura paralela en su arquitectura.

Figura 13

Estructura paralela del controlador PID de un grado de libertad



Las ganancias del controlador tienen un propósito en la dinámica del sistema, la ganancia proporcional tendrá el efecto de disminuir el tiempo de subida, el cual, reducirá, pero nunca se eliminará, el error de estado estable. La ganancia integral se encarga de mitigar el error de estado estable, sin embargo, puede llegar a deteriorar la respuesta dinámica del sistema; la ganancia diferencial o derivativa mejora la estabilidad relativa del sistema y disminuye el sobre impulso, dando lugar a una mejor respuesta transitoria.

Descripción del Controlador PID de Dos Grados de Libertad

Al igual que el controlador *1DoF-PID*, el controlador *PID* de dos grados de libertad cuenta con una estructura paralela y estándar (Vilanova, 2016). La estructura paralela del controlador en el dominio de Laplace está dada por la ecuación (22).

$$U(s) = \left(\beta K_p + \frac{K_i}{s} + \gamma K_d s\right) R(s) - \left(K_p + \frac{K_i}{s} + K_d s\right) Y(s) \qquad (22)$$

Donde $\beta \; y \; \gamma$ son pesos de la referencia en la ganancia proporcional y derivativa del controlador. Normalmente γ es cero para evitar un cambio instantáneo brusco en la

señal de salida del controlador cuando ocurre un cambio de referencia $R(s)$ (Vilanova, 2016). Por lo tanto, la ecuación (22) se reduce a la siguiente expresión.

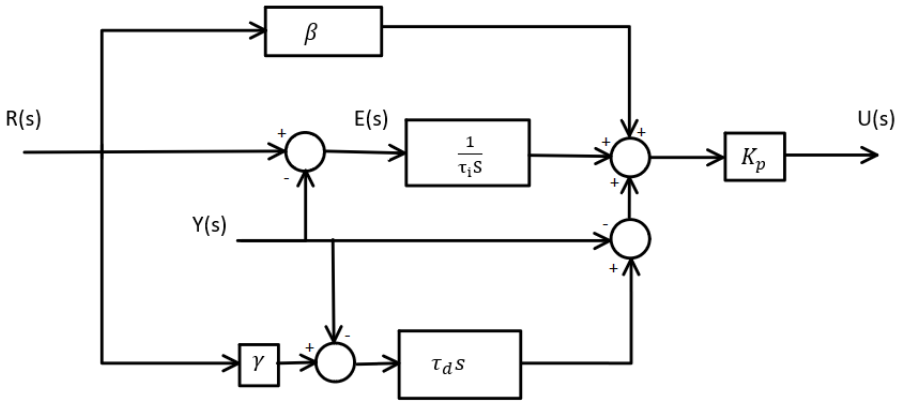
$$U(s) = \left(\beta K_p + \frac{K_i}{s}\right) R(s) - \left(K_p + \frac{K_i}{s} + K_d s\right) Y(s) \tag{23}$$

El modelo del controlador *2DoF-PID* en su estructura estándar esta dado por (24) en el dominio de Laplace. La Figura 14 muestra la arquitectura del controlador de dos grados de libertad con estructura estándar.

$$U(s) = K_p \left(\beta + \frac{1}{\tau_i s}\right) R(s) - K_p \left(1 + \frac{1}{\tau_i s} + \tau_d s\right) Y(s) \tag{24}$$

Figura 14

Arquitectura del controlador 2DoF-PID estándar



Se puede hallar el controlador *PID* estándar equivalente a través de las ganancias de la estructura paralela, tanto para el controlador *1DoF-PID* y el *2DoF-PID* mediante las siguientes relaciones.

$$K_p = K_p \tag{25}$$

$$\tau_i = \frac{K_p}{K_i} \tag{26}$$

$$\tau_d = \frac{K_d}{K_p} \tag{27}$$

Diseño y Sintonización de los Parámetros de los Controladores *PID*

Sintonización de Parámetros PID de un Grado de Libertad.

Para este estudio se realizará la sintonización del controlador *1DoF-PID* con estructura estándar presentada en (19), por lo tanto, se busca determinar los valores de los parámetros $K_p, \tau_i \; y \; \tau_d$, que permiten modificar la dinámica de la planta hacia la deseada. Normalmente, se busca que la dinámica del sistema controlador-planta en lazo cerrado, reproduzca la dinámica deseada de un sistema de segundo orden representado por la ecuación (28). Los parámetros de desempeño para modelar la dinámica esperada del sistema se establecen de forma general mediante: el tiempo de establecimiento (t_{ss}), el porcentaje de sobre impulso $(\%MP)$ y el error de estado estacionario (e_{ss}). Debido a que se utiliza un controlador *PID*, la acción integral produce un error estacionario prácticamente de cero, mientras que t_{ss} y $\%MP$ están en función del factor de amortiguamiento (ξ) y la frecuencia natural no amortiguada (ω_n). Estos parámetros influyen directamente en la ubicación de los polos dominantes del sistema de segundo orden, que son los que influyen directamente en la estabilidad y el comportamiento transitorio del sistema en lazo cerrado. Por lo tanto, el objetivo para la sintonización del controlador, es poder encontrar los parámetros del *1DoF-PID*, de manera que los polos dominantes del sistema en lazo cerrado reproducen la dinámica deseada.

$$P(s) = \frac{\omega_n^2}{s^2 + 2\xi\omega_n s + \omega_n^2} \tag{28}$$

Debido a que el comportamiento transitorio del sistema de la ecuación (28) depende básicamente del denominador, se define al polinomio deseado como ΔP_d, el cual puede ser calculado empleando (29). Como resultado se obtiene el polinomio que contiene la dinámica deseada del sistema de lazo cerrado.

$$\Delta P_d = s^2 + 2\xi\omega_n s + \omega_n^2 \tag{29}$$

El tiempo de estabilización para un sistema de segundo orden, en el que la respuesta de salida del sistema alcanza el rango de 2% del valor final, puede ser calculado de manera aproximada utilizando (30). Para calcular el porcentaje máximo de sobre impulso se puede emplear la ecuación (31). Cabe aclarar que t_{ss} y $\%MP$ son valores de diseño conocidos, por lo tanto, se puede emplear (30) y (31) para calcular los parámetros ξ y ω, los cuales establecen la dinámica deseada del sistema, véase la ecuación (29).

$$t_{ss} \cong \frac{4}{\xi\omega_n} \tag{30}$$

$$\%MP = 100e^{\frac{-\xi\pi}{\sqrt{1-\xi^2}}} \tag{31}$$

El sistema del motor obtenido en (15), se puede simplificar como una planta $G(s)$ de tercer orden de la forma mostrada en (32).

$$G(s) = \frac{K}{s(s^2 + as + b)} \tag{32}$$

Donde:

$$K = \frac{K_t}{J \cdot L} \tag{33}$$

$$a = \frac{B_m \cdot L \ + \ J \cdot R}{J \cdot L} \tag{34}$$

$$b = \frac{B_m \cdot R \ + \ K_e \cdot K_t}{J \cdot L} \tag{35}$$

La función de transferencia del controlador *1DoF-PID* con estructura estándar mostrada en (19), se puede representar como la expresión mostrada en la ecuación (36) o en su defecto para un análisis más simplificado se puede emplear la ecuación (37).

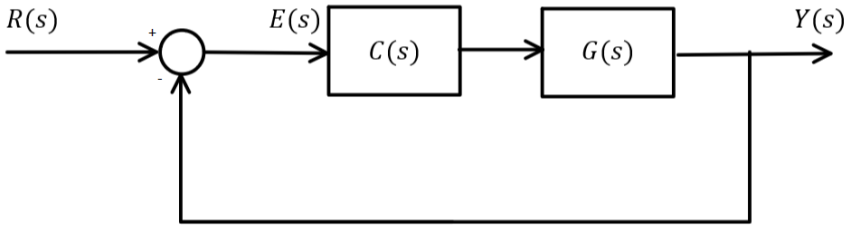
$$C(s) = \left[\frac{\tau_d K_p s^2 + K_p s + \frac{K_p}{\tau_i}}{s}\right] \tag{36}$$

$$C(s) = \left[\frac{d_2 s^2 + d_1 s + d_0}{s}\right] \tag{37}$$

La Figura 15 muestra el diagrama de bloques del sistema retroalimentado gobernado por el controlador *1DoF-PID*. Empleado el diagrama de bloques se puede obtener la función de transferencia mostrada en (38), la cual relaciona la salida del sistema $Y(s)$ y la referencia $R(s)$.

Figura 15

Diagrama de bloques del sistema gobernado por un controlador 1DoF-PID.



$$\frac{Y(s)}{R(s)} = \frac{C(s)G(s)}{1 + C(s)G(s)} \tag{38}$$

A partir de la ecuación (32), (37) y (38), se obtiene la función de transferencia del sistema servo-controlado en lazo cerrado mostrada a continuación:

$$\frac{Y(s)}{R(s)} = \frac{K(d_2 s^2 + d_1 s + d_0)}{s^4 + a\, s^3 + (b + K\, d_2)\, s^2 + K\, d_1\, s + K\, d_0} \tag{39}$$

El polinomio característico del sistema obtenido en (39), se muestra en la ecuación (40), como resultado se obtiene un polinomio de cuarto orden, por lo tanto, la ecuación del polinomio característico deseado, debe ser del mismo grado. Por esta razón, al polinomio característico presentado en (29), se le debe añadir dos polos adicionales para que este polinomio se convierta en un polinomio de cuarto grado. Los polos p adicionales, deben estar alejados de 5 a 10 veces de los polos dominantes del sistema (los polos más cercanos al origen) (Kuo & Golnaraghi, n.d.). Con este criterio los polos alejados se ubicarán en la zona de polos insignificantes, los cuales tienen un efecto despreciable ante el comportamiento generado por los polos dominantes. La ecuación (41) representa el polinomio deseado de orden 4, donde los polos dominantes dependen de los parámetros de desempeño deseado, y los polos no dominantes se encuentran en una zona cuyos efectos son insignificantes en la dinámica del sistema.

$$\Delta P_{1DoF}(s) = s^4 + a\,s^3 + (b + K\,d_2)\,s^2 + K\,d_1\,s + K\,d_0 \tag{40}$$

$$\Delta P_d(s) = (s+p)^2(s^2 + 2\,\xi\,\omega_n\,s + \omega_n^2) \tag{41}$$

Trabajando la expresión (41) se obtiene el siguiente resultado:

$$\begin{aligned}\Delta P_d = s^4 + (2p + 2\xi\omega_n)\,s^3 + (p^2 + 4p\xi\omega_n + \omega_n^2)\,s^2 \\ + (2p^2\xi\omega_n + 2p\omega_n^2)\,s + p^2\omega_n^2\end{aligned} \tag{42}$$

Igualando cada término del polinomio de la ecuación (40) y (42), se obtiene el sistema de ecuaciones mostrado en (43)

$$\begin{aligned} s^3&: \quad a = 2p + 2\xi\omega_n \\ s^2&: \quad b + Kd_2 = p^2 + 4p\xi\omega_n + \omega_n^2 \\ s^1&: \quad Kd_1 = 2p^2\xi\omega_n + 2p\omega_n^2 \\ s^0&: \quad Kd_0 = p^2\omega_n^2 \end{aligned} \tag{43}$$

Resolviendo el sistema de ecuaciones obtenemos los parámetros auxiliares del controlador (d_0, d_1, d_2), además se debe cumplir la condición de que la parte real de polo p sea 5 a 10 veces mayor a los polos dominantes del sistema.

$$d_0 = \left(\frac{\omega_n^4}{K}\right)\xi^2 + \left(-\frac{a\omega_n^3}{K}\right)\xi + \frac{a^2\omega_n^2}{4K} \tag{44}$$

$$d_1 = \left(\frac{2\omega_n^3}{K}\right)\xi^3 + \left(-\frac{2a\omega_n^2}{K}\right)\xi^2 + \left(\frac{a^2\omega_n}{2K} - \frac{2\omega_n^3}{K}\right)\xi + \frac{a\omega_n^2}{K} \tag{45}$$

$$d_0 = \left(-\frac{3\omega_n^2}{K}\right)\xi^2 + \left(\frac{a * \omega_n}{K}\right)\xi + \frac{a^2 + 4\omega_n^2 - 4b}{4K} \tag{46}$$

$$p = -\xi\omega_n + \frac{a}{2} \tag{47}$$

Donde

$$K_p = d_1 \tag{48}$$

$$\tau_i = \frac{K_c}{d_0} \tag{49}$$

$$\tau_d = \frac{d_2}{K_c} \tag{50}$$

$$p \gg 5(-\xi\omega_n) \tag{51}$$

El resultado de la sintonización producen un sistema en lazo cerrado estable y con los parámetros de desempeño deseado, sin embargo, como se observa en el modelo de lazo cerrado presentado en (39), el modelo presenta ceros que dependen de los valores sintonizados del controlador, si bien los ceros no afectan en la estabilidad del sistema, estos pueden causar un sobre impulso incluso en sistemas sobre amortiguados (Kuo & Golnaraghi, n.d.; Leong & Doyle, 2017); por lo que el porcentaje de sobre impulso $\%MP$ deseado puede verse afectado por los ceros del sistema. Este fenómeno puede reducirse significativamente al utilizar un controlador *PID* de dos grados de libertad como se mostrará posteriormente. La Tabla 2 muestra los parámetros de desempeño deseado del sistema en lazo cerrado, proporcionados en términos de los indicadores que determinan el comportamiento dinámico del sistema, basados en los requerimientos impuestos de diseño.

Tabla 2

Parámetros de diseño de la respuesta transitoria deseada del sistema en lazo cerrado.

Parámetro	Símbolo	Rango de Valor deseado	Valor Utilizado
Tiempo de Establecimiento	t_{ss}	Menor a 1.5s	1.5s
Porcentaje de sobre impulso	$\%MP$	Menor a 5%	1%
Factor de amortiguamiento	ξ	Mayor a 0.691 y menor a 1	0.8261
Frecuencia natural no amortiguada	ω_n	Menor a 3.8641 Rad/s	3.2281

La Tabla 3 muestra los parámetros del controlador sintonizados a partir de los datos de la Tabla 2 y las ecuaciones (48), (49) y (50).

Tabla 3

Parámetros sintonizados del controlador 1DoF-PID

Parámetro	Símbolo	Valor Sintonizado
Ganancia proporcional	K_p	23.1151
Constante de tiempo integral	τ_i	0.5578s
Constante de tiempo derivativa	τ_d	0.1587s

La metodología basada en modelos permite realizar una serie de simulaciones puramente de *Software,* para realizar un análisis previo antes de ser implementado. La Figura 16 muestra el comportamiento dinámico del sistema controlado, para una entrada de escalón unitario. La Tabla 4 presenta un resumen del desempeño del sistema, así como los polos y ceros del mismo en lazo cerrado obtenidos del diagrama de polos y ceros, ver Figura 17. Como se puede evidenciar los polos no dominantes del sistema cumplen con el criterio de ser 5 o 10 veces mayor a los polos dominantes. Sin embargo, los ceros se encuentran cerca de los polos dominantes del sistema, esto genera que los dos ceros presentes en el modelo (39), generen un incremento del ancho de banda (Kuo & Golnaraghi, n.d.) y un aumento del porcentaje de sobre impulso; esto ocasiona que el $\%MP$ obtenido esté fuera del valor deseado, no obstante, el porcentaje de sobre impulso no resulta ser exageradamente mayor al 5% permitido, los demás parámetros de desempeño que fueron obtenidos, se encuentran dentro del rango deseado.

Figura 16

Respuesta transitoria del sistema gobernado por controlador 1DoF-PID ante una entrada de escalón unitario

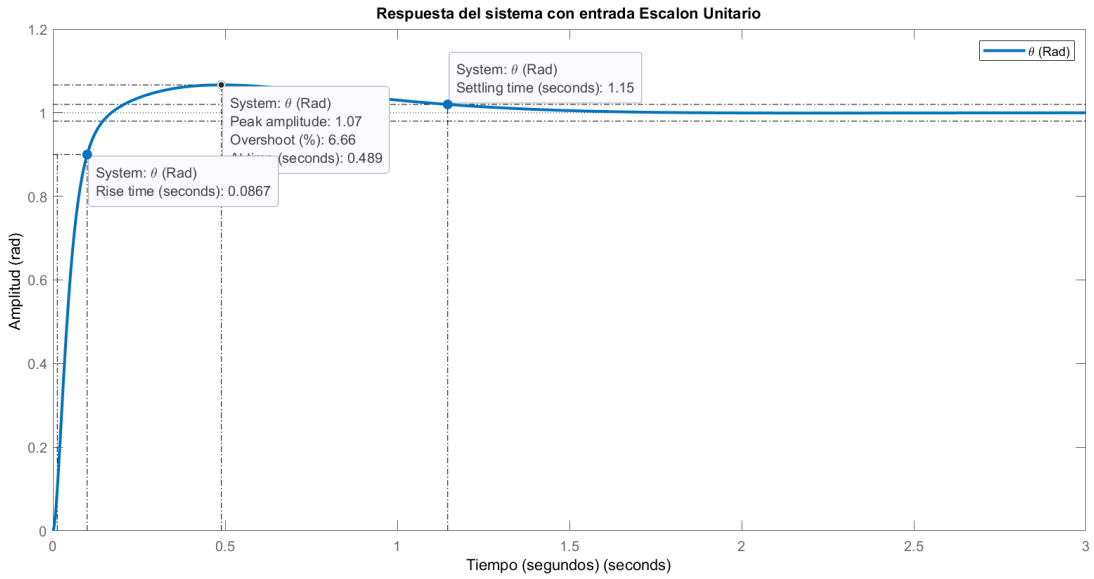


Figura 17

Diagrama de Polos y Ceros del sistema Gobernado por el controlador 1DoF-PID

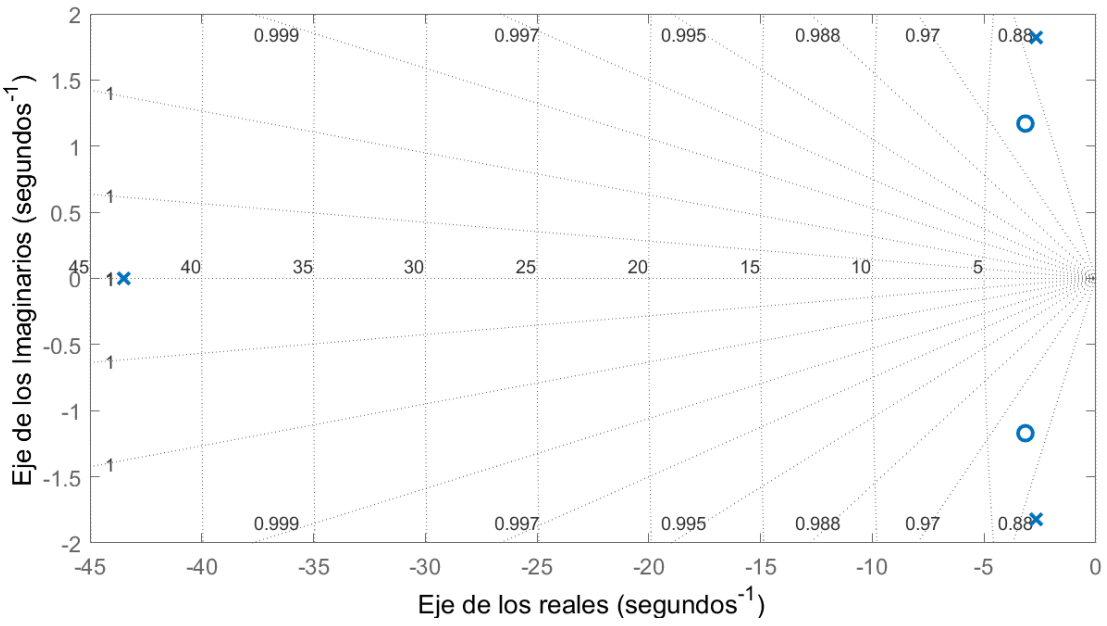


Tabla 4

Parámetros de la respuesta transitoria del sistema ante una entrada de escalón unitario y polos y ceros del sistema servo-controlado con compensador 1DoF-PID

Parámetro	Valor
Tiempo de Establecimiento (t_{ss})	$1.15\ s$
Porcentaje de sobre impulso ($\%MP$)	6.6%
Tiempo de subida (t_r)	0.0867 s
Ancho de banda (BW)	$24.3\ rad/s$
Polos Dominantes	$-2.6667 \pm 1.8192i$
Polos no dominantes	-43.4905
Ceros	$-3.1507 \pm 1.1704i$

Sintonización de Parámetros Controlador PID de Dos Grados de Libertad

Al igual que el caso del *1DoF-PID* se realizará la sintonización del controlador *PID* de dos grados de libertad con estructura estándar, con la posibilidad de transformar a su forma paralela utilizando las relaciones (25), (26) y (27). El objetivo de control es establecer el valor de los parámetros $K_p, \tau_i, \tau_d\ y\ \beta$, que permitan al sistema controlado en lazo cerrado, producir la dinámica deseada de un sistema de segundo orden. Los

parámetros de diseño impuesto son los mismos que en el caso del controlador de un grado de libertad, es decir t_{ss}, $\%MP$ y e_{ss}, al tener una acción integral, el error de estado estable prácticamente es cero. Por otro lado, se puede emplear las relaciones (30) y (31) para calcular el factor de amortiguamiento (ξ) y la frecuencia natural no amortiguada (ω_n), empleando t_{ss} y $\%MP$ como parámetros diseño conocidos.

La salida del controlador *2DoF-PID* con estructura estándar presentada en (24), (19) puede ser reordenada para propósitos de análisis como:

$$U(s) = C_R(s)R(s) - C_Y(s)Y(s) \tag{52}$$

Donde:

$$C_R(s) = K_p\left(\beta + \frac{1}{\tau_i s}\right) \tag{53}$$

$$C_Y(s) = K_p\left(1 + \frac{1}{\tau_i s} + \tau_d s\right) \tag{54}$$

Empleado el diagrama de bloques mostrado en la Figura 18, se puede calcular la función de transferencia de lazo cerrado de $G(s)$ y $C_y(s)$, véase la ecuación (55).

$$\frac{Y(s)}{R'(s)} = \frac{G(s)}{1 + C_Y(s)G(s)} \tag{55}$$

Donde $R'(s)$ es la salida del bloque $C_R(s)$. La función de transferencia de lazo cerrado mostrada en (55) en serie con la función de transferencia $C_R(s)$, al realizar la operación entre bloques se obtiene la función de transferencia mostrada en (56), la cual relaciona la salida $Y(s)$ y la referencia $R(s)$. A continuación, se puede realizar el análisis para el ajuste de los parámetros del controlador.

$$\frac{Y(s)}{R(s)} = \frac{C_R(s)G(s)}{1 + C_Y(s)G(s)} \tag{56}$$

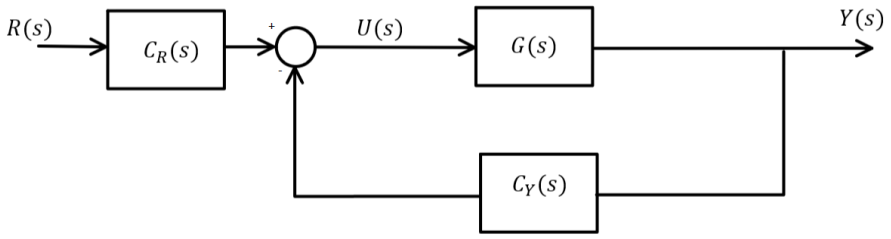
Para un análisis más simplificado, las expresiones obtenidas en las ecuaciones (53) y (54), pueden ser reducidas a las expresiones mostradas en (57) y (58).

$$C_R(s) = \frac{\beta K_p s + \frac{K_p}{\tau_i}}{s} = \frac{\beta d_1 s + d_0}{s} \tag{57}$$

$$C_Y(s) = \frac{K_p \tau_d s^2 + K_p s + \frac{K_p}{\tau_i}}{s} = \frac{d_2 s^2 + d_1 s + d_0}{s} \tag{58}$$

Figura 18

Diagrama de bloques del sistema gobernado por un controlador 2DoF-PID



A partir de las ecuaciones (32) (57) y (58) reemplazadas en (56), se obtiene la función de transferencia del sistema servo-controlado en lazo cerrado mostrado en (59).

$$\frac{Y(s)}{R(s)} = \frac{(K\beta d_1)s \; + \; K d_0}{s^4 + \; a s^3 + \; (b \; + \; K \, d_2) s^2 + \; K \, d_1 \, s \; + \; K \, d_0} \tag{59}$$

Donde el polinomio característico del sistema es:

$$\Delta P_{2DoF}(s) = s^4 + \; a s^3 + \; (b \; + \; K \, d_2) s^2 + \; K \, d_1 \, s \; + \; K \, d_0 \tag{60}$$

Al igual que en el caso del controlador *1DoF-PID*, el polinomio característico del sistema obtenido en (59), es de orden 4. Por lo tanto, el polinomio deseado debe ser del mismo grado, por lo que se puede utilizar (41). Curiosamente, la ecuación característica

del sistema gobernado con el controlador *2DoF-PID* mostrado en (60), es la misma que la obtenida en el análisis del controlador *1DoF-PID* obtenido en (40). Es así que, al igualar la ecuación (60) y (42), se obtiene el mismo sistema de ecuaciones presentadas en (43). Por esta razón, se puede utilizar las mismas expresiones obtenidas en (44), (45), (46), (48), (49) y (50) para obtener los parámetros K_p, τ_i, τ_d. Hay que considerar que el controlador de dos grados de libertad depende de un parámetro adicional β, para el análisis de sintonización del parámetro β, se puede utilizar el siguiente criterio: colocar el cero del sistema en la región donde sea insignificante su efecto sobre la dinámica del sistema, o en su defecto ubicarlo de tal manera que se cancele con uno de los polos no dominantes; como resultado se obtiene un mejor desempeño global del sistema servo-controlado.

Para calcular el valor de β, se utiliza el numerador de la función de transferencia obtenida en (59). Por otro lado, la ecuación (61) muestra el factor de uno de los polos no dominantes extraído de (41) e igualada a cero. Para colocar el cero del sistema lejos de los polos característicos y encima de uno los polos no dominantes del sistema, se debe igualar las ecuaciones (61) y (62) y se despejar β, véase la ecuación (63). Como se puede observar, β está en función de datos conocidos, dado que p y τ_i, se obtienen después de resolver el sistema de ecuaciones mostrado en (43); como resultado, el efecto del cero es insignificante en la dinámica del sistema.

$$s + \frac{d_0}{\beta d_1} = 0 \tag{61}$$

$$s + p = 0 \tag{62}$$

$$\beta = \frac{d_0}{p d_1} = \frac{1}{p\ \tau_i} \tag{63}$$

La Tabla 5 muestra los parámetros de desempeño deseado sobre el sistema en lazo cerrado, desencadenando los parámetros que describen la dinámica deseada del

sistema. La Tabla 6 muestra los parámetros del controlador sintonizados a partir de los datos de la Tabla 5 y las ecuaciones obtenidas en (48), (49) y (50).

Tabla 5

Parámetros de diseño de la dinámica deseada del sistema en lazo cerrado con controlador 2DoF-PID.

Parámetro	Símbolo	Rango de Valor deseado	Valor Utilizado
Tiempo de Establecimiento	t_{ss}	Menora 1.5s	1.5s
Porcentaje de sobre impulso	$\%MP$	Menor a 5%	1%
Factor de amortiguamiento	ξ	Mayor a 0.691 y menor a 1	0.8261
Frecuencia natural no amortiguada	ω_n	Menor a 3.8641 Rad/s	3.2281

Tabla 6

Parámetros del controlador 2DoF-PID

Parámetro	Símbolo	Valor Sintonizado
Ganancia proporcional	K_p	23.1151
Constante de tiempo integral	τ_i	0.5578 s
Constante de tiempo derivativa	τ_d	0.1587 s
Peso de la referencia en la ganancia proporcional	β	0.0412

Mediante una simulación de *Software* se puede encontrar la respuesta del sistema ante una entrada de escalón unitario. La Figura 19 muestra la respuesta transitoria del sistema gobernado con el controlador *2DoF-PID* ante una entrada de escalón unitario; la dinámica transitoria resultante posee un mejor comportamiento que el obtenido con el compensador *1DoF-PID*, esto debido a que el modelo servo-controlado presentado en (59) tiene un solo cero que puede ser colocado de manera que no tenga efectos sobre la

dinámica dominante del sistema. La Figura 20 muestra como los polos no dominantes y el cero se encuentran alejado más de 10 veces de los polos dominantes del sistema

Figura 19

Respuesta transitoria del sistema gobernado por controlador 2DoF-PID ante una entrada de escalón unitario

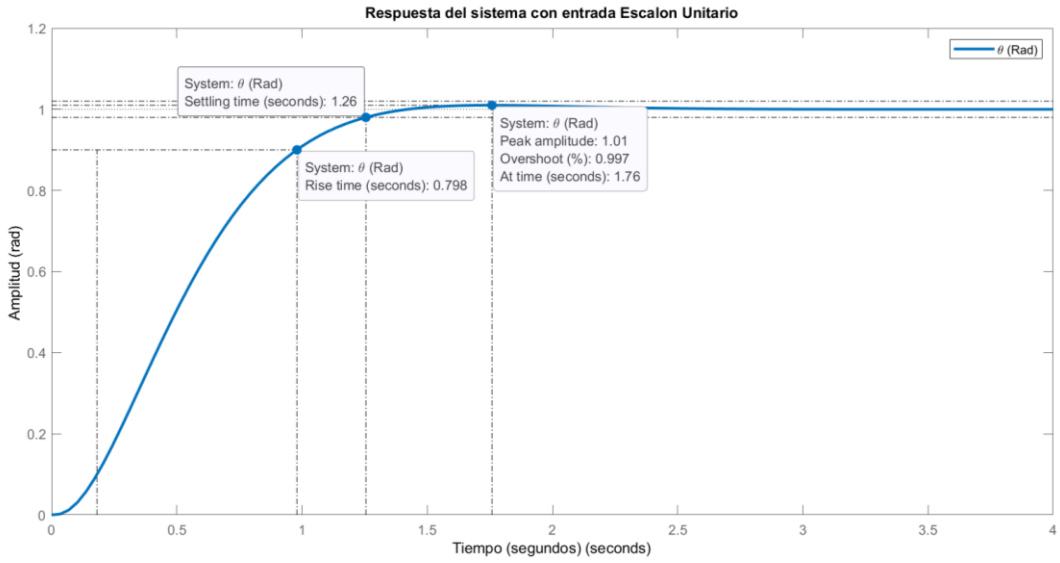
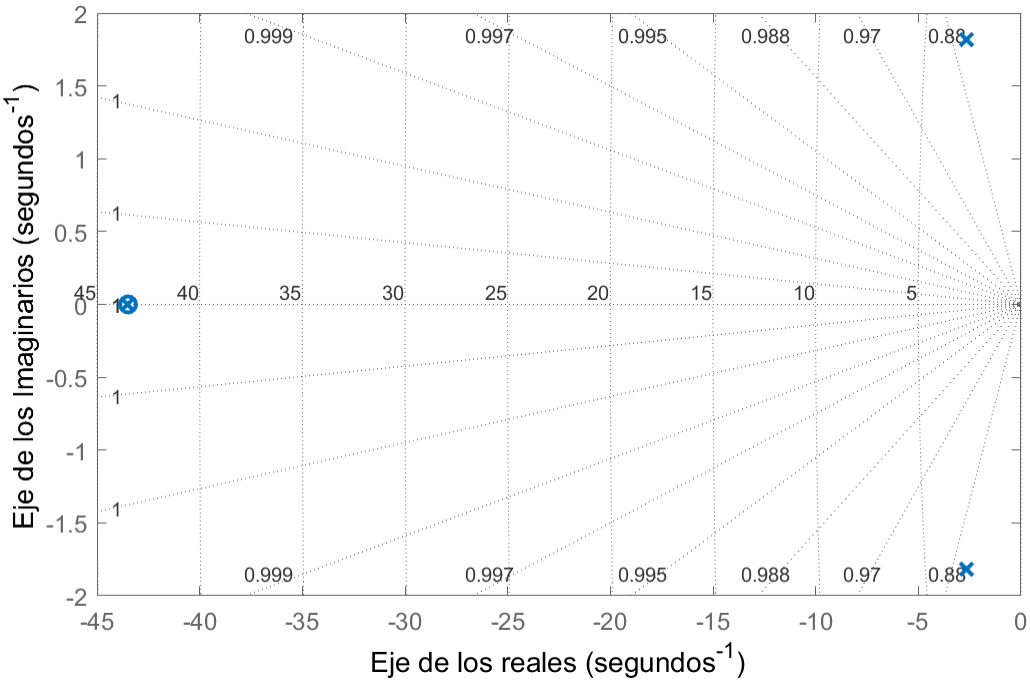


Figura 20

Diagrama de Polos y Ceros del sistema gobernado con el controlador 2DoF-PID



La Tabla 7 muestra un resumen de los parámetros de desempeño del sistema controlado en lazo cerrado, los parámetros $\boldsymbol{t_{ss}}$ y $\%\boldsymbol{MP}$ se encuentran dentro del rango deseado de diseño, este desempeño resulta ser superior al obtenido con el controlador *1DoF-PID*. Esto se debe a que se eliminó el efecto de los ceros sobre el sistema servo-controlado, ubicando el único cero dentro de la región despreciable en la dinámica del sistema; para este estudio el cero se ubica en misma posición que los polos alejados, en otras palabras, este efecto representa una especie de cancelación del polo y del cero no dominante, provocando que la dinámica del sistema sea influenciada prácticamente solo por los polos dominantes que generan la dinámica deseada del sistema controlado.

Tabla 7

Parámetros de la respuesta transitoria del sistema ante una entrada de escalón unitario y polos y ceros del sistema servo-controlado con compensador 2DoF-PID

Parámetro	Valor
Tiempo de Establecimiento (t_{ss})	$1.26\ s$
Porcentaje de sobre impulso ($\%MP$)	0.99%

Tiempo de subida (t_r)	$0.798\ s$
Ancho de banda (BW)	$2.69\ rad/s$
Polos Dominantes	$-2.6667 \pm 1.8192i$
Polos no dominantes	-43.4905
Ceros	-43.4905

Digitalización de Controladores

En los sistemas modernos y en la práctica actual es común implementar los algoritmos del controlador *PID* en un sistema basado en microprocesadores operados con señales digitales (Podržaj, 2018). Debido a la tecnología basada en la electrónica digital, la implementación de los controladores se ha vuelto más sencilla, dejando atrás circuitos analógicos basados en amplificadores operacionales, resistencias y capacitores, que provocan una implementación compleja y que en la práctica dificultan realizar algún ajuste al controlador una vez implementado. Por estas razones se tiene que discretizar los controladores para ser ejecutados en una plataforma de electrónica digital.

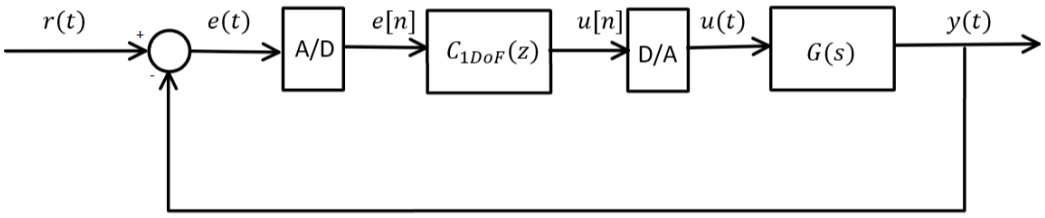
Discretización del Controlador PID de Un Grado de Libertad

El bucle de controlador digital mostrado en la Figura 21 es similar al bucle de control analógico convencional, la diferencia radica en que la señal de error $e(t)$ se muestrea en el tiempo, y se cuantifica en amplitud mediante un convertidor de analógico a digital (A/D). El muestreo (A/D) generalmente ocurre a una velocidad constante, a la que se denomina frecuencia de muestreo f_s , que es el inverso del periodo de muestreos T_s, como se muestra en (64). El proceso de digitalización del controlador permite que el compensador sea implementable en un sistema electrónico digital, para el análisis del modelo del controlador se utilizará el dominio de tiempo discreto y la transformada Z. El bloque C_{1DoF} es el controlador *PID* de un grado de libertad y $G(s)$ es la planta del motor modelada.

$$f_s = \frac{1}{T_s} \tag{64}$$

Figura 21

Bucle de control digital con controlador 1DoF-PID



La forma estándar del controlador *1DoF-PID* mostrado en (18), está constituido por la suma de tres acciones de control: la proporcional *(P)*, la integral *(I)* y la derivativa *(D).* Para un análisis simplificado se realizará la digitalización de cada una de estas acciones del controlador *1DoF-PID*.

Discretización De La Acción Proporcional. La acción proporcional del controlador *1DoF-PID* estándar está determinada por:

$$u_p(t) = K_p e(t) \tag{65}$$

Donde u_p es la acción proporcional del controlador estándar. Al realizar la digitalización de la acción proporcional, el análisis pasa a ser en el dominio del tiempo discreto. La ecuación (66) representa la acción proporcional en tiempo discreto, al realizar la transformada Z a partir de (66), se obtiene (67). La función de transferencia de la acción proporcional mostrada en (68) se puede obtener despejando (66).

$$u_p(k) = K_p e(k) \tag{66}$$

$$U_p(z) = K_p E(z) \tag{67}$$

$$\frac{U_p(z)}{E(z)} = K_p \tag{68}$$

Discretización De La Acción Integral. La acción Integral del controlador *1DoF-PID* estándar está dada por:

$$u_i(t) = \frac{K_p}{\tau_i}\int_0^t e(\tau)d\tau \tag{69}$$

Donde u_i es la acción integral del controlador *1DoF-PID*. Para el análisis en el dominio del tiempo discreto, la integral puede aproximarse como una suma infinitesimal de trapecios, lo que se conoce como aproximación trapezoidal. Esta estrategia proviene de un concepto de aproximación por métodos numéricos, para posteriormente trasladar el análisis al dominio de la transformada z. La acción integral aproximada está determinada por la ecuación (70).

$$u_i(k) = \frac{K_p}{\tau_i}\sum_{h=0}^{k} T_s\left[\frac{e[h]+e[h-1]}{2}\right] \tag{70}$$

Donde $T_s\left[\frac{e[h]+e[h-1]}{2}\right]$, representa la semisuma de las bases por la altura, fórmula del área de un trapecio, donde las bases son $e[h]$ y $e[h-1\}$, y la altura viene dado por el periodo de muestreo T_s. La ecuación (71) es exactamente igual a la expresión (70), pero retardada una muestra.

$$u_i(k-1) = \frac{K_p}{\tau_i}\sum_{h=0}^{k} T_s\left[\frac{e[h-1]+e[h-2]}{2}\right] \tag{71}$$

Si se resta la ecuación (71) de (70) se obtiene el siguiente resultado:

$$u_i(k) - u_i(k-1) = \frac{K_p}{\tau_i}\frac{T_s}{2}\sum_{h=0}^{k} e[h-1] - e[h-2] \tag{72}$$

Realizando la sumatoria mostrada en (72) se obtiene la siguiente expresión:

$$u_i(k) - u_i(k-1) = \frac{T_s}{2}\frac{K_p}{\tau_i}\,[e[k] + e[k-1]] \tag{73}$$

Empleando la transformada z en (73) se obtiene (74), despejando se obtiene la función de transferencia de la acción integral mostrada en (75).

$$U_i(z)(1 - z^{-1}) = \frac{T_s}{2}\frac{K_p}{\tau_i}\,E(z)(1 + z^{-1}) \tag{74}$$

$$\frac{U_i(z)}{E(z)} = \frac{T_s}{2}\frac{K_p}{\tau_i}\frac{(1 + z^{-1})}{(1 - z^{-1})} \tag{75}$$

Discretización De La Acción Derivativa. La acción derivativa del controlador *1DoF-PID* estándar, está determinada como:

$$u_d(t) = K_p\tau_d\frac{de(t)}{dt} \tag{76}$$

Donde u_d es la acción derivativa del controlador *1DoF-PID*; para el análisis del dominio de tiempo discreto, se puede utilizar el método de derivación simple para aproximar la derivada, tal como se muestra en la ecuación (77).

$$u_d(k) = K_p\tau_d\frac{e(k) - e(k-1)}{T_s} \tag{77}$$

Utilizando la transformada z en (77) se obtiene (78), Realizando las operaciones matemáticas respectivas se obtiene la función de transferencia de la acción derivativa como se muestra en (79).

$$U_d(z) = \frac{K_p\tau_d}{T_s}(1 - z^{-1})E(z) \tag{78}$$

$$\frac{U_d(z)}{E(z)} = \frac{K_p\tau_d}{T_s}(1 - z^{-1}) \tag{79}$$

Obtenida la discretización de la acción proporcional (68), integral (75) y derivativa (79), se las puede combinar para obtener la función de transferencia del modelo del controlador *PID* de un grado de libertad completamente digitalizado en su forma estándar, como se muestra en la ecuación (80).

$$\frac{U(z)}{E(z)} = K_p \left[1 + \frac{T_s}{2} \frac{K_p}{\tau_i} \frac{(1 + z^{-1})}{(1 - z^{-1})} + \frac{K_p \tau_d}{T_s} (1 - z^{-1}) \right] \tag{80}$$

Sin embargo, el modelo obtenido en (80), con la aproximación trapezoidal no es la única forma de realizar la integración aproximada, por lo que el modelo puede tener ligeras variaciones en la aproximación de la integral. Existen otras dos aproximaciones de la integral que son muy utilizadas, Euler en adelanto *(Forward Euler) dada por* (81) y Euler en atraso *(Backward Euler) dada por* (82) (Erickson & Maksimović, 2020). *La* Tabla 8 muestra un resumen de las funciones de transferencia en el dominio z de las aproximaciones de la acción integral más utilizadas.

$$u_i(k) = u_i(k-1) + \frac{K_p \, T_s}{\tau_i} e(k-1) \tag{81}$$

$$u_i(k) = u_i(k-1) + \frac{K_p \, T_s}{\tau_i} e(k) \tag{82}$$

Tabla 8

Aproximaciones numéricas de la acción integral.

Método de aproximación	*Función de Transferencia*
Trapezoidal	$\frac{T_s}{2}\frac{K_p}{\tau_i}\frac{(1+z^{-1})}{(1-z^{-1})}$
Euler en adelanto	$\frac{K_p\,T_s}{\tau_i}\frac{z^{-1}}{1-z^{-1}}$
Euler en atraso	$\frac{K_p\,T_s}{\tau_i}\frac{1}{1-z^{-1}}$

Al utilizar las diferentes aproximaciones de la integral mostrada en la Tabla 8 , el modelo obtenido en (80) no es el único, por lo que el modelo presentado (83) con aproximación *Euler en adelanto* y (84) con aproximación *Euler en atraso*, también serían modelos válidos del controlador *1DoF-PID*.

$$\frac{U(z)}{E(z)} = K_p\left[1 + \frac{K_p\,T_s}{\tau_i}\frac{z^{-1}}{1-z^{-1}} + \frac{K_p\tau_d}{T_s}(1-z^{-1})\right] \tag{83}$$

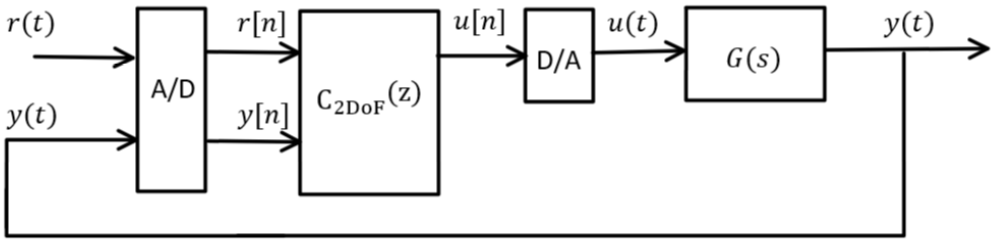
$$\frac{U(z)}{E(z)} = K_p\left[1 + \frac{K_p\,T_s}{\tau_i}\frac{1}{1-z^{-1}} + \frac{K_p\tau_d}{T_s}(1-z^{-1})\right] \tag{84}$$

Discretización Del Controlador PID De Dos Grados De Libertad

La Figura 22 muestra la arquitectura del sistema en lazo cerrado con el controlador *PID* de dos grados de libertad digitalizado; construido a partir del modelo presentado en (52), el cual está constituido por $C_R(S)$ y $C_Y(s)$ pero discretizados. Se puede obtener una acción de control a*soc*iado a la referencia $r[n]$ y otra a*soc*iada a la salida del sistema $y[n]$. La acción de control a*soc*iada a la señal de referencia está representada en la ecuación (85), y la a*soc*iada a la salida del sistema se muestra en la ecuación (86). La acción de control total del controlador *2DoF-PID* en el dominio del tiempo se puede calcular empleando (87).

Figura 22

Bucle de control digital con controlador 2DoF-PID



$$u_r(t) = K_p\left(\beta\, r(t) + \frac{1}{\tau_i}\int_0^t r(\tau)d\tau\right) \tag{85}$$

$$u_y(t) = K_p\left(y(t) + \frac{1}{\tau_i}\int_0^t y(\tau)d\tau + \tau_d\frac{dy(t)}{dt}\right) \tag{86}$$

$$u(t) = u_r(t) - u_y(t) \tag{87}$$

Donde $u_r(t)$ es la acción de control relacionada a la referencia, $u_y(t)$ es la acción del control relacionado a la salida del sistema y $u(t)$ es la acción del controlador *PID* de dos grados de libertad. Como se puede observar, $u_y(t)$ solamente tiene acción proporcional e integral, mientras que $u_r(t)$ posee una acción proporcional, integral y derivativa, por lo tanto, se puede aplicar el mismo criterio de discretización utilizado en el análisis del controlador *1DoF-PID*. En donde la acción derivativa utiliza el método de aproximación de derivación simple, por otra parte, el empleo de las diferentes aproximaciones de la acción integral presentada en la Tabla 8, permite obtener 3 modelos válidos para el controlador digital *2DoF-PID*. El primer modelo está representado en la ecuación (88) el cual utiliza una aproximación trapezoidal para la acción integral.

$$U(z) = C_{R_{Tp}}(z)R(z) - C_{Y_{Tp}}(z)Y(z) \tag{88}$$

Donde:

$$C_{R_{Tr}}(z) = K_p\left(\beta + \frac{T_s}{2}\frac{K_p}{\tau_i}\frac{(1+z^{-1})}{(1-z^{-1})}\right) \tag{89}$$

$$C_{R_{Tr}}(z) = K_p\left(1 + \frac{T_s}{2}\frac{K_p}{\tau_i}\frac{(1+z^{-1})}{(1-z^{-1})} + \frac{K_p\tau_d}{T_s}(1-z^{-1})\right) \tag{90}$$

El segundo modelo mostrado en la ecuación (91) utiliza la aproximación Euler en adelanto para la acción integral.

$$U(z) = C_{R_{Fw}}(z)R(z) - C_{Y_{Fw}}(z)Y(z) \tag{91}$$

Donde:

$$C_{R_{Fw}}(z) = K_p\left(\beta + \frac{K_p\,T_s}{\tau_i}\frac{z^{-1}}{1-z^{-1}}\right) \tag{92}$$

$$C_{Y_{Fw}}(z) = K_p\left(1 + \frac{K_p\,T_s}{\tau_i}\frac{z^{-1}}{1-z^{-1}} + \frac{K_p\tau_d}{T_s}(1-z^{-1})\right) \tag{93}$$

Finalmente, el tercer modelo expuesto en la ecuación (94), utilizando la aproximación Euler en atraso para la acción integral.

$$U(z) = C_{R_{Bw}}(z)R(z) - C_{Y_{Bw}}(z)Y(z) \tag{94}$$

Donde:

$$C_{R_{Bw}}(z) = K_p\left(\beta + \frac{K_p\,T_s}{\tau_i}\frac{1}{1-z^{-1}}\right) \tag{95}$$

$$C_{Y_{Bw}}(z) = K_p\left(1 + \frac{K_p\,T_s}{\tau_i}\frac{1}{1-z^{-1}} + \frac{K_p\tau_d}{T_s}(1-Z^{-1})\right) \tag{96}$$

Existen varios trabajos que los autores utilizan la aproximación de Euler en adelanto como el presentado en (Abdelhamid et al., 2017; Pathak et al., 2020), sin

embargo, el estudio presentado en (Soldati et al., 2016), utiliza una aproximación *Euler en atraso* en su discretización del controlador *1DoF-PID*, el cual fue implementado en un procesador ARM. El estudio señala que la aproximación *Euler en adelante* no siempre conserva la estabilidad que posee el sistema en tiempo continuo luego de la discretización, cosa que sí ocurre con la aproximación de *Euler en atraso.* Además, este enfoque produce una expresión más sencilla de utilizar en los algoritmos que otras propuestas. Esta es la motivación para utilizar la aproximación *Euler en atraso* en el modelado del *IP-Core* empleando los modelos del controlador *1DoF-PID* mostrada en (84) y *2DoF-PID* mostrada en (94). Para otros fines, el método de integración trapezoidal no debe ser despreciado en su totalidad, pues marcas reconocidas como *Siemens* o *Schneider-Electric* utilizan la discretización con aproximación trapezoidal en su algoritmo de controlador *PID* digital (Kovela et al., 2017).

Selección del Periodo de Muestreo

Un aspecto muy importante al momento de digitalizar un sistema de control, es hallar un periodo de muestreo adecuado, esto se debe, a que periodos de muestreo muy grandes puede desestabilizar el sistema, ya que pueden provocar que los polos en el dominio z se ubiquen fuera del círculo unitario (Ogata, 1995). De igual manera un tiempo de muestreo muy grande podría imposibilitar la reconstrucción de la señal en tiempo continuo; por otro lado, tiempos de muestreo muy cortos, incrementan el costo computacional en el dispositivo digital (Åström & Wittenmark, 2013). En el área de los sistemas de control no existe una fórmula exacta para encontrar el periodo de muestreo, la selección de muestreo se basa en métodos heurísticos como los que se presenta en (Ogata, 1995), donde considera que un periodo de muestreo adecuado en un sistema sobre amortiguado ante una entrada tipo escalón, es muestrear de ocho a diez veces durante el tiempo de levantamiento de la respuesta. Para esta investigación se utilizará

el criterio presentado por los autores en (Åström & Wittenmark, 2013), donde aborda la selección del periodo de muestreo en función de la respuesta de un sistema de control en tiempo continuo. Se utilizará el tiempo de subida del sistema controlador-planta ante una entrada de escalón unitario. El periodo de muestreo T_s está determinado por la ecuación (97).

$$T_s = \frac{t_r}{N_r} \tag{97}$$

Donde t_r es el tiempo de subida del sistema, N_r es el número de muestras por tiempo de subida. El autor sugiere elegir N_r entre 4 y 10. Por lo tanto el tiempo de muestreo puede estar en un rango como se lo muestra a continuación en (98)

$$\frac{t_r}{10} < T_s < \frac{t_r}{4} \tag{98}$$

Para un sistema de segundo orden con factor de amortiguamiento $0 < \xi < 1$ y frecuencia natural no amortiguada ω_n, el tiempo de subida se define como

$$t_r = \omega_n^{-1} e^{\frac{\varphi}{\tan(\varphi)}} \tag{99}$$

Donde $\varphi = Arccos(\xi)$. Para un sistema como $\xi \approx 0.7$, el periodo de muestreo se puede calcular como:

$$\frac{0.2}{\omega_n} < T_s < \frac{0.6}{\omega_n} \tag{100}$$

Análisis Del Periodo De Muestreo Para El Controlador *1dof-PID*. El periodo de muestreo para el controlador *1DoF-PID* se lo puede obtener utilizando directamente el criterio mostrado en (98), al observar la Tabla 4, se observa que $t_r = 0.0867\ s$. Por lo que

el rango sugerido para el periodo de muestreo queda determinado por la expresión (101). Se debe aclarar que el uso de la ecuación (99) para el cálculo del tiempo de subida se ve restringida debido a que la dinámica del sistema está influenciada por los ceros del sistema en lazo cerrado.

$$0.0087\ s < T_s < 0.0217\ s \tag{101}$$

Análisis Del Periodo De Muestreo Para El Controlador *2dof-Pid*. Para el caso del tiempo de muestreo del sistema gobernado por el controlador *2DoF-PID*, se puede utilizar el tiempo de subida presentado en la Tabla 7, o calculando de manera directa a través de la ecuación (99), porque para este caso, la dinámica del sistema no se ve afectada por los ceros y polos no dominantes, por tal motivo empleando la expresión (99) se ha calculado el tiempo de subida aproximado, que es muy similar al obtenido en la Tabla 7. El resultado del tiempo de subida aproximado se muestra en (102).

$$t_r = 0.7450 \tag{102}$$

Empleando (98) y (102) se calcula el periodo de muestreo sugerido para el controlador *2DoF-PID*, el cual se muestra ecuación (103).

$$0.0745\ s < T_s < 0.186\ s \tag{103}$$

Bajo este criterio podemos utilizar los valores sugeridos, sin embargo, el límite del periodo de muestreo mínimo para este caso donde se utiliza una *FPGA* no es tan obligatorio. Esto se debe a que el límite inferior sirve para limitar el costo computacional del dispositivo a implementar. Para este estudio, se utilizará una tecnología basada en plataformas reconfigurables para implementar los controladores, cuya capacidad computacional es superior a la de un microprocesador clásico, de modo que las tazas de muestreo pueden ser superior a las de un microcontrolador convencional. El periodo de muestreo más bajo, es el del sistema gobernado por el controlador *1DoF-PID*, debido a

que necesita un tiempo de muestreo mucho menor al requerido por el controlador *2DoF-PID*; para el caso del sistema gobernado por el controlador *1DoF-PID* se ha decidido utilizar un periodo de muestreo de $T_s = 0.01\ s$, el cual está dentro del rango recomendado en la expresión (101). Por otro lado, para el caso del sistema gobernado por el compensador *2DoF-PID*, se ha optado por utilizar el mismo periodo de muestreo del controlador *1DoF-PID*, ya que, aunque está por debajo del rango obtenido en (103), no supone un mayor costo computacional. La frecuencia de muestreo estaría dada por $f_s = \frac{1}{T_s} = 100Hz$, la cual es una frecuencia baja comparada con las frecuencias que pueden que soportan plataformas basadas en *Hardware* reconfigurable, que pueden alcanzar $100\ Mhz$ o frecuencias superiores (Digilent, 2016b, 2016a).

Resumen

En este capítulo se ha revisado las estructuras clásicas del controlador *PID* de uno y dos grados de libertad, se ha utilizado la forma estándar de los controladores para realizar el proceso de sintonización de los parámetros que constituyen a los controladores *1DoF-PID* y *2DoF-PID*. Por otro lado, se realizó una primera simulación y validación en *Software*, del comportamiento de todo el sistema de retroalimentación gobernado con cada uno de los controladores previamente sintonizados. Como se esperaba, la estrategia de control *2DoF-PID* presenta un mejor comportamiento en este tipo de sistemas servo-controlados, ya que al tener un parámetro de sintonía adicional se pueden ajustar los ceros del sistema, de manera que no influyan en la dinámica deseada del mismo. Si se observa el resumen de la Tabla 4 y Tabla 7, tanto el sistema gobernado por el controlador *1DoF-PID y el* controlador *2DoF-PID* tienen los mismos polos. Se sabe que los polos dominantes contienen la dinámica deseada del sistema, sin embargo, existen algunas diferencias en el desempeño de cada controlador. Para el caso del controlador *2DoF-PID,* el $\%MP$ es aproximadamente del 1%, mientras que el $\%MP$ del controlador *1DoF-PID* es

del 6,6%, este fenómeno ocurre por la ubicación que tienen los ceros del sistema en cada escenario. Para el escenario con el controlador *1DoF-PID*, los ceros del sistema se ubican muy cerca de los polos dominantes, esto provoca un aumento del porcentaje de sobre impulso, mientras que para el segundo escenario con el controlador *2DoF-PID*, el cero del sistema se ubica en uno de los polos no dominantes del sistema, en consecuencia, la dinámica resultante está prácticamente influenciada sólo por los polos dominantes, que contienen la dinámica deseada.

Si bien el análisis para la sintonización del controlador y la simulación del sistema fue realizado en el dominio del tiempo continuo, en la práctica no resulta factible su implementación, menos en un sistema basado en electrónica digital. Por tal motivo, se ha realizado el proceso de discretización de cada uno de los controladores para poderlos implementarlos en *Hardware*. En el proceso de digitalización, se ha discretizado la acción proporcional, integral y derivativa que constituye la forma estándar de cada compensador. Para el proceso de discretización de la acción proporcional se debe mantener la misma constante K_p, mientras que para digitalizar la acción derivativa se ha utilizado el método de derivación simple, finalmente para el proceso de discretización de la acción integral se presentó tres maneras de aproximar el proceso de integración. La primera manera es empleando la aproximación trapezoidal, la segunda forma es utilizando la aproximación *Euler en adelanto* y adicionalmente se puede utilizar la aproximación *Euler en atraso*. Cada una de las aproximaciones de la integral puede ser empleada de acuerdo a la necesidad planteada, sin embargo, se ha destacado el uso de la aproximación de Euler en atraso, la cual basada en la revisión bibliográfica señala la capacidad que tiene para garantizar la estabilidad del sistema después de la discretización; no obstante, la aproximación trapezoidal es utilizada en los algoritmos de control de empresas reconocidas como *Siemens* o *Schneider-Electric*.

Finalmente, se realizó el análisis de la selección del periodo de muestreo que se debe utilizar en cada controlador. Si bien el periodo de muestreo para el controlador *2DoF-PID* es mayor al que necesita el controlador *1DoF-PID*, se ha optado por utilizar el mismo periodo de muestreo de $0.01s$ ambos casos; este periodo seleccionado no ocasiona mayor esfuerzo en plataformas basadas en *FPGA,* las cuales soportan frecuencias de reloj de $100MHz$ o incluso superiores. Una vez obtenido el modelo digitalizado y el periodo de muestreo seleccionado, el controlador puede desarrollarse en una plataforma electrónica digital, lo que implica pasar a la fase de modelado y generación del *Hardware* del *IP-Core*.

Modelamiento y Generación del *Hardware* del *IP-Core*

Introducción

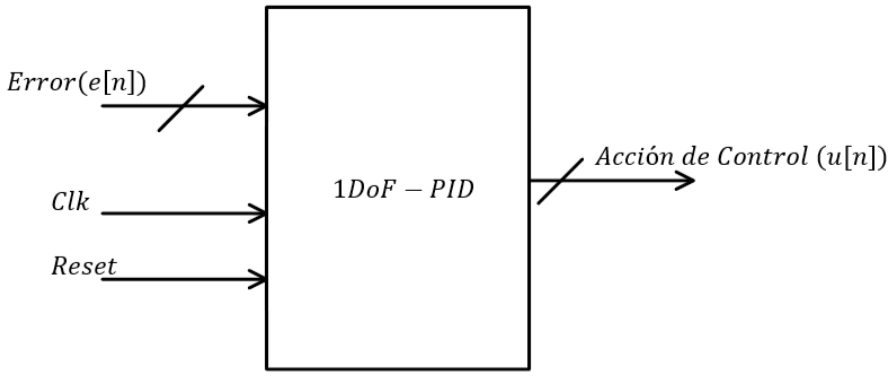
Para diseñar el *Hardware* de los *IP-Cores*, es necesario desarrollar el modelo a partir de las especificaciones descritas en uno de los lenguajes más utilizados como: C/C++, Matlab, SystemC o incluso directamente en *VHDL*, después de lo cual, hay que realizar la síntesis de alto nivel del diseño. En este sentido esta sección presenta el diseño del *Hardware* de los controladores *PID* de un grado de libertad y de dos grados de libertad respectivamente. Para la descripción del *Hardware* se ha optado por utilizar la herramienta de ingeniería Matlab, que dispone de una librería que se integra al instalar el *Software Vivado Design Suite* y *Xilinx System Generator (XSG)*, que permite utilizar el entorno *Matlab-Simulink* para el diseño e implementación del *Hardware* (XILINX, 2012). A través de este entorno Vivado-Matlab, se puede realizar la construcción de *IP-Cores* y la verificación *Hardware In The Loop*, posteriormente se puede generar los archivos necesarios como los archivos VDHL del *IP-Core*.

Diseño y Construcción del *IP-Core* Basado en ll Modelo del Controlador *PID* de un Grado de Libertad

Para la construcción del controlador *1DoF-PID* en forma de tecnología de *IP-Core*, es necesario realizar una planificación o proyección del *Hardware*. Desde el punto de vista del nivel de abstracción más alto; el *IP-Cores* puede verse como un tejido del *Hardware* que tiene la función de ser un filtro digital. La Figura 23 muestra el diagrama de entradas y salidas del *IP-Core* propuesto, donde clk es la señal de reloj, rst es la señal de reseteo, $e[n]$ es la señal de error del proceso y $u[n]$ es la acción de control. El *IP-Core* está internamente compuesto por un *Hardware* especializado para realizar la función del filtro digital *1DoF-PID*, el cual se detalla a continuación.

Figura 23

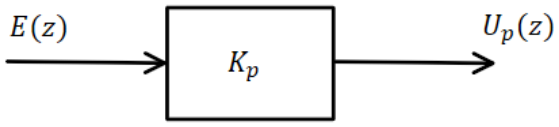
Diagrama de I/O del IP-Core 1DoF-PID.



El filtro digital desarrollado en *Hardware* debe tener la capacidad de realizar la acción proporcional, integral y derivativa. Para la construcción del filtro *1DoF-PID*, se empleará el análisis de cada una de las acciones del controlador, donde se define a la acción proporcional como u_p, a la acción integral como u_i y finalmente la acción derivativa como u_d. EL proceso de digitalización de la acción proporcional, da como resultado la misma ganancia K_p, por tanto, la acción proporcional es el resultado de la señal de error multiplicada por ganancia proporcional, observe en el diagrama de bloques presentado en la Figura 24.

Figura 24

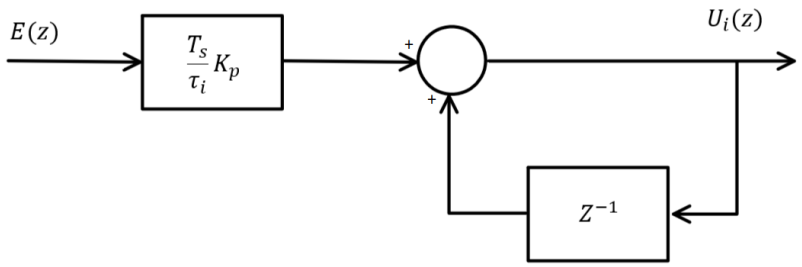
Filtro de la acción proporcional del controlador 1DoF-PID



El filtro digital de la acción integral se muestra en la Figura 25, la cual es factible de ser implementada en el *IP-Core*. Se ha adoptado por utilizar la aproximación *Euler en atraso* en el proceso de integración de la señal de error $e[n]$, pues esta aproximación puede garantizar la preservación de la estabilidad del sistema modelado de forma analógica incluso después de ser discretizado.

Figura 25

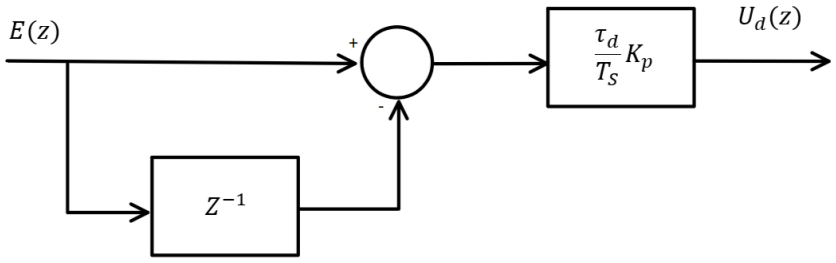
Filtro de la acción integral del controlador 1DoF-PID



Por último, el filtro de la acción derivativa es mostrado en la Figura 26, con lo cual a través de este modelado se puede implementar las tres acciones de control para sumarlas y obtener el controlador estándar *1DoF-PID* mostrada en (84). Se debe tomar en cuenta que la acción de control puede estar limitada por las capacidades físicas de la planta, es decir que para el motor modelado cuyo voltaje de entrada ideal es de 240V, tener una acción de control mayor a este voltaje, no tendría efecto o causaría daño en el sistema, por lo que la salida de control estaría limitada a estar dentro del rango permitido de voltaje.

Figura 26

Filtro de la acción derivativa del controlador 1DoF-PID



Con el análisis del *Hardware* interno del *IP-Core*, es necesario especificar el tipo de dato y resolución de las señales que se utilizarán en el proceso computacional realizado por el *IP-Core*. Para este estudio se realizará el análisis de dos escenarios, el escenario A contempla una precisión simple de punto flotante con estándar IEEE-754 de 32 bits para las operaciones internas del *IP-Core*, mientras que el escenario B contempla una precisión de punto fijo en las operaciones aritméticas; cada escenario evaluará la respuesta del sistema controlado y el costo de recursos de *Hardware* que implica. En el apartado de resultados se presentarán las ventajas y desventajas de cada escenario. La Tabla 9 muestra un resumen del escenario A, mientras que la Tabla 10 muestra el resumen del escenario B. El estándar IEEE-754 de punto flotante simple convencional, está compuesto por 8 bits que forman la parte exponencial y 24 bits para la parte fraccionaria. Para el caso de aproximación de punto fijo se ha utilizado una resolución de 32 bits, donde 22 bits están destinados para la representación decimal y el resto para la representación del número entero con signo; esta estrategia es suficiente para representar los diferentes valores de los cálculos del controlador, cuya acción de control está limitada en un valor finito entre -240 V y 240 V. El uso de 32 bits para cada escenario tiene como objetivo tener una comparación más justa, para observar el impacto en el

costo computacional de cada uno de los escenarios utilizando el mismo número de bits de resolución.

Tabla 9

Escenario con precisión punto flotante para el controlador 1DoF-PID

Entrada/Salida	Tipo de Dato	Resolución
clk	Booleano	1 bit
rst	Booleano	1 bit
$e[n]$	Real Flotante	32 bits
$u[n]$	Real Flotante	32 bits

Tabla 10

Escenario con precisión punto fijo para el controlador 1DoF-PID

Entrada/Salida	Tipo de Dato	Resolución
clk	Booleano	1 bit
rst	Booleano	1 bit
$e[n]$	Real punto fijo	32 bits
$u[n]$	Real punto fijo	32 bits

Implementación del Hardware IP-Core PID de Un Grado De Libertad

La implementación del *Hardware* del *IP-Core PID* de un grado de libertad está basada en la función presentada en (84), la acción del controlador es la suma de las acciones, proporcional, integral y derivativa construidas como un filtro digital, como se muestra en la Figura 24, Figura 25 y Figura 26 respectivamente. Existe un ligero cambio en la implementación, donde la ganancia proporcional sale como factor común. Adicionalmente, debido a las restricciones físicas del sistema, se ha desarrollado un bloque de *Hardware* adicional para saturar la señal de control en los límites de tensión máxima y mínima que son nominales para la planta. La Figura 27 muestra la implementación del controlador *1DoF-PID* realizado con recursos de *Hardware*. La Figura

28 muestra el bloque de *saturación*, el cual está basado en las especificaciones de entrada de voltaje de la planta, donde el voltaje nominal para el motor es de 240V.

Figura 27

Modelo del controlador PID de un grado de libertad en XSG

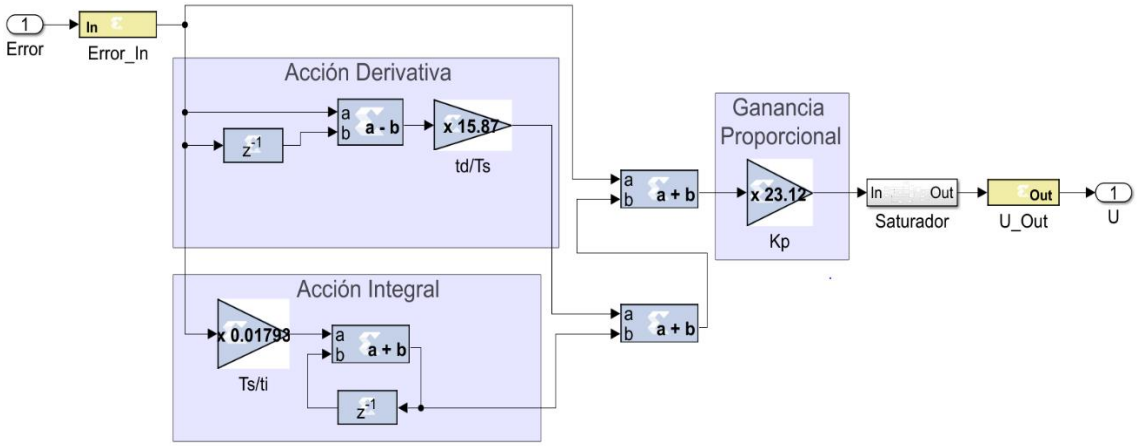
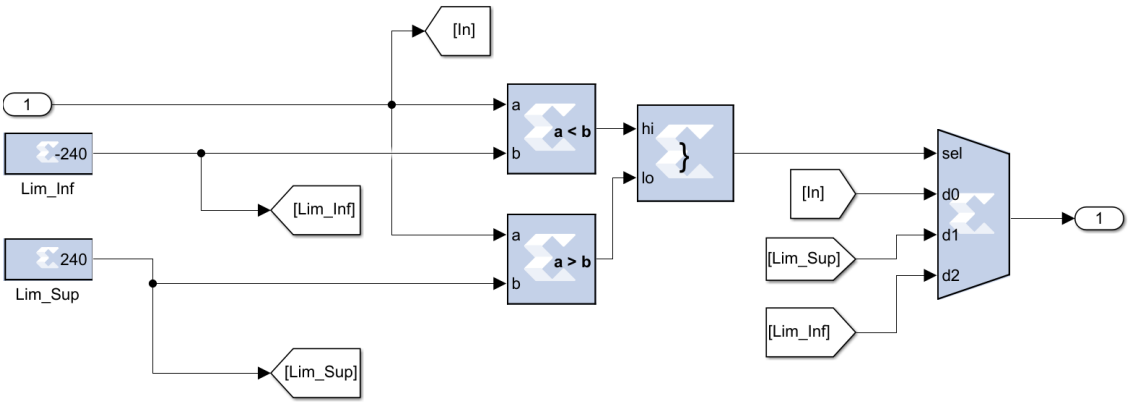


Figura 28

Modelo del bloque de saturación de la señal de control en XSG



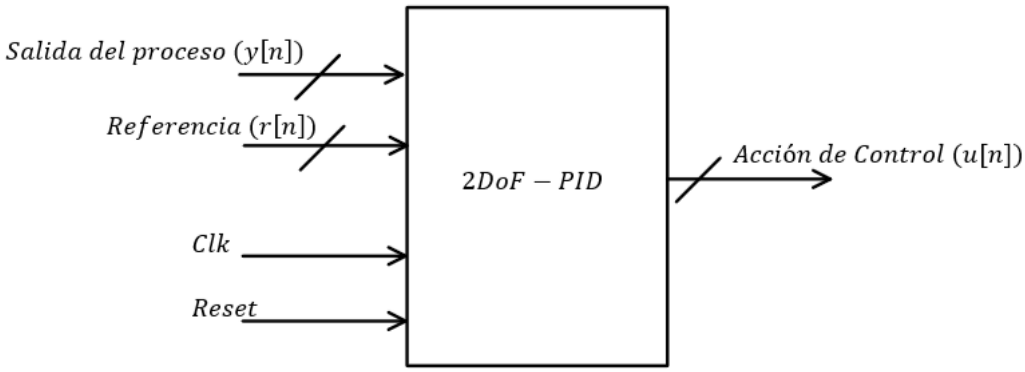
Diseño y Construcción del *IP-Core* Basado en el Modelo del Controlador *PID* de Dos Grados de Libertad

La planificación del *Hardware* del *IP-Core* se basa en el modelo del controlador *2DoF-PID* de la ecuación (94), de modo que la estructura de *Hardware* del *IP-Core* puede

cumplir la función de ser un filtro digital. La Figura 29 muestra el diagrama de entradas y salidas del *IP-Core* propuesto desde un punto de vista de alto nivel de abstracción, donde clk es la señal de reloj, rst es la señal de *reset*, $r[n]$ es la señal de referencia, $y[n]$ es la señal medida del proceso y $u[n]$ es la acción de control. El *Hardware* del *IP-Core* está compuesto por una serie de elementos especializados que realizan la función del filtro digital *2DoF-PID.*

Figura 29

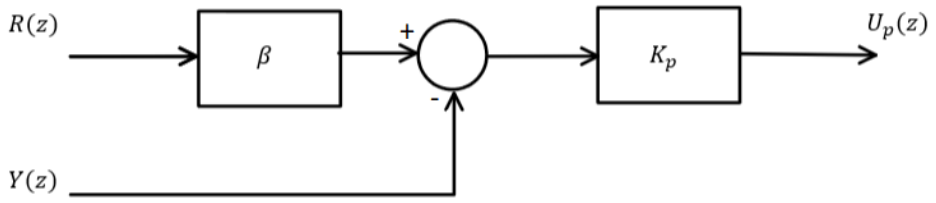
Diagrama de I/O del IP-Core 2DoF-PID



El *Hardware* del *IP-Core 2DoF-PID*, está construido con filtros digitales que realizan la acción proporcional $\boldsymbol{u_p}$, integral $\boldsymbol{u_i}$ y derivativa $\boldsymbol{u_d}$. Para el caso del filtro de acción proporcional existe una ganancia adicional que es el factor $\boldsymbol{\beta}$, el diagrama de bloques del filtro de acción proporcional para implementarse en *Hardware,* se muestra en la Figura 30.

Figura 30

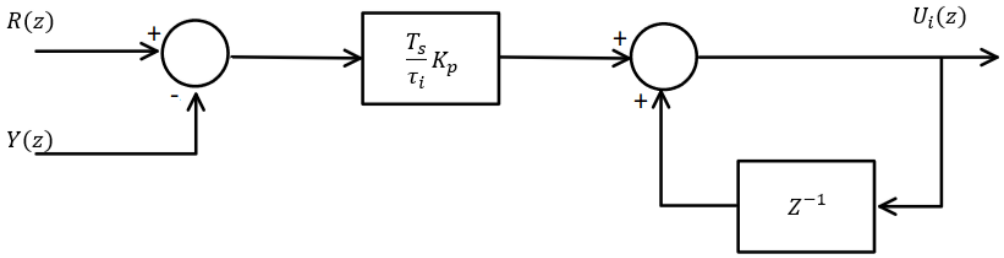
Filtro de la acción proporcional del controlador 2DoF-PID



El diagrama de bloques mostrado en la Figura 31, representa el filtro de acción integral que se implementara en el *Hardware* del *IP-Core*; para el controlador *2DoF-PID*, se ha adoptado utilizar la aproximación Euler en atraso para aproximar la acción integral por los motivos mencionados en el controlador *1DoF-PID*.

Figura 31

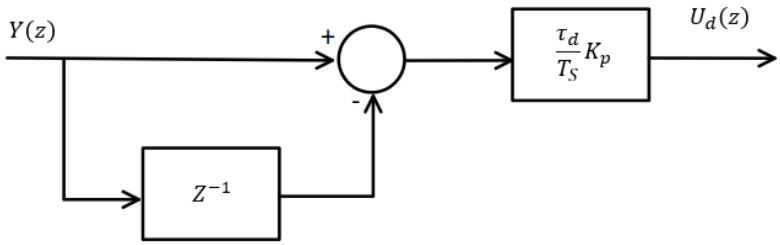
Filtro de la acción integral del controlador 2DoF-PID



Finalmente, el diagrama de bloques mostrado en la Figura 32 muestra el filtro de acción derivativo para ser implementado en el *Hardware* del *IP-Core*. Con la acción de los filtros de acción proporcional, integral y derivativa se puede formar la acción de control total del controlador *2DoF-PID* mostrada en el modelo de la ecuación (94).

Figura 32

Filtro de la acción derivativa del controlador 2DoF-PID



Con el análisis del *Hardware* interno del *IP-Core*, es necesario especificar el tipo de dato y resolución de las señales serán utilizadas en el proceso computacional realizado por el *IP-Core*. Para este estudio se realizaron los mismos escenarios planteados para el controlador *1DoF-PID.* presentados en la sección “Diseño y construcción del IP-Core basado en el modelo del controlador PID de un grado de libertad”. En el apartado de resultados se presentarán las ventajas y desventajas de cada escenario. La Tabla 11 muestra un resumen del escenario A, mientras que la Tabla 12 muestra el resumen del escenario B.

Tabla 11

Escenario con precisión punto flotante para el controlador 2DoF-PID

Entrada/Salida	Tipo de Dato	Resolución
clk	Booleano	1 bit
rst	Booleano	1 bit
$y[n]$	Real Flotante	32 bits
$r[n]$	Real Flotante	32 bits
$u[n]$	Real Flotante	32 bits

Tabla 12

Escenario con precisión punto fijo para el controlador 2DoF-PID

Entrada/Salida	Tipo de Dato	Resolución
clk	Booleano	1 bit

Entrada/Salida	Tipo de Dato	Resolución
rst	Booleano	1 bit
$y[n]$	Real punto fijo	32 bits
$r[n]$	Real punto fijo	32 bits
$u[n]$	Real punto fijo	32 bits

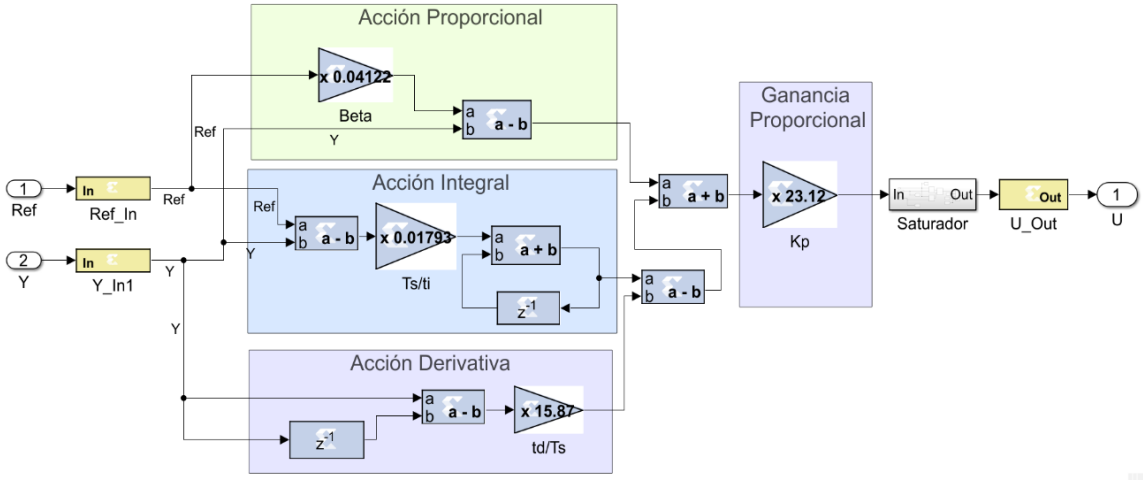
Implementación del Hardware IP-Core PID de Dos Grados de Libertad

La implementación del *Hardware* del *IP-Core PID* de dos grado**s** de libertad está basada en la función discreta presentada en (94). La acción del controlador es la consecuencia de la suma de la acción proporcional, integral y derivativa, diseñadas e implementadas como filtros digitales.

La acción proporcional, integral y derivativa está basada en las arquitecturas mostradas de las Figura 30, Figura 31 y Figura 32 respectivamente. Al igual que en caso del controlador *1DoF-PID*, se ha variado la ubicación de la ganancia proporcional al ser un factor común en las tres acciones, además se ha colocado el mismo bloque limitador presentado en la Figura 28, de modo que la acción de control permanezca dentro de los rangos en donde la señal de voltaje de entrada sea efectiva. La Figura 33 muestra la implementación del controlador *2DoF-PID* en *Hardware*, el cual integra el bloque de saturación y el filtro encargado de realizar la acción total del controlador.

Figura 33

Modelo del controlador PID de dos grados de libertad en XSG



Resumen

Este capítulo describió las arquitectura y diagrama de bloques proyectados para generar el *IP-Core*, además, se abordó la estrategia para implementar los algoritmos de control *1DoF-PID* y *2DoF-PID* digitales, planteándolos desde el punto de vista de un filtro digital, los cuales son factibles de ser implementados en *Hardware*. Una vez realizada la estrategia de discretización, se propusieron dos escenarios para evaluar la implementación del *Hardware,* para cada uno de los dos controladores modelados. Estos escenarios están directamente relacionados con la forma de implementar las operaciones *DSP* realizadas por el controlador digital; para el escenario A, las operaciones *DSP* emplean una precisión de punto flotante de 32 bits con el estándar *IEEE-754*, mientras que el escenario B se utilizó una precisión de punto fijo de 32 bits en las operaciones *DPS*. La diferencia entre estas dos reside en el algoritmo para transformar los bits a una representación numérica de tipo real, teóricamente el algoritmo con operaciones en coma flotante tiene un mayor costo computacional.

El modelamiento de *Hardware* fue realizado a través de las herramientas de *Xilinx System Generator,* esta herramienta proporciona una biblioteca para implementar el *Hardware* del *IP-Core* directamente en *Simulink.* Como resultado se pudo realizar la implementación del *Hardware* a través de bloques funcionales, que posteriormente con

la herramienta de *XSG* generó la descripción del *Hardware* del *IP-Core* en *VHDL*. Además, *XSG* permite configurar el tipo de datos que va a gestionar en el *Hardware* para realizar operaciones *DSP*, así como proporcionar las herramientas para configurar los buses de interfaz *AXI*.

Validación del Modelo y Resultados

Introducción

En este capítulo se describen los ensayos experimentales que se realizó para validar la metodología propuesta, para la implementación y validación experimental es necesario utilizar una plataforma de *Hardware* reconfigurable, la cual alberga: la unidad de procesamiento, los periféricos y el tejido de *Hardware* reconfigurable en la misma unidad. El dispositivo Basys-3 que alberga la *FPGA Artix-7* de *Xilinx* es la herramienta que permite desarrollar potentes sistemas embebidos. En la Sección Introductoria se describen las características de la plataforma Basys-3 y del *Hardware* utilizado para realizar la experimentación y validación de la metodología.

Para demostrar la aplicabilidad de la metodología en un entorno industrial, se implementó el sistema servo-controlado, donde los controladores son sintetizados en *Hardware* para ser parte de un *IP-Core*. La sección “Implementación y Pruebas Hardware In The Loop del IP-Core”, describe el proceso de implementación y simulación *Hardware In The Loop,* con el objetivo de demostrar el buen rendimiento de los controladores sintetizados por *Hardware* basados en la metodología propuesta. Para ello, se ha cuantificado el uso de los recursos *Hardware* necesarios para la implementación del *IP-Core* y el rendimiento de los controladores utilizando las métricas de un sistema de lazo cerrado.

Descripción Del *Hardware* De Ensayo

El *Hardware* de ensayo permite realizar las pruebas de síntesis y funcionamiento del *Hardware* modelado, una de las mejores opciones para este tipo de trabajos es utilizar una plataforma de *Hardware* reconfigurable basada en tecnología de *FPGA*. El presente estudio utiliza la plataforma Basys-3 (Digilent, 2016a) mostrada en la Figura 34. Es un

dispositivo fabricado por la empresa *Digilent*, él *Hardware* de la plataforma está compuesto por una serie de componentes y periféricos de entradas y salidas, la Tabla 13 muestra un resumen de los principales elementos de la plataforma reconfigurable seleccionada.

Figura 34

Plataforma Basys-3 de Digilent basada en la FPGA Artix7-35T

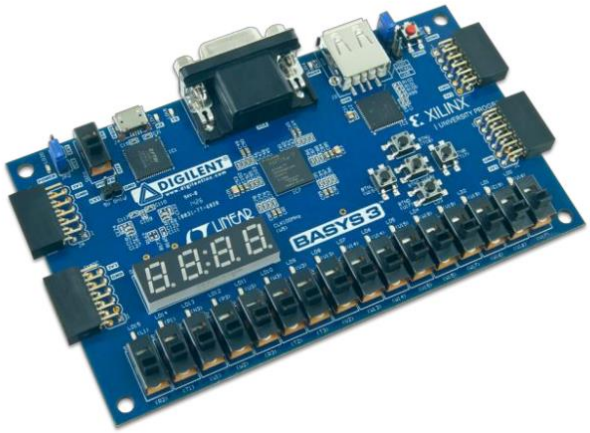


Tabla 13

Elementos y periféricos disponibles en la plataforma reconfigurable Basys-3.

Componente/Periférico	Cantidad
Interruptores	16
Displays 7 segmentos	4
Salida VGA 12 bits	1
Puerto USB-JTAG	1
Leds	16
Puente USB-UART	1
Puerto USB HID	1
Pulsadores	5
Módulo de periférico ADC	1
Reloj oscilador de 100MHz	1
FPGA	1

El componente principal a analizar en el *Hardware de la tarjeta,* es la *FPGA*, que constituye la pieza fundamental de la plataforma. La tarjeta de desarrollo Basys-3 cuenta con una *FPGA Artix-7 35T* de Xilinx, la cual está optimizada para ejecutar lógicas de alto rendimiento. La Tabla 14 muestra las principales características y recursos disponibles de la *FPGA Artix-7 35T*, los cuales serán utilizados para realizar la síntesis e implementación de los controladores *PID* diseñados.

Tabla 14

Recursos de la FPGA Artix-7 35T- 1CPG236C

Recursos	Cantidad disponible
Celdas Lógicas	33280
LUTs	20800
Bloques de RAM	1800 Kbits
registros	41600
Segmentos *DSP*	90
CLB Flip-Flops	41600
Capacidad de reloj interno	450 MHz
Conversor análogo digital On-Chip de 12 bits	1

La plataforma de desarrollo Basys-3 al estar basada en una *FPGA Artix-7* de Xilinx es compatible para funcionar con el *Software Vivado Design Suite,* el cual incluye herramientas que facilitan el flujo de diseño, permiten gestionar los recursos de las *FPGA* e incluye herramientas de síntesis de alto nivel que se complementan con el *Software* ingeniería de Matlab. Este entorno compuesto por *Software* y *Hardware*, permitirá realizar las validaciones del modelo y la metodología propuesta.

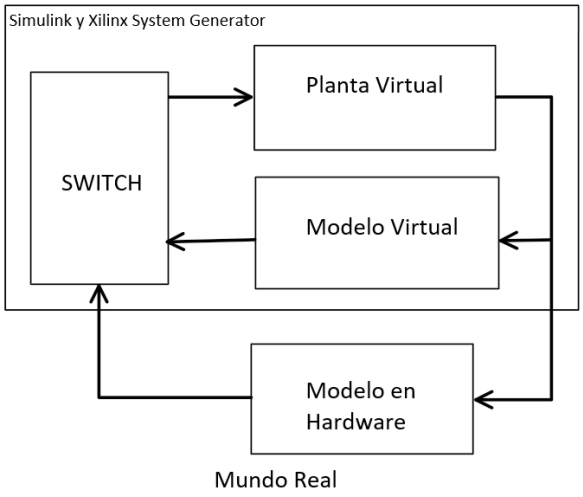
Implementación y Pruebas *Hardware* In The Loop del *IP-Core*

Mediante la metodología de pruebas *Hardware In The Loop,* se puede realizar una verificación del *Hardware* de los controladores diseñados, en dónde se pueden combinar elementos de *Software* que representan al sistema servo-motor y de *Hardware* que

contiene a los controladores. En esta sección, para la verificación e implementación de los controladores, se utilizaron los 2 escenarios mencionados en la sección *"Diseño y construcción del IP-Core basado en el modelo del controlador PID de un grado de libertad."*, en los cuales se realizó una verificación del desempeño del sistema, y muestra del costo de los recursos empleados para generar el *Hardware* de los *IP-Cores*. Para el primer escenario, el *Hardware* utiliza una precisión decimal de punto flotante para realizar las operaciones *DSP*, mientras que en el segundo escenario utiliza una precisión de punto fijo en las operaciones del procesamiento digital de señales internas del controlador. La Figura 35 muestra el esquema general propuesto para realizar la simulación *HIL,* donde se consideran todos los escenarios propuestos en *Software* y *Hardware*.

Figura 35

Arquitectura general de la simulación Hardware In The Loop



Para realizar la simulación *HIL* que se presentó en el quinto paso del flujo de la metodología mostrada en la Figura 9, es necesario haber implementado el *Hardware* en base al flujo presentado en la Figura 4; en la cual una vez introducidas las especificaciones del modelo, se debe realizar la síntesis del *Hardware*, para posteriormente proceder a realizar el *Mapping*, la implementación (*Place&Route*) y la

generación del archivo *Bit**s**tream* para ser descargado en la *FPGA*. Mediante las interfaces de *JTAG* y las herramientas de *Vivado Design Suite*, se puede realizar una comunicación del *Hardware* del *FPGA* con el ambiente del *Simulink,* en donde se puede incluir elementos virtuales, Ver Figura 36.

Figura 36

Entorno de Prueba Hardware In The Loop

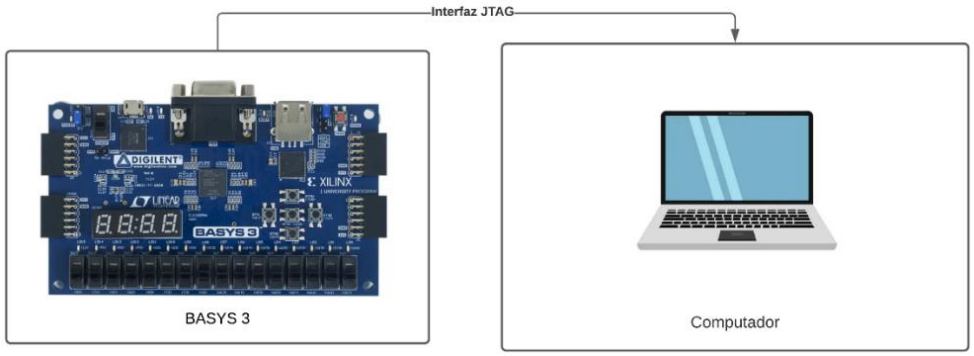
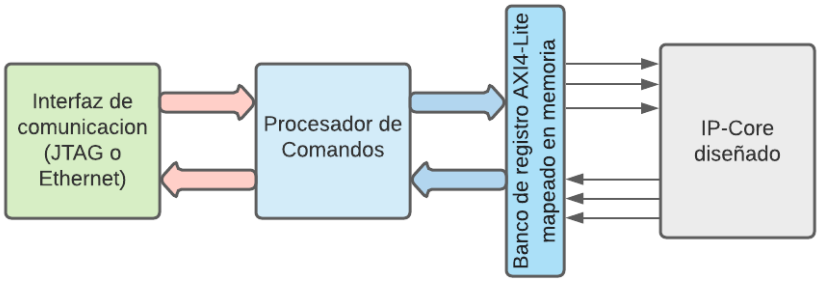


Figura 37

Mecanismo comunicación utilizado en simulación Hardware in The Loop



Nota. Adaptado de Vivado Design Suite User Guide: Model-Based *DSP* Design Using System Generator (p. 174), por XILINX, 2020.

El mecanismo que permite integrar la comunicación entre el ambiente de *Simulink y* el *IP-Core* implementado en la *FPGA* se presenta en la Figura 37, el mismo que consta de los siguientes componentes:

- **Interfaz de comunicación:** Es utilizada para realizar la comunicación entre el computador *Host* y el *IP-Core*, mediante la recepción de comandos y envío de respuestas.
- **Procesador de comandos:** Analiza y ejecuta los mensajes de comandos.
- **Banco de registro *AXI*4-Lite mapeado en memoria:** Utiliza comandos de escritura para configurar los datos de estímulo, los cuales se introducen al *Hardware* implementado. Del mismo modo se utilizan comandos de lectura para consultar las salidas del *IP-Core* mapeadas en memoria.

Resultados de Síntesis y Recursos de *Hardware* del Controlador *PID* de un Grado de Libertad

Las herramientas de *Vivado Design Suite through* reportan el análisis de tiempos y recursos de *Hardware* en la síntesis del diseño. La Tabla 15 muestra un resumen de las rutas de tiempo obtenidas al sintetizar el controlador *1DoF-PID* implementado con precisión de punto fijo y punto flotante en las operaciones del procesamiento de señales. Hay que resaltar la importancia del parámetro *Worst Slack time,* debido a que esta medida de tiempo permite identificar rá*PID*amente si existe algún path o ruta con infracciones de tiempo en la síntesis del diseño. Para que el diseño sea factible de ser implementado, el WST tiene que ser estrictamente mayor que cero; para los dos escenarios el proceso de síntesis no presentó infracciones de tiempo.

Tabla 15

Análisis de rutas de tiempo de la síntesis de Hardware del controlador 1DoF-PID

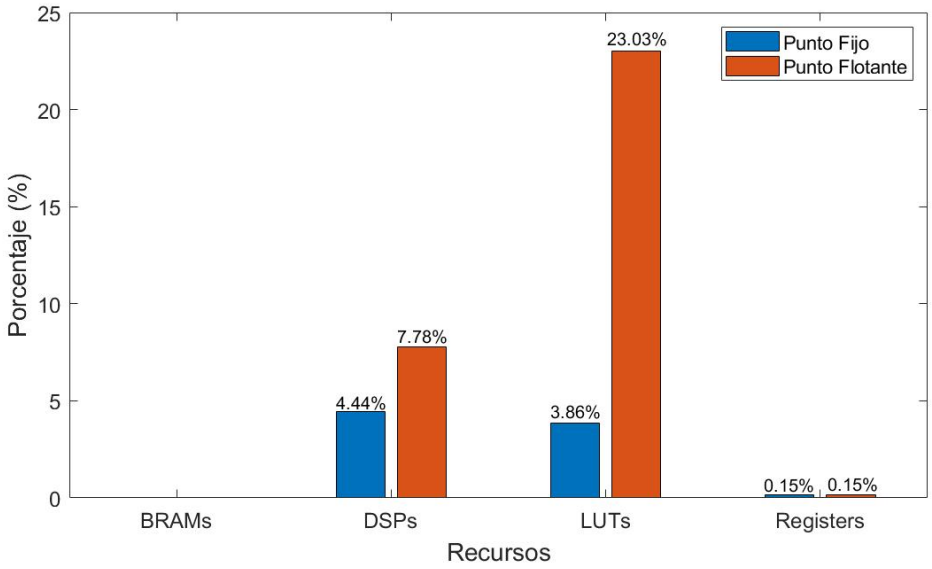
Rutas de Tiempo (Time Paths)	Precisión Punto Fijo	Precisión Punto Flotante
Worst Slack (ns)	5.561	2.872
Delay (ns)	4.307	7.073
Logic delay (ns)	2.708	4.748

Routing delay (ns)	1.599	2.325

Otro aspecto importante a la hora de desarrollar el diseño es saber cuántos recursos *Hardware* se requerirá. La Figura 38 muestra el porcentaje de recursos necesarios que deben ser utilizados para implementar el diseño del controlador *1DoF-PID* en la *FPGA Artix-35T*; como se puede observar, el uso de recursos *LUTs (Look Up Tables)* es significativamente mayor en las operaciones de precisión de punto flotante, esto se debe a que el algoritmo de cálculo para las operaciones de punto flotante es más complejo que el algoritmo que realiza las operaciones de precisión de punto fijo; lo que significa que la implementación de punto flotante requiere casi 6 veces más recursos *LUTs* que el requerido para la precisión de punto fijo.

Figura 38

Recursos de Hardware utilizados para implementar controlador 1DoF-PID utilizando precisión de punto flotante y de punto fijo



Resultados de Síntesis y Recursos de *Hardware* del Controlador *PID* de Dos Grados de Libertad

La Tabla 16 muestra el resumen de análisis de rutas de tiempos obtenidos después de la síntesis del diseño del controlador *2DoF-PID,* implementada con precisión de punto fijo y punto flotante en las operaciones del procesamiento de señales. El parámetro del *Worst Slack time,* resulta ser positivo en cada una de las implementaciones propuestas, como resultado no se presentó ninguna infracción en la síntesis y no tiene restricciones para ser descargados en la *FPGA*.

Tabla 16

Análisis de rutas de tiempo de la síntesis de *Hardware* del controlador *2DoF-PID*

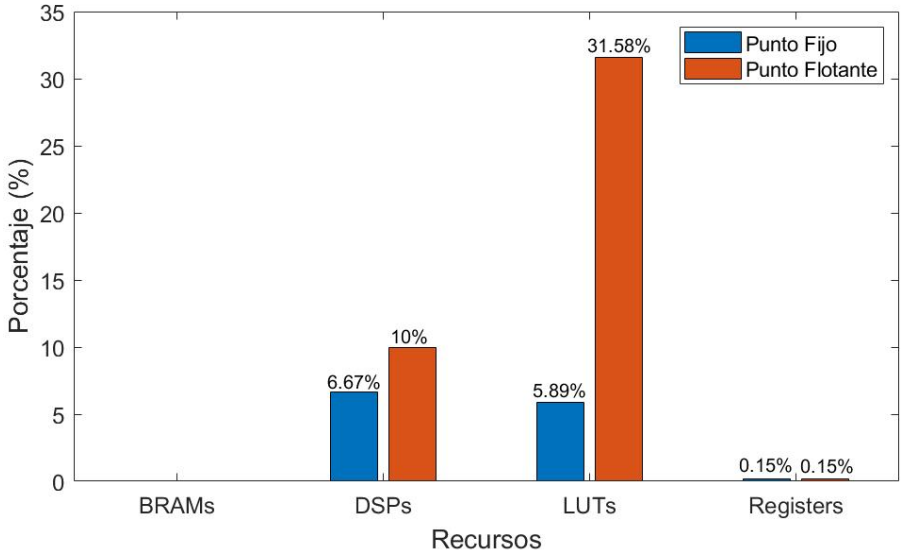
Rutas de Tiempo (Time Paths)	Precisión Punto Fijo	Precisión Punto Flotante
Worst Slack (ns)	5.514	5.561
Delay (ns)	4.431	4.307
Logic delay (ns)	2.832	2.708
Routing delay (ns)	1.599	1.599

Al igual que ocurre con el caso del controlador *1DoF-PID*, el uso de recursos de *Hardware* en el diseño del compensador *2DoF-PID* es significativamente mayor utilizando una precisión de punto flotante. La

Figura 39 muestra el porcentaje de uso de recursos empleados para implementar el controlador *2DoF-PID* en una *FPGA Artix 35T*, utilizando precisión de punto fijo y punto flotante en las operaciones *DSP*. Se puede observar cómo hay un mayor impacto en el uso de recursos de *LUTs* utilizados en la precisión de punto flotante, que llegan a utilizar casi 5 veces más recursos de *LUTs* que la precisión de punto fijo, por las mismas razones mencionadas anteriormente para el controlador *1DoF-PID*.

Figura 39

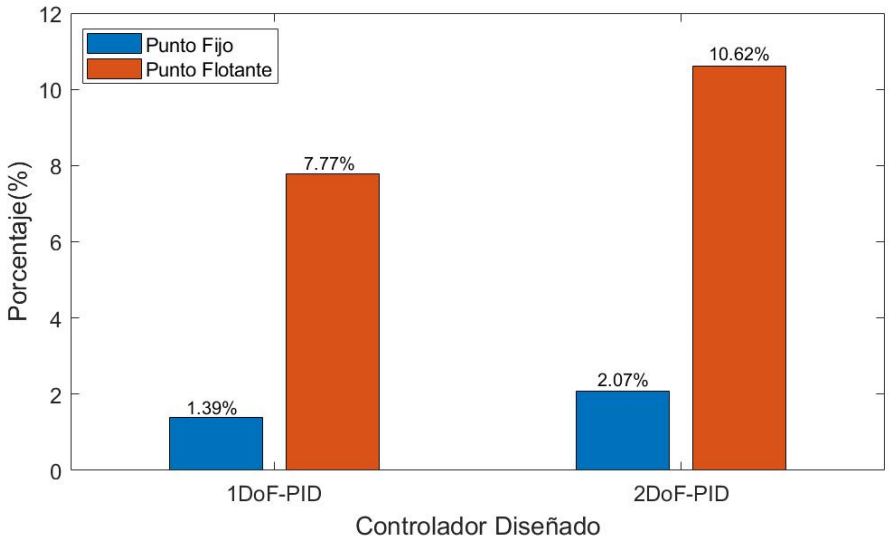
Recursos de Hardware utilizados para implementar controlador 1DoF-PID utilizando precisión de punto flotante y de punto fijo



A manera análisis, la Figura 40 muestra el porcentaje de *Hardware* que se debe utilizar de la *FPGA Artix 35T*, para la implementación de los diseños de controlador *PID* de uno y dos grados de libertad, utilizando precisión de punto fijo o punto flotante en las operaciones *DSP*. Como se puede apreciar, existe una gran diferencia en el uso de recursos de *Hardware* que se necesita para implementar los controladores, al utilizar una precisión de punto flotante y una precisión de punto fijo. El uso de la precisión de coma flotante tiene un gran impacto en el uso de recursos de *Hardware* del *IP-Core*, esto se debe tenerse en cuenta, ya que el diseñador de un sistema *SoC* necesitará recursos de *Hardware* para integrar otros tipos de *IP-Cores*, como procesadores, módulos de memoria, módulos periféricos y otros. En el caso de diseños de *SoC* complejos, en los que el uso de recursos *Hardware* es bajo, se puede desarrollar el *Hardware* utilizando una buena aproximación de punto fijo, por lo que cada *IP-Cores* no superará el 2,07% de los recursos de la *FPGA* utilizada en la experimentación para el caso del controlador *2DoF-PID*, para el caso de la implementación del controlador de un grado de libertad el porcentaje de recursos disminuye; por otro lado, si el sistema requiere un *IP-Cores* que manipule los datos con precisión de punto flotante, se debe tener en cuenta que los recursos necesarios pueden alcanzar el 10,62% en el diseño de controlador *2DoF-PID.*

Figura 40

Recursos de Hardware ocupados en cada diseño de controlador con precisión de punto fijo y punto flotante.



Desempeño Del Controlador *1dof-PID* Implementado En *Hardware*

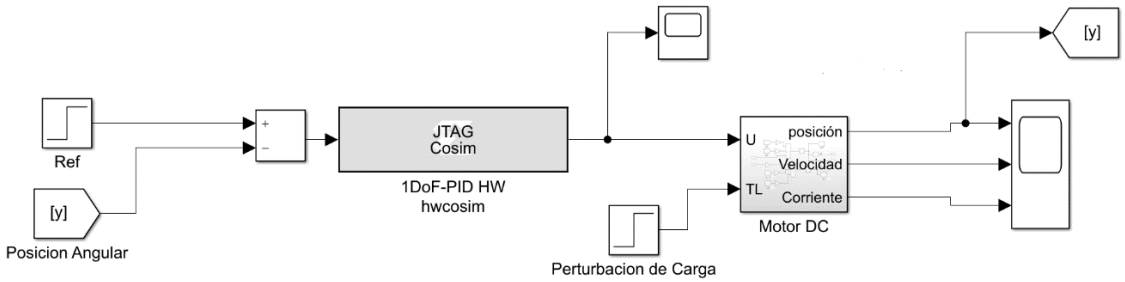
La Figura 41 muestra la arquitectura de la simulación *HIL* del controlador *1DoF-PID* implementada e integrada con el ambiente de *Simulink*. La arquitectura está compuesta por el bloque *1DoF-PID HW*, este bloque permite que haya una transferencia de datos a través del cable de comunicación *JTAG*, entre el *Hardware* del *IP-Core* implementado en la *FPGA* y el bloque del modelo del motor DC, construido en *Simulink,* a partir del espacio de estados presentado en (8). Para el estudio del desempeño del modelo de *Hardware* del controlador *PID* de un grado de libertad, se analizó el comportamiento del sistema retroalimentado bajo ciertas pruebas, con el objetivo de evaluar la respuesta del sistema ante una entrada escalón unitario, el seguimiento a la referencia y el rechazo a las perturbaciones de carga. Cada prueba se las realizó bajo los escenarios en donde las operaciones *DSP* utilizan precisión de punto fijo y punto flotante, adicionalmente el *Hardware* implementado dispone de un bloque de saturación de la

señal de control. Para comparar el desempeño del controlador construido en *Hardware* se realizó la experimentación y comparación entre los siguientes cuatro casos de estudio:

1. El primer caso está compuesto por un escenario totalmente basado en *Software*, en el que tanto el controlador como la planta, se desarrollaron con recursos netamente de *Software* y sin restricción en la salida de control.
2. El segundo caso está compuesto por un controlador digital implementado en *Software,* con un bloque de saturación en la salida del compensador.
3. El tercer caso incluye el controlador implementado en *Hardware,* con una precisión de punto fijo y con un bloque de saturación en la señal de control.
4. El último caso corresponde al sistema controlado con el compensador construido en *Hardware,* con una precisión de punto flotante para las operaciones internas del controlador y cuenta con un bloque que limita la señal de salida máxima de control.

Figura 41

Simulación Hardware In The Loop (HIL) del controlador 1DoF-PID



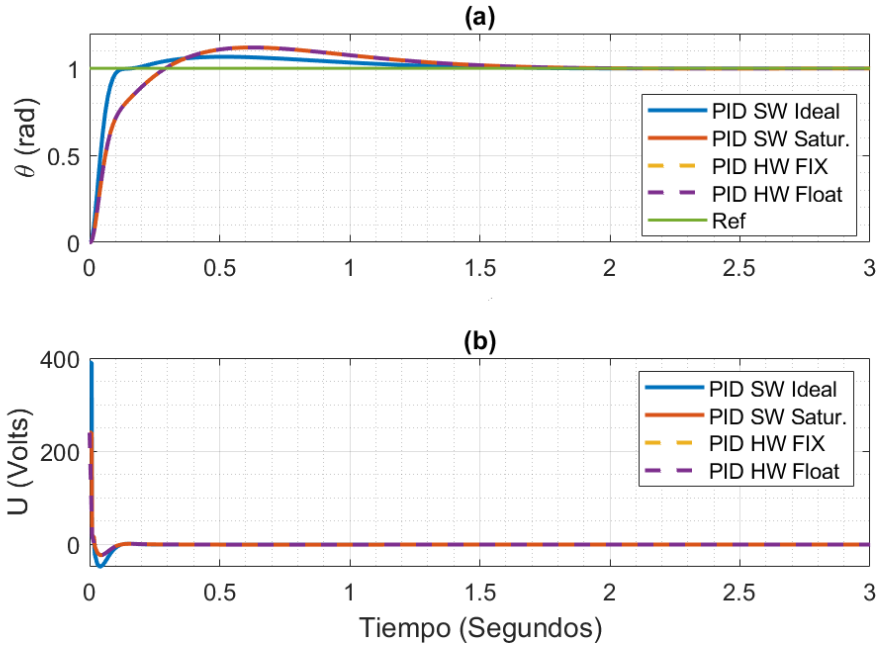
Desempeño del Sistema Ante Una Entrada de Tipo Escalón Unitario

La Figura 42 muestra la respuesta ante una entrada escalón unitario en el sistema servo-controlado implementado con 4 estrategias. La primera implementación se realizó

completamente en *Software* y sin el bloque de saturación en la señal de control *(PID SW Ideal)*, la segunda implementación se realizó también en *Software*, pero se colocó el bloque de saturación en la señal de control *(PID SW Satur)*, la tercera implementación consiste en la integración por *Hardware* del controlador, haciendo uso la precisión de punto fijo en las operaciones *DSP* e incorporando el bloque de saturación en la señal de control *(PID HW FIX)*, finalmente se implementó el controlador *PID* de un grado de libertad en *Hardware*, utilizando la estrategia de punto flotante y con el bloque de saturación a la salida del controlador *(PID HW FLOAT)*. Como se puede observar, la respuesta del sistema con el compensador desarrollado en *Software*, sin el bloque de saturación en la señal de control, presenta una dinámica diferente a las demás implementaciones que incorporan el bloque de saturación en la señal de control. Al observar la Figura 42(a) se puede evidenciar de manera clara, como las implementaciones que poseen el bloque de saturación a la salida del controlador poseen un incremento del $\%MP$ y del t_{ss}. La Tabla 17 muestra un resumen del desempeño de cada controlador *1DoF-PID* implementado, la principal diferencia entre el controlador sin limitador y los demás compensadores con el bloque de saturación, se encuentra en la salida de la señal de control u(t), cuyo límite máximo establecido alcanza el voltaje nominal del motor de $240\,V$. Si observamos la Figura 42(b) se puede apreciar como la acción de control del compensador sin bloque de saturación llega a ser aproximadamente $400\,V$, mientras que, para el resto de los casos, la acción de control se limita a los $240\,V$. La restricción de la salida de control provoca un aumento del sobre impulso en la dinámica del sistema retroalimentado, por lo que es muy importante implementar este tipo de restricciones, debido a que la salida de control limitada, impide la existencia de tensiones superiores a $240\,V$, porque esto tendría un efecto perjudicial para la planta o en su defecto es imposible de ser implementada.

Figura 42

Respuesta al escalón unitario del sistema controlador con el compensador 1DoF-PID



Nota. (a) Respuesta del sistema ante entrada escalón unitario. (b) Respuesta de la acción de control.

Entonces se puede inferir que el bloque de saturación tiene un efecto importante en el comportamiento del sistema, esto se debe a que, cuando los sensores y actuadores se saturan, la accesibilidad de la planta se reduce efectivamente (Bernstein, n.d.). Esto origina el fenómeno *Windup*, el cual es un problema que aparece por la acumulación de error en la acción integral (da Silva et al., 2018); como resultado los controladores con el bloque de saturación pasan de tener un $\%MP$ teórico de 6.65% a un valor aproximado del 12%. Además, hay un aumento en el tiempo de establecimiento que pasa a ser de $1.46\,s$, no obstante, este parámetro se encuentra dentro del límite máximo del diseño de $1{,}5s$; dejando atrás este inconveniente, cabe destacar que la implementación del controlador *1DoF-PID* en *Hardware* fue satisfactoria, ya que el rendimiento producido por el

Hardware es similar al rendimiento del controlador implementado en *Software* con la misma restricción.

Tabla 17

Desempeño del controlador 1DoF-PID ante una entrada escalón unitario

Tiempo de Establecimiento $(\boldsymbol{seg})$	**Tiempo de Subida** $(\boldsymbol{seg})$	**Sobre Impulso** $(\%)$	**Error de estado estable** $(\boldsymbol{rad})$	**Esfuerzo de control** $(\boldsymbol{v})$
PID digital SW sin restricciones				
1.19	0.06	6.65	2.63e-12	489.86
PID digital SW con restricciones				
1.46	0.18	12.0005	3.04e-12	289.0524
PID Hardware precisión punto fijo				
1.46	0.18	12.0001	4.97e-6	289.0648
PID Hardware precisión punto flotante				
1.46	0.18	12.0007	6.91e-6	289.0520

Desempeño del Sistema Ante Seguimiento de Referencia.

El comportamiento que posee un sistema ante un seguimiento a la referencia determina un aspecto importante a la hora de evaluar un controlador, pues el controlador debe ser capaz de presentar un buen seguimiento de referencia, al menos en un rango establecido; para este caso se ha limitado al giro del rotor en un rango de $[\mathbf{0}\ \mathbf{2\pi}]\ \boldsymbol{rad}$, puesto que, para la mayoría de las aplicaciones de control de articulaciones de brazos robóticos, este es el rango máximo efectivo. La Figura 43 muestra la respuesta del sistema ante cambios a la entrada de referencia que ocurren en tres intervalos de tiempo y con los mismos escenarios expuestos en esta sección:

1. ***Intervalo 0 a 3 segundos:*** Para este intervalo el comportamiento de cada controlador es el mismo que el observado en el análisis de la respuesta al escalón unitario, ya que en este intervalo la referencia es una señal escalón unitario, como

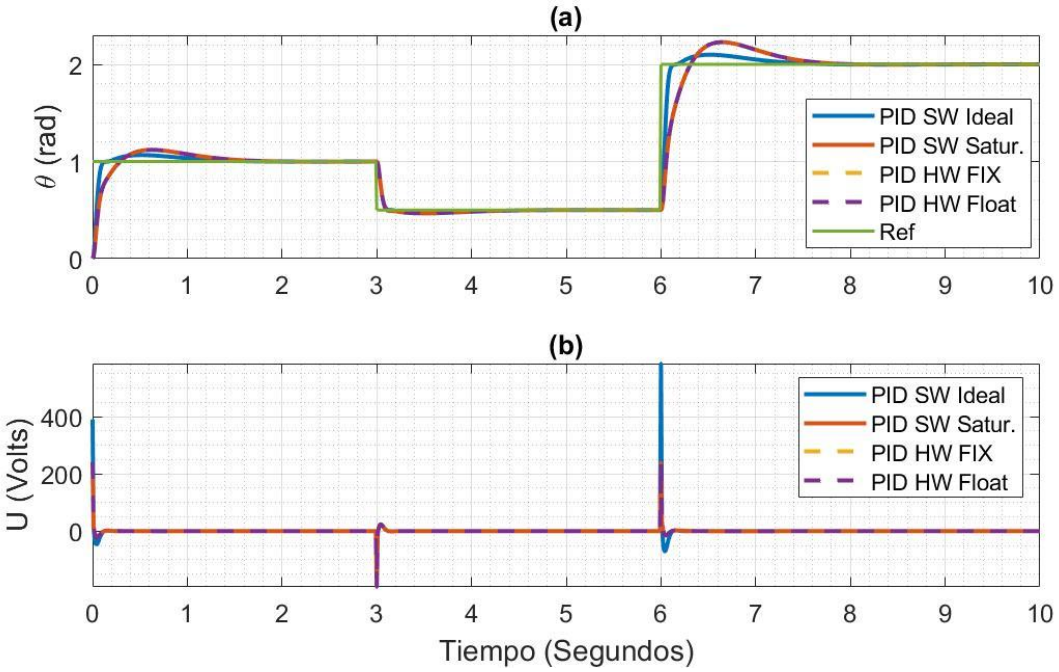
consecuencia el sistema presenta el mismo comportamiento analizado anteriormente.

2. ***Intervalo de 3 a 6 segundos:*** Para este intervalo la referencia pasó de $1\ rad$ a $0.5\ rad$, cómo se puede observar en la Figura 43(b), la acción de control no se satura, por ende, no aparece el efecto *Windup*, por lo que el $\%MP$ se mantiene cerca del valor teórico al igual que el tiempo de establecimiento.
3. ***Intervalo de 6 a 10 segundos:*** Para el último intervalo, la referencia deseada pasa a ser $2\ rad$, el cual resulta ser un salto significativo, para el controlador *PID* sin restricción en la salida de control; existe una señal $u(t)$ que supera los $500\ V$, mientras que en los demás controladores la señal máxima de control sigue siendo 240V, esto ocasiona que aparezca de nuevo el fenómeno de *Windup*, que incrementa $\%MP$ aproximado de $11.45\ \%$.

La Tabla 18 muestra un resumen del comportamiento de los controladores implementados en los tres intervalos de tiempos mencionados. Se puede destacar que en el régimen de estado permanente del sistema, el tiempo de establecimiento y el error de estado estable se encuentran dentro de los rangos de diseño deseados, siendo e_{ss} prácticamente nulo y el tiempo de establecimiento menor a $1.5\ s$.

Figura 43

Respuesta del sistema ante cambios de referencia deseada con el controlador 1DoF-PID



Nota. (a) Respuesta del sistema ante cambios de referencia. (b) Respuesta de la acción de control.

Tabla 18

Desempeño del sistema ante cambios de referencia deseada con el controlador 1DoF-PID

Intervalo de tiempo $(\boldsymbol{seg})$	**Referencia** $(\boldsymbol{rad})$	**Tiempo de Establecimiento** $(\boldsymbol{seg})$	**Sobre Impulso** (%)	**Valor final alcanzado** $(\boldsymbol{rad})$	**Error de estado estable** $(\boldsymbol{rad})$
		PID digital SW sin restricciones			
0 a 3	1	1.19	6.65	0.9998	-1.8779e-4
3 a 6	0.5	1.18	6.652	0.5001	9.3826e-5
6 a 10	2	1.08	4.9871	2.0000	1.1586e-5
		PID digital SW con restricciones			
0 a 3	1	1.46	12.005	0.9996	-3.9667e-4

Intervalo de tiempo $(\boldsymbol{seg})$	**Referencia** $(\boldsymbol{rad})$	**Tiempo de Establecimiento** $(\boldsymbol{seg})$	**Sobre Impulso** $(\%)$	**Valor final alcanzado** $(\boldsymbol{rad})$	**Error de estado estable** $(\boldsymbol{rad})$
3 a 6	0.5	1.13	6.4811	0.5001	6.8925e-5
6 a 10	2	1.47	11.4532	2.0000	1.1891e-5
		PID Hardware precisión punto fijo			
0 a 3	1	1.46	12.001	0.9996	-4.0423e-4
3 a 6	0.5	1.13	6.4821	0.5001	6.3537e-5
6 a 10	2	1.47	11.4529	2.0000	
		PID Hardware precisión punto flotante			
0 a 3	1	1.46	12.007	0.9996	-3.9961e-4
3 a 6	0.5	1.13	6.4818	0.5001	7.2332e-5
6 a 10	2	1.47	11.4534	2.0000	1.1219e-5

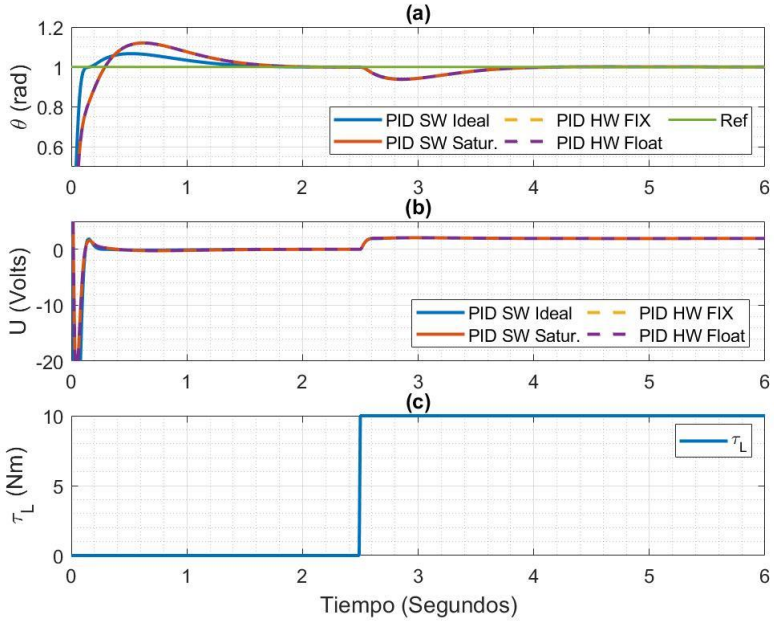
Desempeño del Sistema Ante Perturbaciones de Carga

Hay que tener en cuenta que el sistema se modeló en un entorno de ideal, donde el torque disipado de la carga τ_L era igual a cero, para esta sección se simuló una perturbación significativa en la carga de $10\ Nm$, la cual representa un 36.5% del torque máximo del motor en vacío y se observó el comportamiento ante esta perturbación. La Figura 44 muestra la perturbación introducida a los $2.5\ s$; el comportamiento del sistema para cada uno de los escenarios es muy similar, el tiempo de recuperación para que el sistema vuelva a su objetivo es aproximadamente $1.1\ s$, el esfuerzo de control aumenta aproximadamente $2\ V$ para todos los controladores implementados. El aumento del esfuerzo de control se debe a que el compensador incrementa en $2\ V$ la acción de control, para rechazar la perturbación de carga introducida. La Tabla 19 resume el comportamiento del sistema ante la perturbación de carga para cada uno de los controladores implementados; una vez que el sistema ha rechazado completamente la perturbación producida por el torque de carga introducido, el compensador desarrollado por el *Hardware* presenta un error en estado estacionario prácticamente nulo. Aunque el

e_{ss} producido por los controladores implementados por *Hardware* es mayor que el de las implementaciones de *Software*, en la práctica el error resulta ser tan insignificante que no supone una diferencia notable en la aplicación real.

Figura 44

Respuesta del sistema ante perturbación de carga con el controlador 1DoF-PID



Nota. (a) Respuesta del sistema ante perturbación en la carga. (b) Respuesta de la acción de control. (c) Perturbación de carga introducida en el sistema.

Tabla 19

Desempeño del Sistema ante perturbación de carga con el controlador 1DoF-PID.

Inicio de perturbación (seg)	**Perturbación de carga** (Nm)	**Esfuerzo de control sin perturbación** (V)	**Esfuerzo de control con perturbación** $(\%)$	**Tiempo de recuperación** (s)	e_{ss} (rad)
		PID digital SW sin restricciones			
2.5	10	489.86	492.0597		7.8902e-10

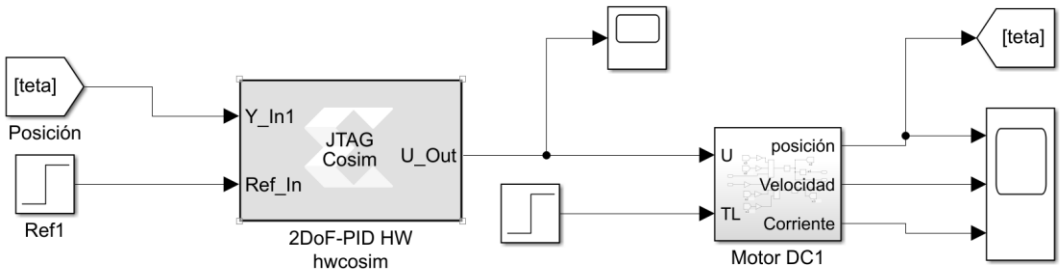
Inicio de perturbación (seg)	**Perturbación de carga** (Nm)	**Esfuerzo de control sin perturbación** (V)	**Esfuerzo de control con perturbación** (%)	**Tiempo de recuperación** (s)	e_{ss} (rad)
		PID digital SW con restricciones			
2.5	10	289.0524	291.2437	1.11	3.5533e-10
		PID Hardware precisión punto fijo			
2.5	10	289.0648	291.2607	1.11	1.2218e-5
		PID Hardware precisión punto flotante			
2.5	10	289.0520	291.2434	1.11	2.8365e-6

Desempeño Del Controlador *2dof-PID* Implementado En *Hardware*

Para evaluar el desempeño del controlador *2DoF-PID* en *Hardware,* se empleó las mismas estrategias de aproximación en las operaciones *DSP utilizadas en el compensador 1DoF-PID*. La Figura 45 muestra la implementación de la simulación *HIL* en *Simulink*, donde el bloque *2DoF-PID HW* representa el *Hardware* del *IP-Core* transferido a la *FPGA,* el cual se comunica con el ambiente de *Simulink* a través de la interfaz *JTAG*. El modelo de la planta se encuentra implementado en el bloque *Motor DC1* y está gobernado por el controlador *2DoF-PID* en *Hardware*. Al igual que en el caso del controlador *1DoF-PID*, se analizará el comportamiento del sistema gobernado por el controlador de dos grados de libertad, bajo las mismas pruebas de desempeño como son la respuesta al escalón unitario, seguimiento a la referencia y rechazo a las perturbaciones en la carga; en donde se comparará el desempeño del *Hardware* frente a los casos de implementación por *Software*.

Figura 45

Simulación Hardware In The Loop (HIL) del controlador 2DoF-PID



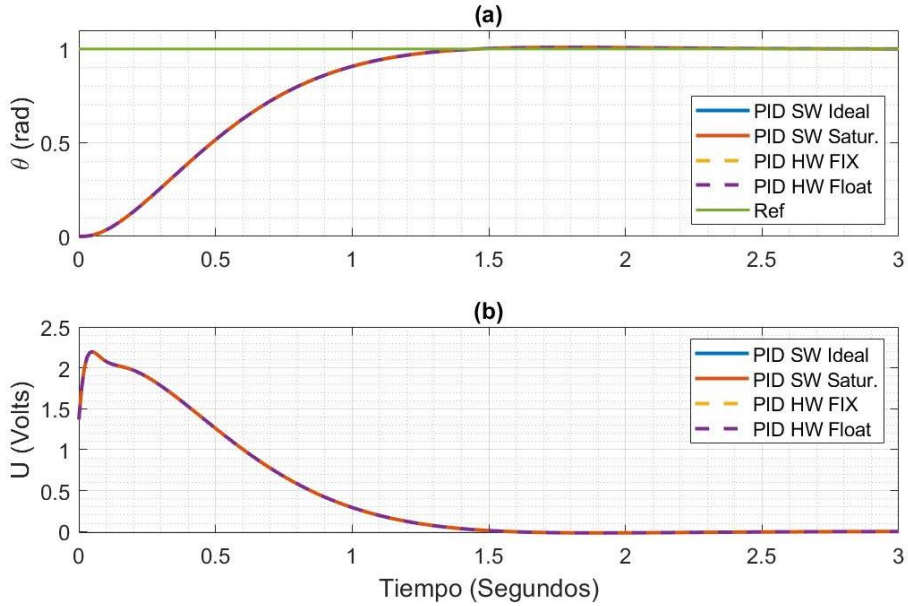
Desempeño del Sistema Ante Una Entrada de Tipo Escalón Unitario

La Figura 46 muestra el comportamiento dinámico del sistema alcanzado ante una entrada de escalón unitario; se puede apreciar, cómo el comportamiento del sistema con el controlador 2DoF-PID, posee una respuesta dinámica más suave que la obtenida con el controlador 1DoF-PID. La Tabla 20 muestra un resumen de los parámetros que determinan el desempeño del controlador en el sistema; donde el controlador construido en Hardware con precisión de punto flotante y punto fijo, tiene aproximadamente el mismo comportamiento que el desarrollado en Software con y sin restricción a la salida del controlador. Se obtuvo como resultado en todos los casos un %MP que no supera 1%, con un tiempo de establecimiento menor a 1.5 s; y con un error de estado estable despreciable en la práctica. Si observamos Figura 46(b), la acción de control del compensador de dos grados de libertad, resulta ser una curva mucho más suave en comparación a la acción de control del controlador 1DoF-PID y por lo tanto la señal u(t) no llega a saturarse, concluyéndose, que el controlador 2DoF-PID no presenta el efecto de windup en la acción integral, por lo que la implementación de este controlador produce una respuesta transitoria más cercana a la del diseño deseado. Hay que destacar que el

esfuerzo de control en el PID de dos grados de libertad implementado con el bloque de saturación, es mucho menor que el obtenido con el controlador 1DoF-PID implementado en iguales condiciones. Existe una reducción del esfuerzo de control aproximada en un 98.9%, como resultado de esta reducción se produce una menor fatiga sobre los actuadores en el sistema.

Figura 46

Respuesta al escalón unitario del sistema controlador con el compensador 2DoF-PID



Nota. (a) Respuesta del sistema ante entrada escalón unitario. (b) Respuesta de la acción de control.

Tabla 20

Desempeño del controlador 2DoF-PID ante una entrada escalón unitario

Tiempo de Establecimiento (seg)	**Tiempo de Subida** (seg)	**Sobre Impulso** (%)	**Error de estado estable** (rad)	**Esfuerzo de control** (V)
PID digital SW sin restricciones				
1.28	0.81	0.8727	2.9672e-12	3.0623

Tiempo de Establecimiento $(\boldsymbol{seg})$	**Tiempo de Subida** $(\boldsymbol{seg})$	**Sobre Impulso** $(\%)$	**Error de estado estable** $(\boldsymbol{rad})$	**Esfuerzo de control** $(\boldsymbol{V})$
PID digital SW con restricciones				
1.28	0.81	0.8727	2.9272e-12	3.0623
PID Hardware precisión punto fijo				
1.28	0.81	0.8727	3.2179e-7	3.0726
PID Hardware precisión punto flotante				
1.28	0.81	0.8723	5.3148e-6	3.0652

Desempeño del Sistema ante Seguimiento de Referencia

Como se puede observar en el resumen presentado en la Tabla 21, el comportamiento del sistema resulta ser simétrico en términos prácticos, tanto para un aumento de la referencia en los intervalos de $0\ s$ a $1\ s$ y $de\ 6\ s$ a $10\ s$, como para una disminución de la misma en el intervalo de $3\ s$ a $6\ s$. Explicado porque en ninguno de los escenarios analizados se produce una saturación de la señal de control, por lo tanto, se evita producir un comportamiento asimétrico en la salida del sistema controlado frente a cambios de aumento o disminución en la señal de referencia. Debido al comportamiento suave de la señal de control y al rango máximo de salida deseado de 2π rad, no se podría producir un comportamiento donde la señal de control presente saturación, evitando así el fenómeno del Windup y una posible asimetría en el comportamiento del sistema ante cambios de referencia. La Figura 47 muestra la experimentación del sistema gobernado por el controlador *2DoF-PID,* el cual fue implementado en *Hardware* y *Software*. La Tabla 21 presenta un resumen de la experimentación, en la cual se puede apreciar que se ha realizado un cambio de consigna para 3 intervalos de tiempo distintos:

1. ***Intervalo 0 a 3 segundos****:* Para este intervalo el comportamiento de cada controlador es el mismo que el observado en el análisis de la respuesta ante escalón unitario del controlador *2DoF-PID*, ya que en este intervalo la referencia

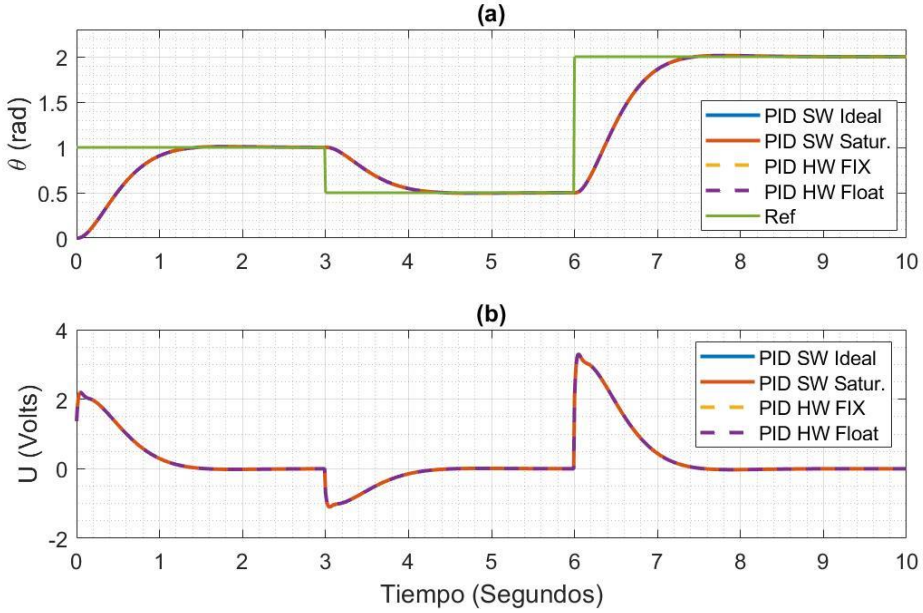
es una señal escalón unitario, como consecuencia tiene el mismo comportamiento analizado anteriormente.

2. ***Intervalo de 3 a 6 segundos:*** Para este intervalo la referencia pasó de $1\ \mathrm{rad}$ a $0.5\ \mathrm{rad}$, cómo se puede observar en el resumen de la Tabla 21, el comportamiento del sistema resulta ser simétrico en términos prácticos, tanto para un aumento de la referencia en los intervalos de $0\,s$ a $1\,s$ y de $6\,s$ a $10\,s$, como para una disminución de la misma en el intervalo de $3\,s$ a $6\,s$. En ninguno de los escenarios analizados se produce una saturación de la señal de control, por lo tanto, se evita producir un comportamiento asimétrico en la salida del sistema controlado frente a cambios de aumento o disminución en la señal de referencia. Debido al comportamiento suave de la señal de control y al rango máximo de salida deseado de 2π rad, no se podría producir un comportamiento donde la señal de control presente saturación, evitando así el fenómeno del *Windup* y una posible asimetría en el comportamiento del sistema ante cambios de referencia. La Figura 47(b), muestra como la acción de control es una señal suave, sin saturación, por lo que el efecto *Windup* no existe; como resultado $\%MP$ se mantiene cerca del valor teórico al igual que el tiempo de establecimiento. Él $\%MP$ se mantiene menor al 1% y con error de estado estable despreciable.
3. ***Intervalo de 6 a 10 segundos:*** Para el último intervalo, la referencia deseada pasa a ser de $2\,rad$, a pesar de ser un salto de consigna significante, el controlador *PID* de dos grados de libertad presenta el comportamiento esperado. La señal de control sigue siendo una señal suave y sin saturación, por tanto, no hay presencia del fenómeno *Windup*, a la vez que los demás parámetros como el $\%MP$ es inferior al 1%, el tiempo de asentamiento inferior a $1{,}5\,s$ y el error de estado estacionario casi nulo.

Como se puede observar en el resumen de la Tabla 21, el comportamiento del sistema resulta ser simétrico en términos prácticos, tanto para un aumento de la referencia en los intervalos de $0\ s\ a\ 1\ s$ y de $6\ s\ a\ 10\ s$, como para una disminución de la misma en el intervalo de $3\ s\ a\ 6\ s$. Explicado porque en ninguno de los escenarios analizados se produce una saturación de la señal de control, por lo tanto, se evita producir un comportamiento asimétrico en la salida del sistema controlado frente a cambios de aumento o disminución en la señal de referencia. Debido al comportamiento suave de la señal de control y al rango máximo de salida deseado de $2\pi\ rad$, no se podría producir un comportamiento donde la señal de control presente saturación, evitando así el fenómeno del *Windup* y una posible asimetría en el comportamiento del sistema ante cambios de referencia, ya sea de subida o de bajada.

Figura 47

Respuesta del sistema ante cambios de referencia deseada con el controlador 2DoF-PID



Nota: (a) Respuesta del sistema ante cambios de referencia. (b) Respuesta de la acción de control.

Tabla 21

Desempeño del sistema ante cambios de referencia deseada con el controlador 2DoF-PID

Intervalo de tiempo $(\boldsymbol{seg})$	**Referencia** $(\boldsymbol{rad})$	**Tiempo de Establecimiento** $(\boldsymbol{seg})$	**Sobre Impulso** (%)	**Valor final alcanzado** $(\boldsymbol{rad})$	$\boldsymbol{e_{ss}}$ $(\boldsymbol{rad})$
		PID digital SW sin restricciones			
0 a 3	1	1.28	0.8727	1.0003	2.7967e-4
3 a 6	0.5	1.28	0.8731	0.4999	-1.3512e-4
6 a 10	2	1.08	4.9871	1.9999	-6.3656-5
		PID digital SW con restricciones			
0 a 3	1	1.28	0.8727	1.0003	2.7067e-4
3 a 6	0.5	1.28	0.8731	0.4999	-1.3541e-4
6 a 10	2	1.23	0.6546	1.9999	-6.3656e-5
		PID Hardware precisión punto fijo			
0 a 3	1	1.28	0.8727	1.0003	2.7057e-4
3 a 6	0.5	1.28	0.8731	0.4999	-1.3541e-4
6 a 10	2	1.23	0.6546	1.9999	-6.3694e-5
		PID Hardware precisión punto flotante			
0 a 3	1	1.28	0.8727	1.0003	2.6089e-4
3 a 6	0.5	1.28	0.8731	0.4999	-1.3229e-4
6 a 10	2	1.23	0.6546	1.9999	-1.0468e-4

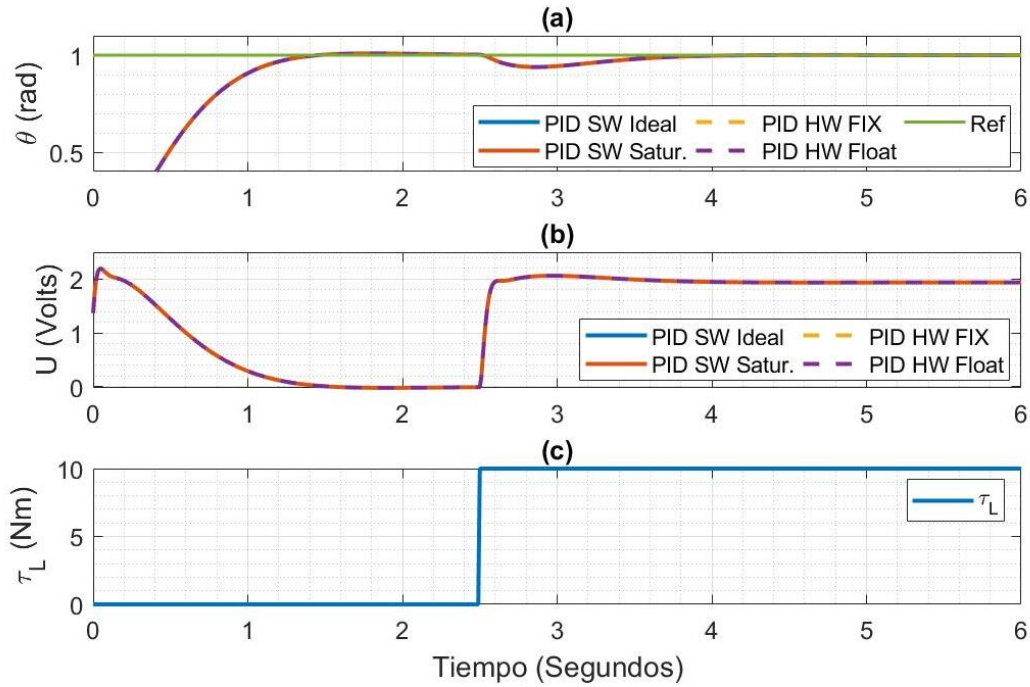
Desempeño Del Sistema Ante Perturbaciones De Carga.

Para observar el desempeño del sistema ante una perturbación en la carga se ha introducido una perturbación de 10Nm, el cual representa el 36.5% del torque máximo del motor en vacío y a voltaje nominal. La Figura 48 muestra la respuesta del sistema ante la perturbación de carga que sucede a los 2.5 s de haber iniciado la experimentación. Una vez que la perturbación inicia, el sistema toma aproximadamente 1.2 s en recuperarse,

sin embargo, el sistema logra su cometido de regular el sistema antes de los 1.5 s. Por otro lado, el esfuerzo de control presenta un aumento aproximado de 2 V, lo cual se debe a que el controlador actúa para rechazar la perturbación y así poder mantener al sistema en el régimen deseado. La Tabla 22 muestra un resumen del comportamiento del sistema ante la perturbación de carga introducida; donde se observa que esfuerzo de control es mucho menor al obtenido con el controlador 1DoF-PID.

Figura 48

Respuesta del sistema ante perturbación de carga con el controlador 2DoF-PID



Nota. (a) Respuesta del sistema ante perturbación de carga. (b) Respuesta de la acción de control. (c) Perturbación de carga introducida en el sistema.

Tabla 22

Desempeño del Sistema ante perturbación de carga con el controlador 2DoF-PID.

Inicio de perturbación (seg)	**Perturbación de carga** (Nm)	**Esfuerzo de control sin perturbación** (V)	**Esfuerzo de control con perturbación** (V)	**Tiempo de recuperación (s)**	e_{ss} (rad)
		PID digital SW sin restricciones			
2.5	10	3.0623	5.2524	1.2	3.4932e-10
		PID digital SW con restricciones			
2.5	10	3.0623	5.2524	1.2	3.4932e-10
		PID Hardware precisión punto fijo			
2.5	10	3.0726	5.2686	1.2	5.4932e-6
		PID Hardware precisión punto flotante			
2.5	10	3.0652	5.2644	1.2	1.9127e-4

Modelos *IP-Cores* Generados

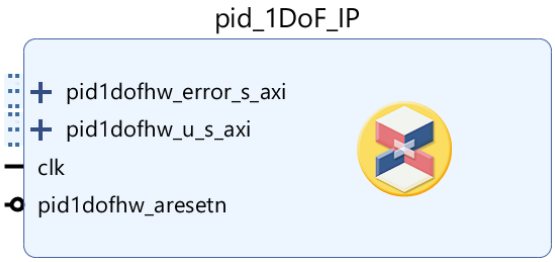
Una vez verificado el desempeño del *Hardware* de los *IP-Cores*, se pueden generar los archivos necesarios para que el diseñador del *SoC* pueda utilizarlos en sus proyectos. A través de la herramienta *SXG,* se pueden configurar los Buses *AXI*, y mediante el *Software Vivado IDE*, se puede importar el *IP-Cores* generado. La Figura 49 muestra el modelo del *IP-Core* generado del controlador *1DoF-PID,* la entrada de la señal de error del proceso y la señal de salida de control se configuraron como interfaces de bus *AXI*; resultando en las siguientes interfaces:

- ***PID1dofhw_error_s_AXI:*** Recibe el valor de la señal de error del proceso
- ***PID1dofhw_u_s_AXI****:* Entrega el valor de la acción de control.
- ***PID1dofhw_aresetn:*** Gobierna sobre el *reset* asincrónico del *IP-Core.*
- ***Clk:*** Se encargada de recibir la señal de reloj del sistema

Las interfaces *AXI* facilitan la comunicación con otros *IP-Cores,* como: procesadores, memorias, interfaces de comunicación, controladores de periféricos, etc., para desarrollar un sistema *SoC* embebido más complejo.

Figura 49

IP-Core del controlador PID de un grado de libertad

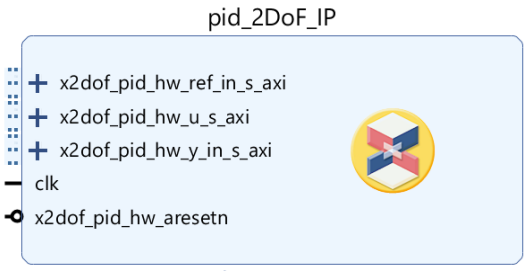


De igual manera en la Figura 50 se representa el *IP-Core* del controlador *PID* de dos grados de libertad, el cual se generó a través de la herramienta de *SXG, la cual* produce los archivos necesarios para integrar el *IP-Core* en la plataforma de diseño de sistemas *SoCs* embebidos; el objetivo es que el *IP-Core* diseñado pueda ser combinado con *IP-Cores* de terceros para generar un *System on Chip* moderno. Al igual que para el *IP-Core 1DoF-PID*, las entradas de la señal de referencia, la señal del proceso y la salida de control fueron configuradas como interfaces *AXI*, para que puedan ser utilizadas en la integración con otros *IP-Cores*; como controladores de periféricos y procesadores. De esta manera se ha generado las siguientes interfaces:

- ***x2dof_PID_hw_red_in_s_AXI*:** Recibe el valor de referencia deseado
- ***x2dof_PID_hw_y_s_AXI*:** Se encarga de recibir el valor medido del proceso,
- **x2dof_*PID*_hw_u_s_*AXI*:** Es utilizada para devolver el valor que debe tomar la acción de control.
- ***x2dof_PID_hw_aresetn:*** Gobierna el *reset* asíncrono del *IP-Core*.
- ***Clk:*** Se encargada de recibir la señal de reloj del sistema

Figura 50

IP-Core del controlador PID de dos grados de libertad



Resumen

En esta sección se realizó un análisis de la implementación y desempeño de los controladores modelados como *IP-Cores*. En primera instancia, dejando de un lado la pregunta de: ¿cuál es el mejor controlador?, se puede observar que se obtuvieron resultados satisfactorios en la implementación de los controladores *1DoF-PID* y *2DoF-PID* como *IP-Cores,* debido a que poseen un desempeño muy semejante a los simulados solamente en *Software*; incluso comparten las mismas restricciones físicas de saturación en la señal de control. Por otro lado, en lo relacionado al desempeño de cada controlador, se evidencia como el controlador *2DoF-PID*, obtuvo un mejor comportamiento en la dinámica del sistema; aunque en ambos casos se obtuvo los mismos polos dominantes como no dominantes, el controlador de dos grados de libertad permite manipular los ceros para ubicarlos en una región que no tenga impacto sobre la dinámica del sistema. En el controlador *1DoF-PID* no se pudo realizar una manipulación de los ceros, por lo que estos influyeron en la dinámica final del sistema controlado, al ubicarse muy cerca de los polos dominantes; este problema podría ser solucionado aplicando otra técnica de control más sofisticada a la que se utilizó en este estudio, sin embargo, hay que resaltar que con la misma técnica de diseño aplicada al controlador *2DoF-PID* se obtuvieron resultados satisfactorios.

Por otra parte, el uso de recursos para implementar cada controlador varía en función de la estrategia de aproximación utilizada para las operaciones *DSP*, ya que el controlador *PID* de un grado de libertad con aproximación de punto flotante alcanza el 7.7% de los recursos de una *FPGA Artix 35T*; mientras que el controlador *2DoF-PID* alcanza el 10,62% de los mismos.

Por otro lado, el uso de una aproximación de punto fijo sólo utiliza el 1.39% de los recursos para el controlador *PID* de un grado de libertad y el $2{,}07\%$ para el controlador *PID* de dos grados de libertad. El recurso que más se ve afectado en cada estrategia de aproximación, es el uso de recursos *LUTs,* ya que para el caso del controlador *1DoF-PID* pasó de necesitar un 3.86% de *LUTs* con aproximación de punto fijo, a utilizar un $23{,}03\%$ de *LUTs* con aproximación de punto flotante, lo que supone un aumento significativo de recursos del 19%. Algo similar ocurre con el controlador *2DoF-PID*, que pasó de utilizar 5.89% de *LUTs* con aproximación de punto fijo a utilizar 31.58% de los recursos *LUTs* con precisión de coma flotante, esto representa un aumento significativo del 25.7% de recursos *LUTs*. Es importante tener en cuenta este aspecto, debido a que los diseños de *SoCs* modernos se componen de un conjunto de *IP-Cores* que trabajan coordinados entre sí, consecuentemente, cada *IP-Core* necesita recursos *Hardware* finitos que puedan ser implementados en un mismo *SOC*. Por lo tanto, el diseñador debe elegir la estrategia que más le convenga, dependiendo de la precisión requerida y de la cantidad de recursos de *Hardware* disponibles en el *SOC.*

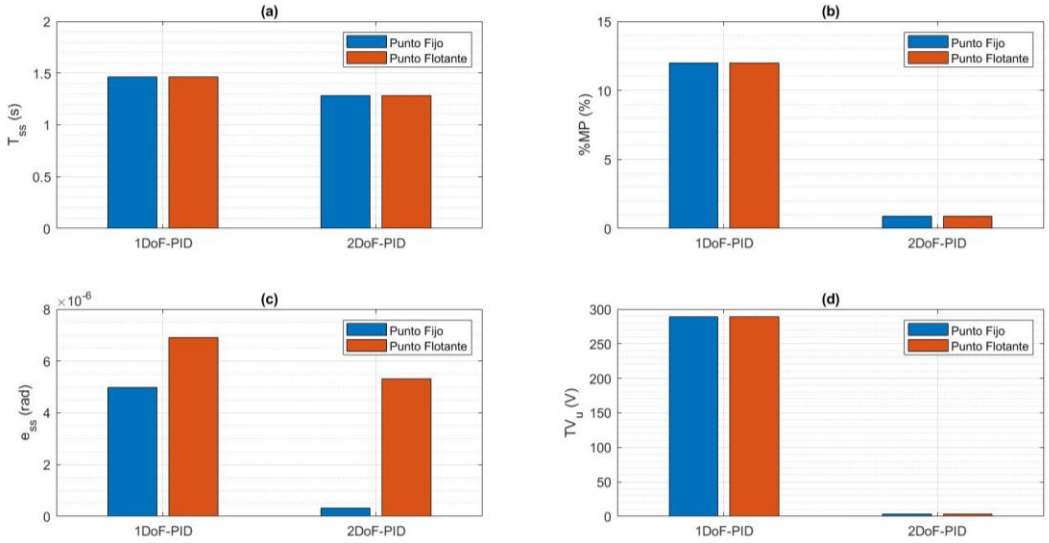
La Figura 51 muestra el análisis del desempeño de los controladores implementados en *Hardware* ante una entrada escalón unitario. La Figura 51(a) indica el tiempo de establecimiento alcanzado por cada uno de los controladores, los mismos que tienen un tiempo de establecimiento menor al impuesto por el diseño de $1.5\ s$; en la Figura 51(b) se observa el porcentaje de sobre impulso que tiene el sistema en cada controlador,

existe una gran diferencia entre el controlador *1DoF-PID* y el controlador *2DoF-PID*, esto se debe a que los ceros presentes en el controlador de un grado de libertad afectan la dinámica del sistema, adicionalmente al tener una acción de control que supera el límite establecido de 240V, se limita la acción de control a través del bloque de saturación. Consecuentemente, aparece el efecto *Windup* provocado por la acumulación de error en la acción integral; esta acumulación de energía ocasiona un incremento del porcentaje de sobre impulso, que posteriormente afecta al tiempo de establecimiento ya que requiere una mayor duración para descargar la energía acumulada. Es necesario resaltar que este inconveniente no ocurre con el controlador *2DoF-PID,* cuyo porcentaje de sobre impulso es aproximadamente 1%.

La Figura 51(c) muestra el error de estado estable que posee cada controlador, si bien este parámetro en la industria es despreciable, debido a que prácticamente el error es cero; se debe destacar que la implementación que utiliza la precisión de punto fijo obtuvo un error menor que el obtenido con la precisión de punto flotante. Normalmente, se podría pensar que el uso de una precisión de punto flotante permite obtener un menor error en el resultado de las operaciones *DSP*, este fenómeno no siempre se produce, la correcta selección del número de bits en la precisión de punto fijo, posibilita establecer una buena exactitud. Por otra parte, la precisión de punto fijo tiene la ventaja de mantener un bajo porcentaje de recursos de *Hardware*.

Figura 51

Desempeño de los parámetros del sistema controlado por 1DoF-PID y 2DoF-PID



Nota. (a) Tiempo de establecimiento. (b) Porcentaje de sobre impulso. (c) Error de estado estable. (d) Esfuerzo de la señal de control

La Figura 51(d) muestra el esfuerzo de control de cada compensador, aquí no tiene mucha relevancia la precisión escogida, pero si la estrategia de control, ya que el controlador *1DoF-PID* tiene un esfuerzo de control mucho mayor al controlador *2DoF-PID*; esto provoca que la acción de control se sature y genere el fenómeno del *Windup* para el sistema gobernado con el controlador de un grado de libertad, mientras que con el controlador de dos grados de libertad se obtiene una acción de control mucho más suave, sin saturaciones y evitando así el efecto *Windup*. Como resultado se obtiene un sistema servo-controlado con mejor rendimiento, en donde los parámetros de desempeño se encuentran dentro del rango establecido de diseño.

Conclusiones Y Trabajos Futuros

Introducción

En el capítulo final de esta tesis, se enuncian diferentes conclusiones derivadas de la investigación efectuada, así como los resultados obtenidos y sus principales aportaciones. Además, se mencionan líneas de investigación abiertas que pueden dar lugar a futuros trabajos.

Conclusiones

A continuación, se resumen las principales conclusiones de la investigación realizada.

Después de un análisis de la metodología propuesta, se observa que la generación de *IP-Cores* basados en modelos permite al diseñador de *Hardware* acortar el tiempo de diseño y verificar los errores de una manera más rá*pid*a, sin la necesidad de realizar pruebas de verificación en un sistema real. Infiriéndose que, las herramientas de simulación y análisis de *Hardware* permiten validar cuantitativamente el rendimiento y el funcionamiento del *Hardware* de los *IP-Cores* en un entorno controlado.

Los modelos de *Hardware* que utilizan sistemas de precisión de punto fijo en las operaciones *DSP*, no siempre generan una pérdida de exactitud en el resultado del procesamiento de la señal. Sin embargo, en la presente investigación la estrategia de punto fijo utiliza un 80% menos de recursos de *Hardware*, comparada a la implementación que utiliza un enfoque de punto flotante en las operaciones *DSP*. No obstante, la estrategia de punto flotante puede utilizarse en aplicaciones más genéricas, en las que los recursos *Hardware* no sean un inconveniente y en las que no sea imperativo un estudio exhaustivo del número de bits necesarios para representar la parte entera y la

decimal, dando lugar a *IP-Cores* genéricos, con buenas prestaciones y fáciles de adaptar a cualquier sistema.

Las pruebas de validación realizadas demostraron que el rendimiento del modelo implementado en *Hardware* es similar al ejecutado por el *Software*. Sin embargo, es necesario destacar las ventajas de mantener una implementación en *Hardware*, porque se produce una aceleración de los procesos, los cuales se ejecutan de manera concurrente, produciendo la liberación de la carga computacional al procesador implementado en el *SoC*, lo que permite que el procesador realice tareas dedicadas o aplicaciones de tiempo real.

Un *IP-Core* está destinado a cumplir una función específica e interactuar con otros *IP-Cores* que conforman un sistema embebido completo; para lo cual es necesario incorporar una interfaz de comunicación estandarizada como *AXI*-4, que permita una fácil integración con *IP-Cores* de terceros, y a su vez minimice el tiempo de diseño e implementación de *SoCs* destinados a la Industria 4.0.

El controlador *PID* de dos grados de libertad, implementado como *IP-Core*, presenta superioridad sobre el controlador *PID* de un grado de libertad. El primero posee un comportamiento más cercano al deseado, no presenta efecto *windup* causado por la acción integradora y el bloque de saturación. Debido a que a pesar de ser factible diseñar y desarrollar un sistema *anti-windup* en el *PID* de un grado de libertad, que permitiría mejorar el comportamiento dinámico del sistema; se debe tener en cuenta que el incremento de *Hardware* adicional para desarrollar el *anti-windup,* podría generar un mayor coste de recursos en *Hardware,* que el obtenido con el controlador *PID* de dos grados de libertad.

Resumen de las principales aportaciones

Esta sección presenta las principales contribuciones de la presente tesis. El orden de presentación está determinado en función de la importancia de las contribuciones.

1. **Propuesta de la metodología basada en modelos para la generación de *IP-Cores***

Es necesario que los diseñadores de *SoCs* dispongan de un catálogo más amplio de *IP-Cores* para la formación de sistemas embebidos que conformen los *CPSs* de la Industria 4.0. En este trabajo se ha presentado una metodología orientada a los diseñadores de *Hardware*, para posibilitar el diseño y generación de *IP-Cores* que puedan ser integrados en un *SoC* moderno a través de interfaces de comunicación estándar como *AXI-4* y de tipo *Plug and Play*. En este sentido, se han propuesto dos arquitecturas de controladores *PID*, para formar parte de dispositivos embebidos capaces de controlar sistemas *CPS* como brazos robóticos, presentes en los procesos de automatización de la Industria 4.0.

El uso de aceleradores de *Hardware* busca que los sistemas *CPS* basados en sistemas *SoCs* embebidos, puedan cumplir con los requisitos esperados en los dispositivos que conforman la Industria 4.0, de forma que estos componentes implementados como *IP-Cores* puedan cooperar con dispositivos similares, integren los modelos de control adecuados, garanticen la alta calidad de los datos procesados y puedan soportar los estándares de comunicación a*SoC*iados a la industria y a los dispositivos *IoT.*

2. **Evaluación experimental de estrategias de control modeladas como *IP-Cores***

En la presente investigación se han realizado diferentes experimentos para validar los modelos generados con la metodología propuesta, que son aplicables a los procesos de control presentes en la Industria 4.0. La validación se ha realizado sobre una plataforma de *Hardware* reconfigurable basada en la *FPGA Artix 35T*, desarrollada por la empresa *Digilent*; que permite evaluar el rendimiento de los controladores *PID* de un grado y de dos grados de libertad, se ha utilizado la metodología de simulación *HIL*, que integra elementos de *Hardware* como los *IP-Cores* implementados en la *FPGA* y elementos de *Software* como el modelo del sistema servomotor propuesto.

Los servomotores se ubican en las articulaciones de los brazos robóticos, presentes en los procesos de fabricación industrial inteligente, cada controlador se desarrolló mediante dos estrategias de aproximación en las operaciones *DSP*. La primera aproximación, que emplea la precisión de punto fijo, utiliza una cantidad moderada de recursos de *Hardware* y produce un comportamiento similar al obtenido en simulaciones generadas por *Software*. La segunda propuesta utiliza una precisión de punto flotante, que requiere mayor cantidad de recursos de *Hardware*, pero produce un comportamiento casi similar al obtenido con la implementación en coma fija. Por lo tanto, para este estudio es más rentable utilizar los *IP-Cores* con precisión de punto fijo, debido a su bajo consumo de recursos de *Hardware* y buen rendimiento.

3. **Generación del *Hardware* de *IP-Cores* implementables en *SoCs***

Se realizó un estudio de *IP-Cores*, presentando una propuesta de diseño basada en una metodología de modelado, generación y verificación del *Hardware*, que se caracteriza por disponer de un sistema de comunicación basado en protocolos *AXI*, el

cuál es un estándar ampliamente utilizado para integrar diferentes *IP-Cores* en un sistema embebido.

El producto final de esta metodología es generar *IP-Cores,* que no sólo estén destinados a cumplir una función, sino que tengan la capacidad de cooperar con diferentes módulos de *Hardware* para cubrir los aspectos requeridos por la Industria 4.0. De este modo, los *IP-Cores* propuestos para el control de procesos, pueden integrarse con otros aceleradores de *Hardware* dedicados a realizar diferentes funciones disponibles en el mercado como: módulos de gestión de comunicaciones *IoT*, procesadores de inteligencia artificial, controladores de periféricos, entre otros. La metodología propuesta servirá como herramienta base para facilitar la generación y verificación de *IP-Cores* basados en modelos que respondan a las necesidades de la Industria 4.0.

Líneas De Trabajo Futuras

Esta sección expone diversas líneas de investigación que el autor propone para dar continuidad al trabajo presentado en esta tesis. Estas líneas son:

- **Integración del *IP-Core* en un diseño *SoC* para gobernar un sistema que pueda formar parte de la industria 4.0.**

Una de las aportaciones de esta tesis es la evaluación del *IP-Core* propuesto, que forma parte del ciclo de diseño para los núcleos de propiedad intelectual. Este procedimiento se ha presentado en un entorno experimental controlado, que permite evaluar el rendimiento individual del *Hardware* del *IP-Core*, sin dejar de lado, que estos poseen la capacidad de integrarse con otros *IP-Cores* para formar un *SoC*. Por lo que es necesario realizar la evaluación del *IP-Core* en un entorno cooperativo, utilizando un conjunto adicional de *IP-Cores* para formar una plataforma *SoC*.

La plataforma embebida propuesta estaría integrada por los siguientes elementos: un procesador encargado de generar el perfil de control de un sistema servo-controlado, el *IP-Core* encargado de realizar la función de controlador, y otros *IP-Cores* encargados de controlar los periféricos de entrada y salida del *SoC*. Adicionalmente, se podrían contemplar otros *IP-Cores* básicos como *ROMs* y *RAMs*, esperando formar una plataforma *SoC* que sea candidata a constituir un *CPS*, el mismo que debe intercambiar información entre los entornos físico y virtual en la Industria 4.0. Para realizar esta experimentación, se sugiere proponer una metodología de diseño y verificación de plataformas *SoCs,* basada en la integración de *IP-Cores* enfocados y modelados para satisfacer las necesidades de la cuarta revolución industrial.

- **Generar *IP-Cores* utilizando la metodología de diseño propuesta de controladores de uso general cuyos parámetros puedan ser configurados por procesadores.**

La necesidad de incorporar este tipo de controladores en diferentes procesos de la Industria 4.0 como: temperatura, flujo, servo-control o sistemas de presión, requiere una demanda de *IP-Cores* que no sean modelados desde cero. Es posible modelar *IP-Cores* con una estructura de *Hardware* fija, pero con parámetros de control reconfigurables, resultando en *IP-Cores* de controladores adaptativos con ganancias de control ajustables por un procesador a través de interfaces *AXI*. Esto reduciría aún más el tiempo de implementación, y la ingeniería no recursiva.

- **Generación de estrategias para reducir efectos de la acción integral en los controladores *PID* en *Hardware***

Es necesario mejorar el desempeño del controlador *1DoF-PID*, que posee un rendimiento menor para sistemas servo-controlados; por lo cual se debe incluir un sistema

anti-windup en el diseño del *Hardware*, que permitirá la reducción considerable del efecto *windup* provocado por el bloque de saturación. Esto podría generar un incremento en el coste del *Hardware*, sin embargo, se incrementaría del desempeño del controlador en este tipo de servo-sistemas.

Bibliografía

Abdelati, M. (2016). FPGA-Based PID Controller Implementation. *IUG Journal of Natural Studies*, *14*(1), 1–19. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.104.1099&rep=rep1&type=pdf

Abdelhamid, B., Radhouane, L., & Bilel, A. (2017). Real time implementation of perturb and observe algorithm and PI controller for DC/DC converter. *2017 18th International Conference on Sciences and Techniques of Automatic Control and Computer Engineering, STA 2017 - Proceedings*, *2018-Janua*, 520–526. https://doi.org/10.1109/STA.2017.8314946

Aheleroff, S., Xu, X., Lu, Y., Aristizabal, M., Pablo Velásquez, J., Joa, B., & Valencia, Y. (2020). IoT-enabled smart appliances under industry 4.0: A case study. *Advanced Engineering Informatics*, *43*(December 2019), 101043. https://doi.org/10.1016/j.aei.2020.101043

Alladi, T., Chamola, V., Parizi, R. M., & Choo, K. K. R. (2019). Blockchain Applications for Industry 4.0 and Industrial IoT: A Review. *IEEE Access*, *7*, 176935–176951. https://doi.org/10.1109/ACCESS.2019.2956748

Almatheel, Y. A., & Abdelrahman, A. (2017, February 27). Speed control of DC motor using Fuzzy Logic Controller. *Proceedings - 2017 International Conference on*

Communication, Control, Computing and Electronics Engineering, ICCCCEE 2017. https://doi.org/10.1109/ICCCCEE.2017.7867673

Alvarez-Gonzalez, F., Griffo, A., Sen, B., & Wang, J. (2017). Real-Time Hardware-in-The-Loop Simulation of Permanent-Magnet Synchronous Motor Drives under Stator Faults. *IEEE Transactions on Industrial Electronics*, *64*(9), 6960–6969. https://doi.org/10.1109/TIE.2017.2688969

ARM. (n.d.-a). *AMBA AXI and ACE Protocol Specification*. Retrieved June 2, 2021, from https://developer.arm.com/documentation/ihi0022/hc

ARM. (n.d.-b). *AMBA Specifications – Arm*. Retrieved May 19, 2021, from https://www.arm.com/products/silicon-ip-system/embedded-system-design/amba-specifications

Åström, K. J., & Wittenmark, B. (2013). Computer-controlled systems: theory and design. In *Courier Corporation* (Vol. 23, Issue 4). Courier Corporation. https://doi.org/10.1177/002072098602300430

Baklouti, M., Marquet, P., Dekeyser, J. L., & Abid, M. (2015). FPGA-based many-core System-on-Chip design. *Microprocessors and Microsystems*, *39*(4–5), 302–312. https://doi.org/10.1016/j.micpro.2015.03.007

Bauer, M., Horch, A., Xie, L., Jelali, M., & Thornhill, N. (2016). The current state of control loop performance monitoring - A survey of application in industry. *Journal of Process Control*, *38*, 1–10. https://doi.org/10.1016/j.jprocont.2015.11.002

Bazeghi, C. (2016). IP Flows. In *Designing with Xilinx® FPGAs: Using Vivado* (pp. 23–33). Springer International Publishing. https://doi.org/10.1007/978-3-319-42438-5_3

Bernstein, D. S. (n.d.). *A Plant Taxonomy for Designing Control Experiments*.

Carvalho, N., Chaim, O., Cazarini, E., & Gerolamo, M. (2018). Manufacturing in the fourth industrial revolution: A positive prospect in Sustainable Manufacturing. *Procedia Manufacturing*, *21*, 671–678. https://doi.org/10.1016/j.promfg.2018.02.170

Chakravarthi, V. S. (2020). A Practical Approach to VLSI System on Chip (SoC) Design. In *A Practical Approach to VLSI System on Chip (SoC) Design*. https://doi.org/10.1007/978-3-030-23049-4

Chen, B., Wan, J., Shu, L., Li, P., Mukherjee, M., & Yin, B. (2017). Smart Factory of Industry 4.0: Key Technologies, Application Case, and Challenges. *IEEE Access*, *6*(c), 6505–6519. https://doi.org/10.1109/ACCESS.2017.2783682

Cigánek, J., Kozák, S., Kozáková, A., Institute of Electrical and Electronics Engineers, IEEE Czechoslovakia Section. Control Systems Society Chapter, & Slovenská spoločnosť pre kybernetiku a informatiku. (2018). Research and education for industry 4.0: Present development. In *2018 Cybernetics \& Informatics (K\&I)* (pp. 1–8).

Corna, N., Lusardi, N., Garzetti, F., Geraci, A., & Gustin, M. (2019). Multi-Channel High-Resolution Pulse-Width Modulation IP-Core Implementation for FPGA and SoC Device. *2019 IEEE Nuclear Science Symposium and Medical Imaging Conference, NSS/MIC 2019*, 2019–2021. https://doi.org/10.1109/NSS/MIC42101.2019.9059801

da Silva, L. R., Flesch, R. C. C., & Normey-Rico, J. E. (2018). Analysis of Anti-windup Techniques in PID Control of Processes with Measurement Noise. *IFAC-PapersOnLine*, *51*(4), 948–953. https://doi.org/10.1016/J.IFACOL.2018.06.100

Deschamps, J.-P., Valderrama, E., & Terés, L. (2019). Complex Digital Circuits. In *Complex Digital Circuits*. Springer International Publishing. https://doi.org/10.1007/978-3-030-12653-7

Deshmukh, G. L., & Kadu, C. B. (2017). Design of two degree of freedom PID controller for temperature control system. *International Conference on Automatic Control and Dynamic Optimization Techniques, ICACDOT 2016*, 586–589. https://doi.org/10.1109/ICACDOT.2016.7877653

Di Mascio, S., Menicucci, A., Gill, E., Furano, G., & Monteleone, C. (2021). Open-source

IP cores for space: A processor-level perspective on soft errors in the RISC-V era. *Computer Science Review*, *39*. https://doi.org/10.1016/j.cosrev.2020.100349

Diao, L., Tang, J., Loh, P. C., Yin, S., Wang, L., & Liu, Z. (2018). An Efficient DSP-FPGA-Based Implementation of Hybrid PWM for Electric Rail Traction Induction Motor Control. *IEEE Transactions on Power Electronics*, *33*(4), 3276–3288. https://doi.org/10.1109/TPEL.2017.2707639

Digilent. (2016a). *Basys 3™ FPGA Board Reference Manual*. www.digilentinc.com

Digilent. (2016b). *ZYBO™ FPGA Board Reference Manual*. www.digilentinc.com

Distel, D. (2017). *Markets for Technology in the Semiconductor Industry – The Role of Ability-Related Trust in the Market for IP Cores*. TECHNISCHE UNIVERSITÄT MÜNCHEN.

Erickson, R. W., & Maksimović, D. (2020). Digital Control of Switched-Mode Power Converters. In *Fundamentals of Power Electronics* (pp. 805–845). Springer International Publishing. https://doi.org/10.1007/978-3-030-43881-4_19

Garrido-Hidalgo, C., Olivares, T., Ramirez, F. J., & Roda-Sanchez, L. (2019). An end-to-end Internet of Things solution for Reverse Supply Chain Management in Industry 4.0. *Computers in Industry*, *112*, 103127. https://doi.org/10.1016/j.compind.2019.103127

Ghobakhloo, M. (2020). Industry 4.0, digitization, and opportunities for sustainability. In *Journal of Cleaner Production* (Vol. 252). Elsevier Ltd. https://doi.org/10.1016/j.jclepro.2019.119869

Griffiths, F., & Ooi, M. (2018). The fourth industrial revolution - Industry 4.0 and IoT [Trends in Future I and M]. *IEEE Instrumentation and Measurement Magazine*, *21*(6). https://doi.org/10.1109/MIM.2018.8573590

Hategekimana, F., Whitaker, T. J. L., Pantho, M. J. H., & Bobda, C. (2018). Secure integration of non-trusted IPs in SoCs. *Proceedings of the 2017 Asian Hardware*

Oriented Security and Trust Symposium, AsianHOST 2017, *2018-May*, 103–108. https://doi.org/10.1109/AsianHOST.2017.8354003

He, J., Guo, X., Meade, T., Dutta, R. G., Zhao, Y., & Jin, Y. (2019). SoC interconnection protection through formal verification. *Integration*, *64*(September), 143–151. https://doi.org/10.1016/j.vlsi.2018.09.007

Hofmann, E., & Rüsch, M. (2017). Industry 4.0 and the current status as well as future prospects on logistics. *Computers in Industry*, *89*, 23–34. https://doi.org/10.1016/j.compind.2017.04.002

Huang, L., Lee, Y. P., Moon, Y. S., & Bae, Y. C. (2017). Noble Implementation of Motor Driver with All Programmable SoC for Humanoid Robot or Industrial Device. *International Journal of Humanoid Robotics*, *14*(4), 1–23. https://doi.org/10.1142/S0219843617500281

Indira, M., & Swapna, B. (2017). Fpga Implementation of low Power Multi Channel Generalized Pid Controller for Industrial Automation Applications. *International Journal of Engineering Trends and Applications*, *4*(6). https://doi.org/10.15373/22778179/nov2012/20

Indri, M., Grau, A., & Ruderman, M. (2018). Guest Editorial Special Section on Recent Trends and Developments in Industry 4.0 Motivated Robotic Solutions. *IEEE Transactions on Industrial Informatics*, *14*(4), 1677–1680. https://doi.org/10.1109/TII.2018.2809000

Jain, S., Govani, P., Poddar, K. B., Lal, A. K., & Parmar, R. M. (2016, October 17). Functional verification of DSP based on-board VLSI designs. *2016 International Conference on VLSI Systems, Architectures, Technology and Applications, VLSI-SATA 2016*. https://doi.org/10.1109/VLSI-SATA.2016.7593030

Johnson, M. A., Moradi, M. H., Crowe, J., Tan, K. K., Lee, T. H., Ferdous, R., Katebi, M. R., Huang, H. P., Jeng, J. C., Tang, K. S., Chen, G. R., Man, K. F., Kwong, S.,

Sánchez, A., Wang, Q. G., Zhang, Y., Zhang, Y., Martin, P., Grimble, M. J., & Greenwood, D. R. (2005). PID control: New identification and design methods. In *PID Control: New Identification and Design Methods*. Springer London. https://doi.org/10.1007/1-84628-148-2

Kornaros, G. (2018). *MULTI-CORE EMBEDDED SYSTEMS*. CRC Press.

Kovela, I. M., Viter, O. S., & Ivaniuk, O. O. (2017). Udc 681.5 optimization of discrete pid-algorithms structure. *Комп'ютерні Технології Друкарства*, *33*, 8--15.

Kuo, B., & Golnaraghi, F. (n.d.). *Ninth Edition Farid Golnaraghi • Benjamin C . Kuo* (9th ed.). 2009.

Lee, J., Bagheri, B., & Kao, H. A. (2015). A Cyber-Physical Systems architecture for Industry 4.0-based manufacturing systems. *Manufacturing Letters*, *3*, 18–23. https://doi.org/10.1016/j.mfglet.2014.12.001

Leong, Y. P., & Doyle, J. C. (2017). Effects of Delays, Poles, and Zeros on Time Domain Waterbed Tradeoffs and Oscillations. *IEEE Control Systems Letters*, *1*(1), 122–127. https://doi.org/10.1109/LCSYS.2017.2710327

Lins, T., & Oliveira, R. A. R. (2020). Cyber-physical production systems retrofitting in context of industry 4.0. *Computers and Industrial Engineering*, *139*. https://doi.org/10.1016/j.cie.2019.106193

Lorandel, J., Prevotet, J. C., & Helard, M. (2015). Dynamic power evaluation of LTE wireless baseband processing on FPGA. *Conference on Design and Architectures for Signal and Image Processing, DASIP*, *2015-Decem*. https://doi.org/10.1109/DASIP.2015.7367265

Macko, D., Jelemenská, K., & Čičák, P. (2018). Simplifying low-power SoC top-down design using the system-level abstraction and the increased automation. *Integration*, *63*(June), 101–114. https://doi.org/10.1016/j.vlsi.2018.06.001

MathWorks. (2020). *Model-Based Design for Embedded Control Systems*.

https://www.mathworks.com/content/dam/mathworks/white-paper/gated/model-based-design-with-simulation-white-paper.pdf

MatWorks. (2020). *Model-Based Design - MATLAB & Simulink*. https://www.mathworks.com/solutions/model-based-design.html

Meng, F., Liu, S., & Liu, K. (2020). Design of an Optimal Fractional Order PID for Constant Tension Control System. *IEEE Access*, *8*, 58933–58939. https://doi.org/10.1109/ACCESS.2020.2983059

Mohamed, K. S. (2015). IP Cores Design from Specifications to Production: Modeling, Verification, Optimization, and Protection. In *IP Cores Design from Specifications to Production: Modeling, Verification, Optimization, and Protection*. https://doi.org/10.1007/978-3-319-22035-2

Mohamed, K. S. (2016). *IP Cores Design from Specifications to Production: Modeling, Verification, Optimization, and Protection* (pp. 13–50). https://doi.org/10.1007/978-3-319-22035-2_2

Nunes, M. L., Pereira, A. C., & Alves, A. C. (2017). Smart products development approaches for Industry 4.0. *Procedia Manufacturing*, *13*, 1215–1222. https://doi.org/10.1016/j.promfg.2017.09.035

O'Donovan, P., Gallagher, C., Bruton, K., & O'Sullivan, D. T. J. (2018). A fog computing industrial cyber-physical system for embedded low-latency machine learning Industry 4.0 applications. *Manufacturing Letters*, *15*, 139–142. https://doi.org/10.1016/j.mfglet.2018.01.005

Ogata, K. (1995). *Discrete-time control systems*. Prentice-Hall, Inc.

Onik, M. M. H., Kim, C. S., & Yang, J. (2019). Personal Data Privacy Challenges of the Fourth Industrial Revolution. *International Conference on Advanced Communication Technology, ICACT*, *2019-Febru*, 635–638. https://doi.org/10.23919/ICACT.2019.8701932

Palma, J. C., De Mello, A. V., Möller, L., Moraes, F., & Calazans, N. (2002). Core communication interface for FPGAs. *Proceedings - 15th Symposium on Integrated Circuits and Systems Design, SBCCI 2002*, 183–188. https://doi.org/10.1109/SBCCI.2002.1137656

Pathak, D., Sagar, G., & Gaur, P. (2020). An Application of Intelligent Non-linear Discrete-PID Controller for MPPT of PV System. *Procedia Computer Science*, *167*, 1574–1583. https://doi.org/10.1016/j.procs.2020.03.368

Pereira, A. C., & Romero, F. (2017). A review of the meanings and the implications of the Industry 4.0 concept. *Procedia Manufacturing*, *13*, 1206–1214. https://doi.org/10.1016/j.promfg.2017.09.032

Perkovic, N., Pranjkic, M., Kolak, I., & Velikic, G. (2017). System for automotive machine vision. *IEEE International Conference on Consumer Electronics - Berlin, ICCE-Berlin*, *2017-Septe*, 243–245. https://doi.org/10.1109/ICCE-Berlin.2017.8210638

Podržaj, P. (2018). Contionuous VS discrete PID controller. *2018 IEEE 9th International Conference on Mechanical and Intelligent Manufacturing Technologies, ICMIMT 2018*, *2018-Janua*, 177–181. https://doi.org/10.1109/ICMIMT.2018.8340444

Prasad, A., Nath, D., Boddupalli, S., Bhunia, S., & Ray, S. (n.d.). *ARK : Architecture for Security Resiliency in SoC Designs with NetworK-on-Chip (NoC) Fabrics*. 88–93.

Riahi, P. A., Navabi, Z., & Lombardi, F. (2005). Simulating faults of combinational IP core-based SOCs in a PLI environment. *Proceedings - IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems*, 389–397. https://doi.org/10.1109/dftvs.2005.60

Rodriguez-Andina, J. J., Valdes-Pena, M. D., & Moure, M. J. (2015). Advanced Features and Industrial Applications of FPGAS-A Review. *IEEE Transactions on Industrial Informatics*, *11*(4), 853–864. https://doi.org/10.1109/TII.2015.2431223

Sengupta, A., Bhadauria, S., & Mohanty, S. P. (2016). Embedding low cost optimal

watermark during high level synthesis for reusable IP core protection. *Proceedings - IEEE International Symposium on Circuits and Systems*, *2016-July*, 974–977. https://doi.org/10.1109/ISCAS.2016.7527405

Sengupta, A., & Kachave, D. (2018). Forensic engineering for resolving ownership problem of reusable IP core generated during high level synthesis. *Future Generation Computer Systems*, *80*, 29–46. https://doi.org/10.1016/j.future.2017.08.001

Sengupta, A., & Roy, D. (2017). Antipiracy-Aware IP Chipset Design for CE Devices: A Robust Watermarking Approach [Hardware Matters]. *IEEE Consumer Electronics Magazine*, *6*(2), 118–124. https://doi.org/10.1109/MCE.2016.2640622

Soldati, A., Zanichelli, R., Brugnano, F., & Concari, C. (2016). Implementing discrete PID controllers: Benchmarking manual vs. Automatic generation of embedded code. *IECON Proceedings (Industrial Electronics Conference)*, 178–183. https://doi.org/10.1109/IECON.2016.7793334

Suman, S. K., & Giri, V. K. (2016). Speed control of DC motor using optimization techniques based PID Controller. *Proceedings of 2nd IEEE International Conference on Engineering and Technology, ICETECH 2016*, 581–587. https://doi.org/10.1109/ICETECH.2016.7569318

Tang, C. S., & Veelenturf, L. P. (2019). The strategic role of logistics in the industry 4.0 era. *Transportation Research Part E: Logistics and Transportation Review*, *129*, 1–11. https://doi.org/10.1016/J.TRE.2019.06.004

Urbina, M., Astarloa, A., Lazaro, J., Bidarte, U., Villalta, I., & Rodriguez, M. (2017). Cyber-Physical Production System Gateway Based on a Programmable SoC Platform. *IEEE Access*, *5*, 20408–20417. https://doi.org/10.1109/ACCESS.2017.2757048

Vaidya, S., Ambad, P., & Bhosle, S. (2018). Industry 4.0 - A Glimpse. *Procedia Manufacturing*, *20*, 233–238. https://doi.org/10.1016/j.promfg.2018.02.034

Valente, G., Muttillo, V., Muttillo, M., Barile, G., Leoni, A., Tiberti, W., & Pomante, L. (2019).

SPOF-slave powerlink on FPGA for smart sensors and actuators interfacing for industry 4.0 applications. *Energies*, *12*(9). https://doi.org/10.3390/en12091633

Verma, B., & Padhy, P. K. (2020). Robust Fine Tuning of Optimal PID Controller with Guaranteed Robustness. *IEEE Transactions on Industrial Electronics*, *67*(6), 4911–4920. https://doi.org/10.1109/TIE.2019.2924603

Vilanova, A. R. (2016). *Advances in Industrial Control Model-Reference Robust Tuning of PID Controllers*.

Wang, S., Wan, J., Li, D., & Zhang, C. (2016). Implementing Smart Factory of Industrie 4.0: An Outlook: *Http://Dx.Doi.Org/10.1155/2016/3159805*. https://doi.org/10.1155/2016/3159805

XILINX. (2012). *System Generator for DSP User Guide*. *UG639(v 13*, 2012.

Xu, L. Da, Xu, E. L., & Li, L. (2018a). Industry 4.0: State of the art and future trends. *International Journal of Production Research*, *56*(8), 2941–2962. https://doi.org/10.1080/00207543.2018.1444806

Xu, L. Da, Xu, E. L., & Li, L. (2018b). Industry 4.0: State of the art and future trends. *International Journal of Production Research*, *56*(8), 2941–2962. https://doi.org/10.1080/00207543.2018.1444806

Yang, X., Yang, C., Peng, T., Chen, Z., Liu, B., & Gui, W. (2018). Hardware-in-the-Loop Fault Injection for Traction Control System. *IEEE Journal of Emerging and Selected Topics in Power Electronics*, *6*(2), 696–706. https://doi.org/10.1109/JESTPE.2018.2794339

Youness, H., Moness, M., & Khaled, M. (2014). MPSoCs and multicore microcontrollers for embedded PID control: A detailed study. *IEEE Transactions on Industrial Informatics*, *10*(4), 2122–2134. https://doi.org/10.1109/TII.2014.2355036

Yudi, J., Humberto Llanos, C., & Huebner, M. (2017). System-level design space identification for Many-Core Vision Processors. *Microprocessors and Microsystems*,

52, 2–22. https://doi.org/10.1016/j.micpro.2017.05.013

Zezulka, F., Marcon, P., Vesely, I., & Sajdl, O. (2016). Industry 4.0 – An Introduction in the phenomenon. *IFAC-PapersOnLine*, *49*(25), 8–12. https://doi.org/10.1016/j.ifacol.2016.12.002