Contenido

1. Estado del Arte

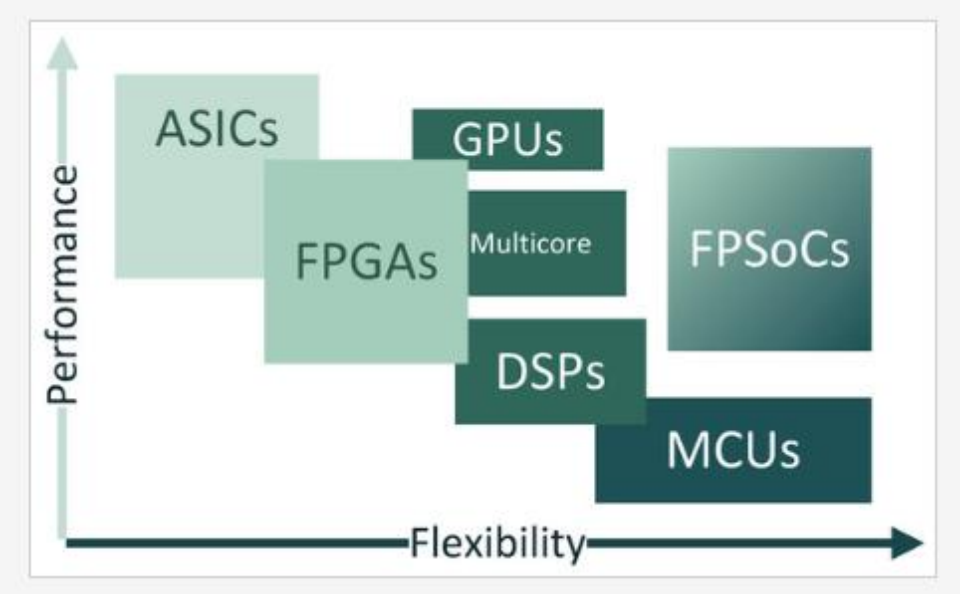
1.1. Introducción

En este estado de la cuestión se ha buscado plasmar aquellos conceptos que permitan tener unas nociones básicas del campo y el entorno en el cual se encuentra involucrado este TFG.

Es por ello por lo que se han condensado en los 3 siguientes puntos aquella información que permita como ya se ha mencionado dar una perspectiva actual en la industria de la temática del TFG mismo.

1.2. SOCs

En la evolución sufrida en las últimas décadas dentro del desarrollo de dispositivos electrónicos, ha dado lugar a la aparición de multitud de estrategias a la hora de abordar y resolver los desafíos que presenta el mercado electrónico actual.
Dentro del desarrollo de la electrónica digital concretamente los circuitos integrados “IC” aparecen conceptos como ASIC, FPGA, SoCs , FPSoCs, ASSP, etc.. , tejiendo una red de dispositivos, propiedades y aplicaciones altamente interconectadas de forma híbrida entre sí, los cuales no permiten discernir correctamente unos conceptos de otros.



Como se puede observar en la siguiente imagen se aprecian una gran variedad de circuitos integrados a cada uno de ellos con sus características particulares frente a otros

1.2.1. Concepto de SOC

Los sistemas en chip (SoC) han surgido en las últimas décadas como una clase importante de sistemas VLSI. Un SOC es un sistema que incorpora la mayoría o todos los componentes necesarios para una aplicación y que utiliza procesadores programables como componentes del sistema.[1]
Otra posible definición de SOC es como el bloque funcional que tiene la mayor parte de la funcionalidad del sistema, excepto algunos bloques de interfaz, que no son realizables por las tecnologías CMOS o compatibles con CMOS.[2]

Los dominios de aplicación afectan de forma sustancial a los periféricos hardware, los tamaños de memorias empleadas y la naturaleza de los buses de comunicación empleados.
En los sistemas SoCs la estructura genérica que se puede encontrar en todos ellos es la

siguiente.

- Processor Cores

Los núcleos (core) son las unidades más pequeñas de un procesador capaces de ejecutar instrucciones e interactuar con otros bloques funcionales de sistema.
La mayoría de los System on Chip, sus procesadores se caracterizan por ser mono o multinúcleo.
Dentro de los procesadores multinúcleos estos arrojan una problemática interesante desde el punto de vista del software debido a la necesidad de coordinación entre sus tareas individuales y colectivas.
De ellos resultan dos tipos de arquitecturas, la AMP y la SMP.
La arquitectura AMP (asymetric multiprocesing), en ella se reparte el software de forma específica para cada core. Estos últimos funcionan de forma independiente con su propia memoria y espacio de memoria reservado de forma exclusiva.
Cada procesador tiene sus propias interrupciones y periféricos.
La forma de comunicar entre Cores será a través de interrupciones o memoria compartida.

Para la arquitectura SMP (Symmetric multiprocessing), en este caso el sistema operativo es compartido por todos los núcleos. El OS determina cual es el núcleo más apto para realizar un trabajo concreto, esto implica que todos sean genéricos. En este formato de estructura la memoria es una zona común a todos los cores, ya que en función de la carga de estos puede requerir el sistema operativo su acceso por uno u otro.
Esta arquitectura es muy común para realizar trabajos genéricos y cuando existen necesidades de cálculo.[2]

- Memorias Embebidas.

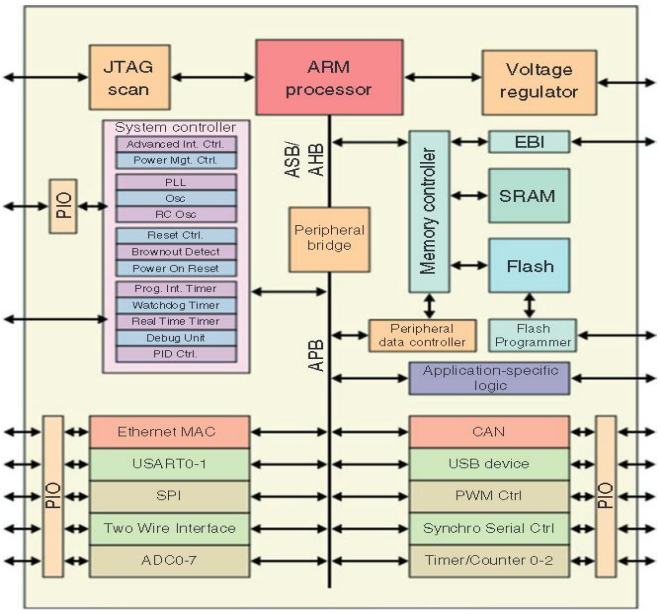
Las memorias integradas se realizan a partir de macros duras para las cuales están disponibles numerosas configuraciones posibles para cada tecnología concreta.
Los vendedores pueden ofrecer herramientas de selección llamados compiladores de memoria.
Entre las opciones habituales disponibles están el número de palabras disponibles, el ancho de bits, el número de sub-bancos o el grado de multiplexación de columnas.
Las pequeñas memorias por otro lado, son diseñadas utilizando flip-flops y registros.

- Cores Analógicos

Los cores analógicos más comunes son amplificadores operaciones, amplificadores de potencia, serializadores, deserializadores, PLL y bloques de circuitos integrados de señales mixtas.

- Interfaz Cores.

Un elemento esencial dentro de los SoCs son las interfaces y bloques de comunicación.
Distintos buses como UART, SPI,AXI o AHB que pueden estar presentes varios a la vez, permiten la comunicación entre los distintos cores del chip.[2]



1.2.1.1. Flujo de diseño e IPs

En las pasadas dos décadas la reutilización de IPs se ha extendido de forma masiva en la industria. Los IP (Intelectual Property) se refieren a la inclusión en el diseño de componentes previamente diseñados y testeados.

A partir del momento en el cual se han empezado a incluir, estos han potenciado todo el ciclo de vida de los circuitos integrados pasando por el diseño, fabricación, ensamblado y distribución.
En otras palabras, se han convertido en una herramienta muy potente a la hora de desarrollar SoCs en la industria mejorando los rendimientos, costes y los tiempos de mercado.

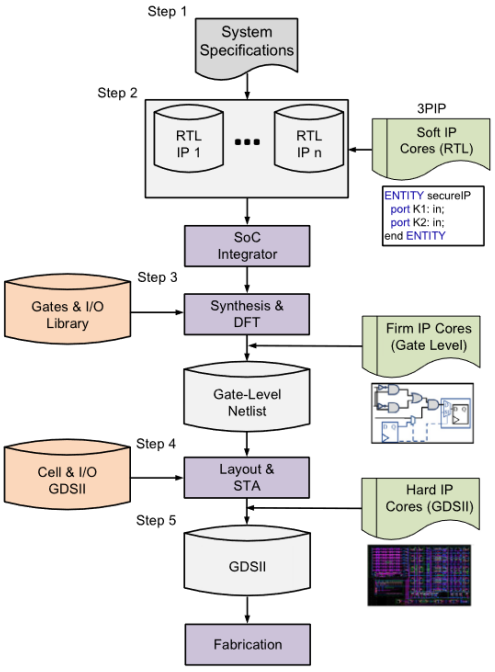
Dentro de los IPs disponibles a incorporar en un SoC se diferencian en función del formato y punto de incorporación al diseño. En función de su flexibilidad y portabilidad se pueden destacar los siguientes tres tipos:[3]

- Hard Cores: Cuando los cores van a ser utilizados en sistemas diferentes, debe ser posibles adaptarlos sin realizar ningún cambio en ellos (o casi ninguno).
  Son, por lo tanto, muy poco flexibles pero muy predecibles y fiables a la hora de implementarlos. Incluyen información del place and route. Un ejemplo de estos tipos de cores son procesadores y memorias.

- Firm Cores: Se encuentran a mitad de camino entre los hard cores y los soft cores, son bloques configurables pero que también contienen algo de información del place and route. Tienen una flexibilidad limitada y una predictibilidad aceptable en la implementación.

- Soft IP: Estos IPs suelen incorporarse en formato de archivos HDL o netlist (listas de puertas lógicas y sus respectivas conexiones). Son altamente flexibles, pudiéndose personalizar de forma específica para cada aplicación. Sin embargo, ese grado de flexibilidad y formato los hacen poco predecibles en la implementación.

La influencia de estos tres tipos de IPs durante el flujo de diseño se puede apreciar en la siguiente imagen.



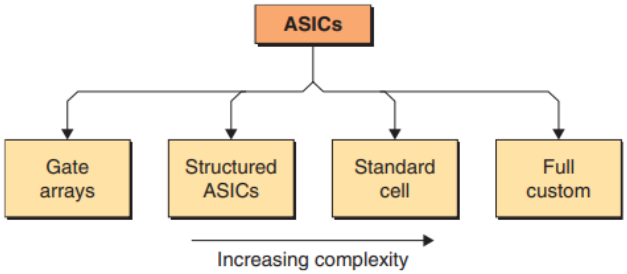
Durante el segundo paso diseño, después de desarrollar/requerir todos los Soft IPs estos son integrados con el fin de generar la descripción RTL de todo el sistema.
Tras la posterior síntesis el integrador de SoC genera a partir de la descripción RTL una lista de red (gate-level netlist) basada en los bloques lógica y de entrada/salida. En este momento el integrador puede añadir los IPs con información del place and route (Firm IP Cores).
Durante la fase de implementación la netlist sintetizada es traducida a una distribución física basada en las propiedades físicas de los bloques lógicos y de entrada/salida. A su vez en este momento el diseñador puede optar por importar un IP de carácter Hard Core. Este IP es proporcionado habitualmente en formato GDSII o su sucesor OASIS.
Estos formatos incorporan información de los datos del IP como las formas, ubicación y propiedad de los elementos que lo integran.
Finalmente se realizan pruebas al diseño entre las cuales se encuentran el análisis temporal y la tasa de consumo del sistema. Una vez concluido satisfactoriamente se genera un archivo en formato GDSII u similar del layout final con el fin realizar su posterior fabricación en la fundición. [4]

1.2.2. Tecnología de Fabricación

Las tecnologías en las que se basan los SoCs actuales son las siguientes: ASICs, ASSPs y FPGAs

- ASIC

Application-Specific Integrated Circuit es un dispositivo cuyo propósito es satisfacer los requisitos para una aplicación particular.
Una clasificación aceptada de este dispositivo lo divide en 4 grandes categorías: (GA) Gate Arrays, Structured ASICs, Standard Cell (SC) y los Full Custom.[5]



En el caso del Full Custom el diseñador tiene el control entero sobre cada capa del chip de silicio fabricado. El nivel de control es alto pudiendo, por ejemplo, alterar las dimensiones de los elementos lógicos individuales. Si por ejemplo se necesitan unas prestaciones temporales sobre una puerta lógica el ingeniero puede modificar sus dimensiones con el fin de obtener dichas consideraciones.
Estos dispositivos pueden incluir parte analógica como comparadores, filtros, convertidores A-D o D-A.
El diseño de estos dispositivos es altamente complejo y requiere de tiempo sin embargo estos concentran la tasa más alta de lógica con el mínimo consumo de energía posible resultando en una reducción de desperdicio en el silicio.

Otro forma de realizar los circuitos es mediante Gate Arrays , los fabricantes como su nombre lo indica generan una malla de resistencias y transistores sin conectar los cuales se recogen en células “Basic Cells” organizadas por capas.
Estos dispositivos son configurados posteriormente en fábrica realizando las conexiones pertinentes entre las distintas celdas según el diseño escogido.

Los Structured ASICs, las capas de máscara lógica son definidas por el proveedor o terceros. La personalización se realiza mediante la creación de capas de metal personalizadas que realizan las conexiones entre los elementos inferiores de capas predefinidos. Este modelo se encuentra a medio camino entre los Gate Arrays y los Standard Cell ya que en este caso solo se produce una pequeña cantidad de capas del chip dando por ello lugar a gastos no recurrentes mucho más pequeños que los dispositivos de celda estándar

Los dispositivos Standard cell tienen muchas semejanzas con las Gate Arrays, a su vez el proveedor ofrece la lógica disponible recogida en los Basic Cells, sin embargo los proveedores de estas nuevas células también suministran bibliotecas de macros duras y blandas que

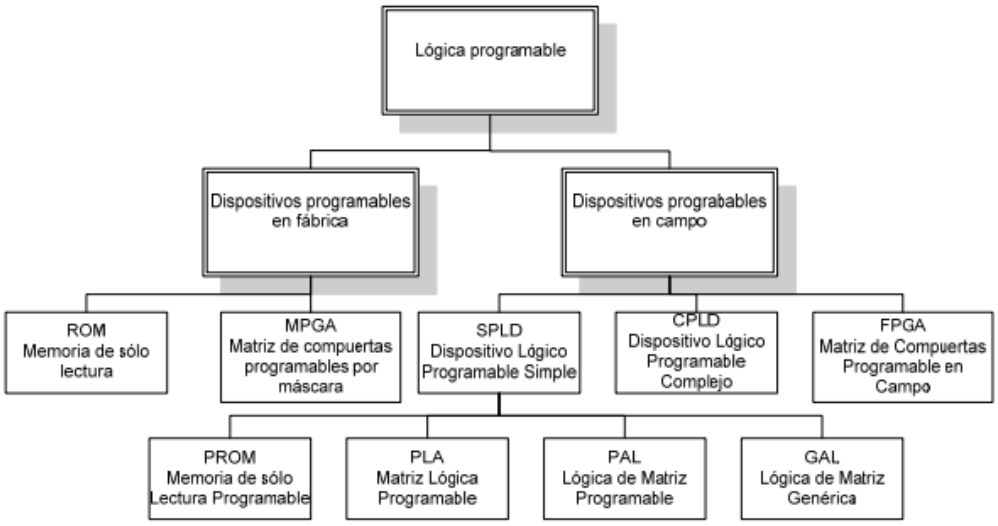
incluyen procesadores, controladores y funciones de comunicación. También suelen incluir una selección de RAM y ROM. Finalmente una particularidad muy importante es la capacidad de los ingenieros en utilizar diseños previamente elaborados o comprar bloques de propiedad intelectual “IP” que se pueden añadir al diseño.

- ASSP

Los ASIC son un componente diseñado y utilizado por una empresa única para un sistema específico.
Sin embargo los Application-Specific Standard Product son de un uso más general, efectivamente se crean utilizando las herramientas y tecnologías ASIC, pero a diferencia de estos primeros son utilizados para un uso más general para múltiples empresas de diseño de sistemas.

- Circuitos Lógicos Programables y FPGAS

Los PLDs reprogramables son componentes electrónicos utilizados para construir circuitos electrónicos digitales reconfigurables. A diferencia de los ASICs estos salen de fábrica sin ninguna función específica y necesitan de ser programados o reconfigurados para poder ser usados.

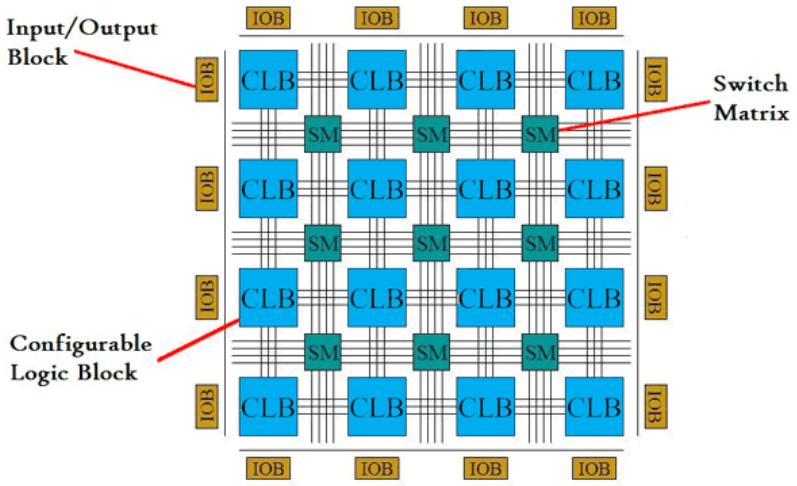


Los PLDs programables en campo pueden dividirse en los siguientes grupos

Los PLA, PAL, CPLD y SPLD sus recursos elementales realizan las operaciones lógicas más sencillas del mismo tipo y suelen combinarse en matrices.Por otro lado los FPGA, sus recursos elementales tienen cierta universalidad pueden realizar operaciones lógicas arbitrarias relativamente complejas, así como el almacenamiento de datos.

Las FPGAs o Field Programable Gate Array son circuitos integrados que contienen un arreglo de bloques lógicos con interconexiones programables a su vez.
El usuario tiene la capacidad de programar tanto las funciones realizadas por cada bloque lógico como las conexiones entre los bloques.

En su interior se distinguen tres elementos básicos, Bloques Lógicos, Bloques de Entrada/Salida y las Interconexiones.



- Bloque Lógicos: Dependiendo del fabricante y el modelo de FPGA estos pueden incluir desde LuTs, multiplexores, compuertas lógicas o flip-flops, pudiendo llegar alguno de ellos a solo contener pares de transistores.
- Las interconexiones programables entre los distintos bloques lógicos pueden variar a su vez dependiendo del modelo de FPGA utilizado destacando dos tipos:
  Las matrices de switches que se encuentran entre cada intersección de las rutas entre bloques, estas soportan cualquier configuración en su conexión sin embargo es una tecnología costosa con el inconveniente de que no se pueden realizar todas las conexiones posibles en un mismo tiempo.

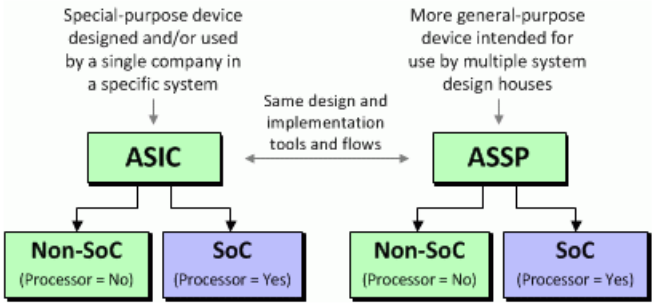
  Conexión a Bloques Lógicos adyacentes. A diferencia de las conexiones con las matrices de switches estas últimas son más rápidas al no tener que pasar por matrices de ruteo.
  Las conexiones generalmente se dan con los cuatro bloques adyacentes al bloque lógico, aunque en algunos casos se pueden conectar a ocho bloques incluyendo sus bloques en diagonal.
- Los Bloques de entrada/salida permiten conectar la lógica interna de la FPGA al medio externo.
  Estos bloques pueden ser configurados como entradas, salidas y entrada/salida gracias a un buffer triestado.
  A su vez contienen flip-flops que permiten almacenar el valor de entrada o de salida.

Los FPGA más modernos incluyen además de estos bloques básicos otros recursos como DSPs los cuales permiten realizar operaciones de suma, multiplicación, multiplicaciones acumuladas

y otras. Bloques de memoria comúnmente llamados BRAM, unidades dedicadas a la gestión de relojes, transceptores de alta velocidad y núcleos IPs que permiten concluir infinidad de tareas.

- Comparativa

Como se puede apreciar el concepto de SoC es amplio, en algunos casos no se puede apreciar la diferencia entre dispositivos ASIC ,ASSP y los SoCs basados en esa tecnología.
Como regla general se consideran como SoCs a las ASIC y ASSP cuando estas últimas incluyen uno o varios procesadores.[6].



Realizar una comparación entre los principales dispositivos basados en silicio en función de su consumo energético, rendimiento y costes es una tarea compleja. En el caso de agruparlas en tres grupos. ASICs/ASSPs , SoCs y FPGAs para el caso de una aplicación concreta comparando entre ellas las implicaciones de consumo, rendimiento, flexibilidad y área se podrían estimar de la siguiente forma.[7]

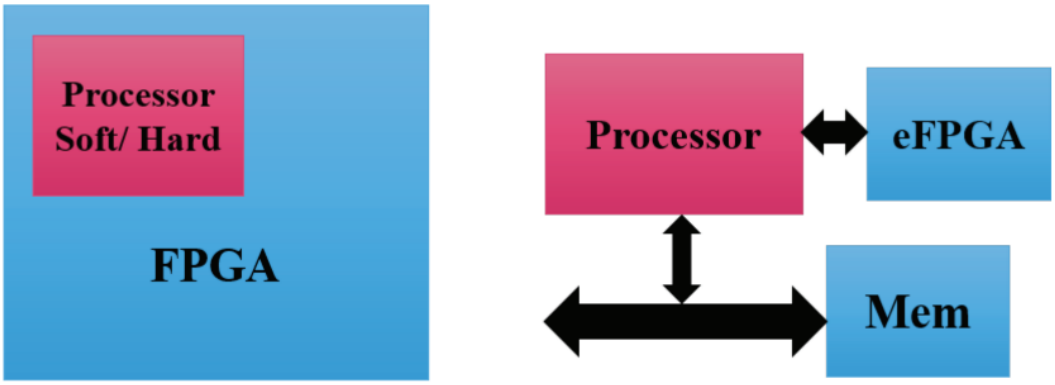
Feature	SoC	ASSP/ASIC	FPGA
Power	Medium	Low	High
Performance	Low	High	High
Flexibility	High (SW)	None	High (HW)
Die Size	Medium	Small	Large

1.2.3. SOCs en FPGA

En las recientes décadas, la industria del semiconductor lleva realizando a crecimiento continuo dispositivos de mayor rendimiento y mayor eficiencia energética.
Con la desaceleración de la ley de Moore, esto ha provocado que los desarrolladores de

sistemas deban esperar para obtener nuevas generaciones de dispositivos con mayores prestaciones.
Este cambio de tendencia supuso la apertura a un nuevo de paradigma y enfoque en el desarrollo de dispositivos, orientando estos a los hardware reconfigurables (mayoritariamente basados en tecnología FPGA).
A pesar de la variedad de arquitecturas en el mercado basadas en software y hardware presentes en el diseño de sistemas embebidos; tenemos por un lado las ASICs que proveen los mayores rendimientos actuales y en contrapuesto los Microcontroladores con soluciones de mayor flexibilidad.
Por el contrario las FPGAs se sitúan en un término medio ofreciendo lo mejor de ambas partes. Sus capacidades de rendimiento y procesamiento son menores que las de una ASIC, pero si superan las de un MCU. Su flexibilidad es mayor que una ASIC gracias a los métodos de reconfiguración dinámica parcial o total pero menor que las de un MCU.[8]

Con este giro de tendencia y la alta variabilidad de diseños presentes en el mercado se pueden dividir en dos grandes grupos: Los FPSoCs y los eFPGA SoCs.



1.2.3.1. FPSoCs

Los principales vendedores de FPGAs han empezado a integrar procesadores duros y blandos (hard/soft) dentro de sus productos. Los FPSoCs “Field-Programmable Systems-on-Chip” emergen como una de las mejores soluciones a medio camino entre rendimiento y flexibilidad. Gracias a ello las FPGAs han dejado de considerarse meros aceleradores hardware para pasar a ser plataformas SoCs de gran potencial.
Intel FPGA, Xilinx AMD, Microsemi, QuickLogic o Lattice son algunas de las compañías en apostar por la siguiente vía dentro de los SoCs reconfigurables[8].

Company	Family	Application Processor		Real-Time Processor/Microcontroller		Logic Cells (K LEs)	Relevant Features
		Architecture	*GHz*	*Architecture*	*GHz*		
Intel FPGA	Arria V SoC	Dual Core ARM Cortex A9	1.05			350 462	AES, SHA IPs Side Channel Prot.
	Arria 10 SoC	Dual Core ARM Cortex A9	1.50			160 1150	AES, SHA IPs Side Channel Prot. Secure Boot
	Agilex SoC	Quad Core ARM Cortex A53	1.40			3000	PUF AES, SHA IPs Side Channel Prot. Secure Boot
	Stratix 10 SoC	Quad Core ARM Cortex A53	1.50			378 2753	
Xilinx	Zynq 7000	Single/Dual Core ARM Cortex A9	0.76 1.00			23 444	PUF Side Channel Prot. AES, SHA, RSA IPs Secure Boot ARM TrustZone
	Zynq UltraScale+	Dual/Quad Core ARM Cortex A53	1.30 1.50	Dual Core ARM Cortex R5	0.53 0.60	103 1143	
Microsemi	SmartFusion			Single Core ARM Cortex M3	0.10	2 6	Side Channel Prot. AES, SHA, RSA IPs
	SmartFusion 2			Single Core ARM Cortex M3	0.16	5 150	PUF Side Channel Prot. AES, SHA, RSA IPs Secure Boot
				RISC V (S/C) 8051 (S/C)			
	IGLOO2			RISC V (S/C) 8051 (S/C)			
	PolarFire			RISC V (SC) ARM Cortex M1 (S/C)		300	Secure Boot
QuickLogic	S3			Single Core ARM Cortex M4 F	0.08	2.4	Power Manag. Unit
Lattice	ECP5			RISC V (S/C)		12 84	AES IPs Neural Network IPs
	iCE40 UltraPlus			RISC V (S/C)		2.8 5.28	

Los recursos hardware en las FPGAs varían de forma muy notoria según el fabricante y el modelo del dispositivo.
Además de los recursos habituales presentes en ellas bloques lógicos, bloques de entrada/salida, interconexiones, etc . Existen en estos nuevos productos orientados a los SoCs ,prestaciones no presentes en antiguas generaciones pero que las presentes si incluyen.

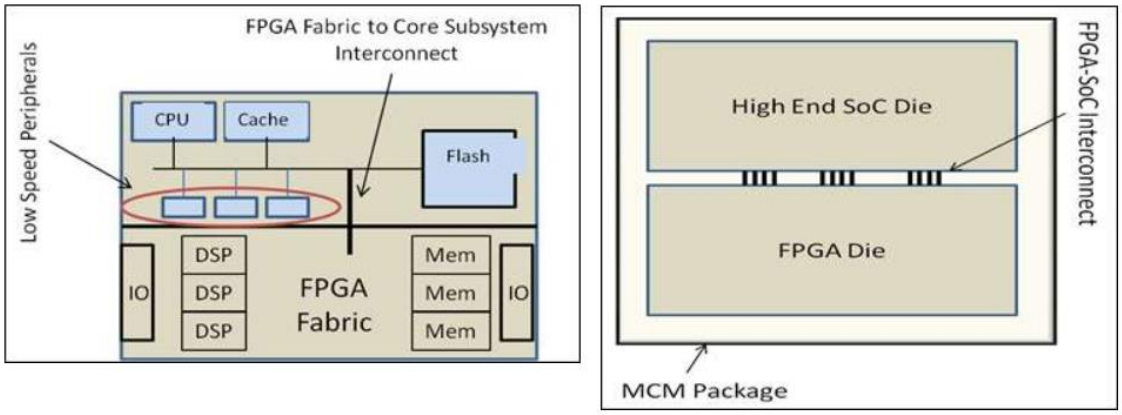
Es el caso para los recursos analógicos, las antiguas FPGAs se encontraban carentes de dichos recursos en particular convertidores analógico-digital y digital-analógico (ADCs y DACs).

Algunos de los principales recursos analógicos nuevos en las FPGAs actuales son los siguientes:[9]

- Los bucles de bloqueo de fase fraccionados (fPLL): Procedentes de Altera permiten implementar la síntesis de frecuencia fraccionada. Estos se consiguen gracias a un modulador delta sigma presente en el bucle de realimentación el cual puede ser controlado por señales externas, lo que le añade mayor flexibilidad.
  Esta estructura mejora el rendimiento de los osciladores de referencia para las comunicaciones digitales. En muchos sistemas industriales actuales se requieren interfaces de comunicación fiables y de alta velocidad.
- Convertidores analógicos digitales: Las series 7 de Xilinx incluyen tecnología AMS (Analog Mixed Signal) que permiten la conversión y configuración de señales a través de los bloques XADC, incluyen 2 ADC de 12 bits y 1Msps, dos amplificadores de seguimiento y retención que permiten hacer muestreo diferencial y multiplexadores que facilitan el acceso a 17 canales.
- Motor de cálculo analógico (ACE): Las FPGAs Microsemi SmartFusion incluyen un bloque denominado ACE, capaz de controlar hasta tres ADCs de 12 bits y 600 ksps con muestreo y retención, así como tres DACs de 24 bits

- Procesadores Hardcore Vs Softcore

Los procesadores embebidos o hardcore principalmente procesadores ARM (Intel y Xilinx) o en el caso de Microsemi, algunos de sus dispositivos incluyen algunos Cortex-M3.
La adición de estos dispositivos a las FPGAs ha llevado al desarrollo de diferentes arquitecturas, actualmente existen dos implementaciones populares que las empresas están siguiendo para producir estos dispositivos [7].



La primera opción (izquierda) consiste en desarrollar un dispositivo SoC-FPGA integrado en una sola matriz con el dispositivo dividido en una parte de FPGA y una parte de procesador más el subsistema del procesador.
La segunda opción (derecha) consiste en colocar el SoC y la FPGA uno al lado del otro como dispositivos discretos, conectarlos y alojarlos en un único paquete MCM (Módulo multichip).
Tanto en los dispositivos de Intel-Altera y Xilinx, el procesador y la parte reconfigurable de la FPGA son alimentados de forma independiente, es posible desactivar uno u otro con el fin de reducir el consumo.[9]

FPGA Vendor	Hard Processor	Soft Processor
Actel	none	third-party only
Altera	ARM	NIOS, NIOS II
Atmel	AVR	third-party only
Lattice	none	third-party only
Quicklogic	MIPS	third-party only
Xilinx	IBM PowerPC	MicroBlaze, PicoBlaze

Además de los procesadores duros algunos fabricantes de FPGA proveen a su vez procesadores blandos (softcore). Es el caso de Xilinx con el microblaze o picoblaze e Intel-Altera con el NIOS II. Existen a su vez multitud de procesadores desarrollados por terceros y disponibles para incorporar según los requisitos del proyecto.

¿Qué es mejor, un procesador softcore o un procesador embebido hardcore? La respuesta, no es directa. Para poder elegir entre un modelo de core u otro se deben analizar los requisitos de

diseño que se quiere implementar. ¿Cuáles son las prestaciones que más interesan en él diseño? Velocidad, recursos ocupados y consumo son los factores que determinan la elección.

Los procesadores duros superan ampliamente en términos de velocidad sus homólogos blandos al no estar estos primeros limitados por las características físicas de la FPGA. El procesador duro existe como un chip único integrado dentro de la FPGA esto se ve reflejado en el uso de espacio. Los procesadores blandos utilizan los recursos físicos de la FPGA para poder ser desplegados. Sin embargo, las altas prestaciones en velocidad de los hardcores se resienten en un mayor consumo energético. Además, como contrapartida al estar integrado en un chip físico dentro de la FGPA su flexibilidad en los diseños es muy baja siendo los procesadores softcore más fácilmente adaptables a modificaciones.[10]

	Soft-core	Hard-core
Power consumption	✓	×
Speed	×	✓
Resource utilization	×	✓
Flexibility for design	✓	×

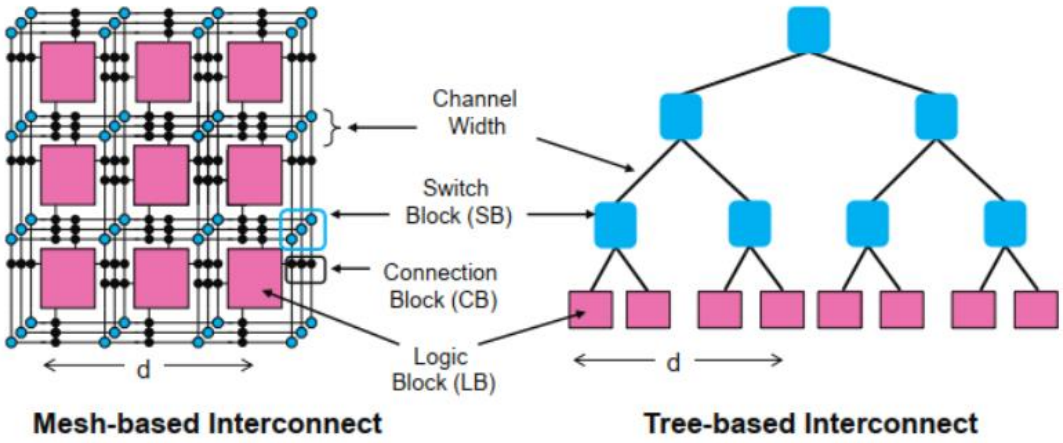
1.2.3.2. eFPGAs en SoCs

Una FPGA embebida (eFPGA) es un núcleo IP que puede integrarse dentro de un dispositivo ASIC o SoC para obtener los beneficios de su lógica programable con mejora de latencia, rendimientos y consumo energético.

Como ya sea ha mencionado las FPGAs son conocidas por su flexibilidad y sencillez en los cambios de diseño. Con los continuos avances y la regla de Moore estos dispositivos se han convertido en una alternativa para el despliegue de SoCs directamente sobre ellas.
Sin embargo, las FPGAs tiene unas características de “silicio” en término de área, potencia y velocidad con respecto a las ASICs.
Efectivamente el impacto positivo de las cualidades de una FPGA se ven afectado por algunas desventajas del uso de dicha tecnología que son las medidas punitivas de rendimiento y sobrecarga de área [11]. Se puede considerar que una FPGA consume de media 12 veces más potencia dinámica que una ASIC equivalente, es unas 40 veces más grande en área y unas 3,2 veces más lenta comparada con las ASICs [12] .
A esta situación se le suma una problemática inherente que afecta de forma sustancial al coste de desarrollo de SoCs. La detección de errores de diseño después de la fabricación. Estos errores pueden deberse a fallos no encontrados durante el proceso de simulación o cambios en los requisitos y especificaciones de diseño.[13]

Por lo tanto, la solución adoptada por los desarrolladores es la incorporación de núcleos lógicos programables basados en la tecnología FPGA en forma de IP embebidos en el SoC que permitan dotar de cierta flexibilidad sus diseños y solventar posibles errores o cambios en el diseño después de su fabricación.

La dos arquitecturas y topologías más populares de las FPGAs y que por lo tanto son la base de las eFPGAs son: Mesh-based Interconnect y Tree-based Interconnect.

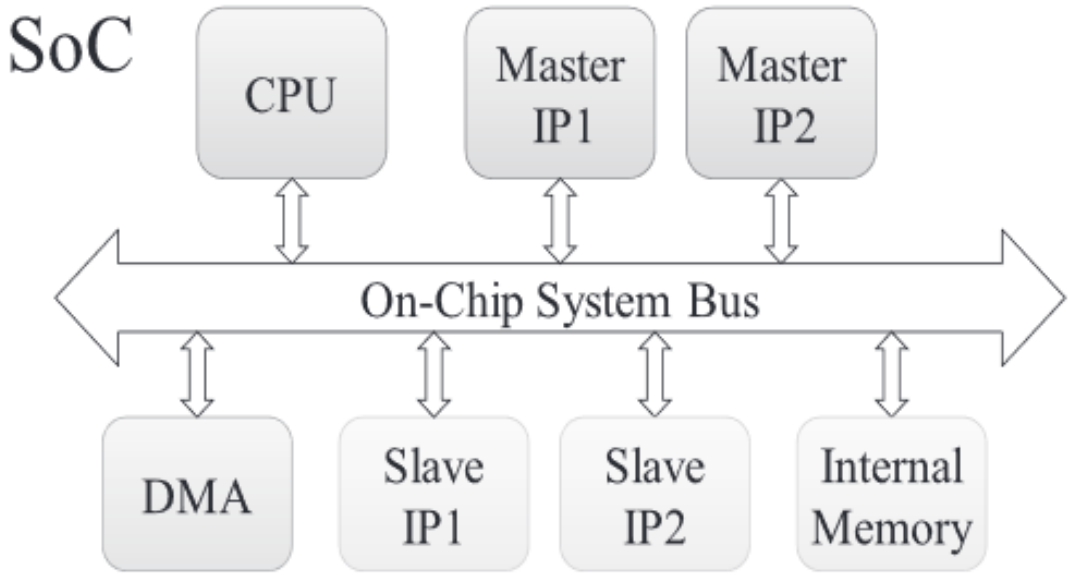


Las arquitecturas Mesh-based Interconect (basadas en malla), los LBs se colocan en una cuadrícula 2D y están rodeados de canales de enrutamiento, las interconexiones de enrutamiento están formadas por bloques de conmutación (CBs) y segmentos de cable (SBs). Los bloques CB conectan los bloques lógicos con los canales adyacentes. Los bloques SBs controlan las conexiones entre los enrutados verticales y horizontales.

Por otro lado, la arquitectura Tree-based Interconnect (basadas en árbol o también llamadas jerárquicas multinivel), en este caso se organizan los bloques lógicos en clusters separados y conectados de forma recursiva para formar una estructura de apariencia jerárquica. Solamente las arquitecturas con más de 2 niveles de jerarquía se consideran jerarquías multinivel.
Cada estructura tiene sus ventajas. Los modelos basados en estructura jerárquica están diseñados para maximizar el uso de la lógica, explotando la ubicación de las conexiones, otorgando mayor rendimiento y prestaciones temporales. En la estructura basada en árbol, el número de interruptores en serie utilizados para conectar dos LBs aumenta como una función logarítmica de la distancia d de Manhattan mientras que el número de segmentos en serie aumenta linealmente con d para la estructura de malla.
Sin embargo, el retardo en cables aumenta exponencialmente a mayores niveles jerárquicos empleados, por lo tanto, el uso de altos niveles jeráquicos en las FPGAs basadas en esta arquitectura pueden disminuir su rendimiento.
El modelo en malla es más escalable al tener una estructura regular.[11]

1.3. Buses de comunicación On-Chip La arquitectura AMBA.

La entrada de los circuitos integrados en la era de los SoC, los cuales se caracterizan como se ha visto en la integración de una gran cantidad de componentes distintos en un mismo chip; CPUs, variados IPs de bloques Lógicos, Bloques de Memorias, circuitos con complejas técnicas de multiplexado, periféricos, etc [14].



Esto produce que el tamaño del diseño se incremente cada vez más y ya no sea posible realizar el desarrollo entero del proyecto debido a los costes de tiempo y recursos, es por ello por lo que se hace inevitable el incremento del uso de IPs. Gracias a la incorporación de estos bloques se incrementa la velocidad en el diseño.
Sin embargo, surge un problema, cada bloque por separado puede presentar interfaces de comunicación distintas al proceder de fabricantes distintos.

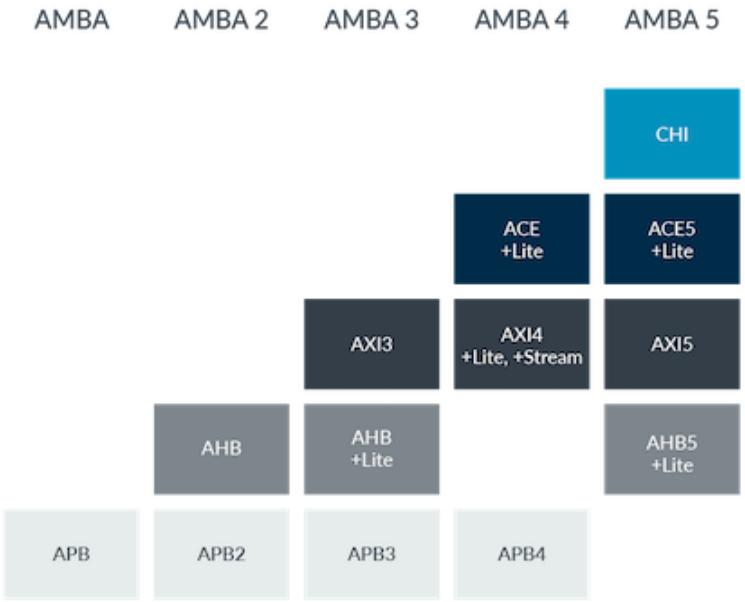
Todo ello resultó en la necesidad de la creación de una arquitectura de comunicación estandarizada que permitiera la integración correcta de distintos IPs en el mismo chip. [15]
Con este propósito han ido surgiendo distintas arquitecturas para regular la comunicación en los diseños Soc.
Algunas de las arquitecturas desarrolladas son: ARM Microcontroller Bus Architecture (AMBA), IBM Core Connect, OpenCores Wishborn , Altera Avalon y SiliconBlackplane de SONY . [16][17]

De todos los buses On-Chip (OCB) que han sido desarrollados en el mercado uno de aquellos que se ha proclamado como un standard de facto es el bus AMBA 4.0 introducido por ARM en 2010.

1.3.1. Arquitectura AMBA

El "*Advanced Microcontroller Bus Architecture*" o bus AMBA introduce 5 buses/interfaces: Advanced eXtensible Interface (AXI),Advanced High Performance Bus (AHB), Advanced Peripheral Bus (APB) y el Advanced Trace Bus (ATB) [18].

Desde su aparición en 1990 han ido sucediéndose numerosas versiones de la arquitectura con el fin de responder a la demanda de nuevos procesadores y tecnologías del mercado llegando hasta nuestros días con la versión 5.0.



Toda la configuración estructural, las señales y la transferencia de módulos se encuentra definida por la especificación AMBA.

Advanced System Bus (ASB).
Es un bus que incorpora la arquitectura Pipeline, soportando múltiples maestros.
Su modo standard de operación se encuentra caracterizado por la figura del árbitro (arbiter) y el decodificador. El árbitro decide que maestro de aquellos que solicitan acceso al bus tiene mayor prioridad. Una vez determinado, el maestro seleccionado inicia la transferencia.
La figura del decodificador selecciona la dirección del bus esclavo y este último realiza una respuesta al bus maestro.
Una vez realizada esta operación, los datos son transmitidos entre maestro y esclavo. [19]

Este bus se caracteriza por tener unas altas prestaciones en el diseño con microcontroladores embebidos de 16 y 32 bits, es por ello por lo que soporta correctamente la conexión a memorias On-chip, procesadores o a su vez con memorias externas.

Esta arquitectura aparecida con la primera versión de AMBA en 1990 ha sido poco a poco remplazada por versiones más modernas de buses dentro de la arquitectura.

Advanced High Performance Bus (AHB)

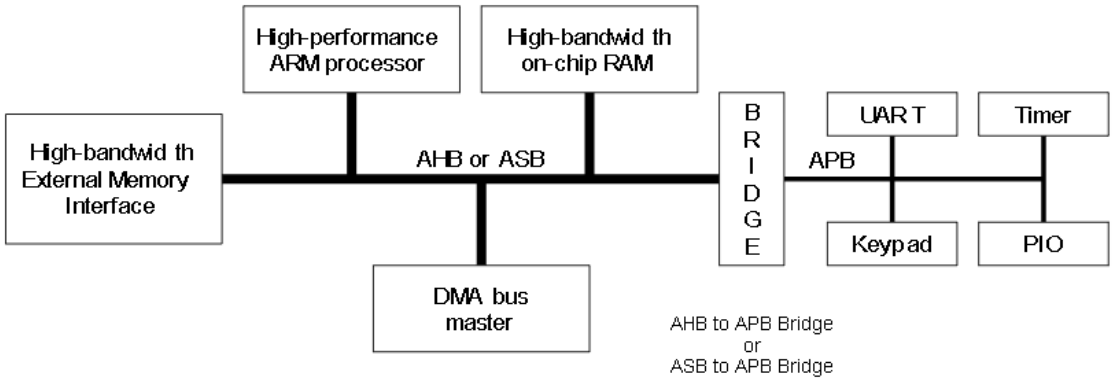
Este bus surge más tarde que el bus ASB e igual que este último es un bus de altas prestaciones incorporando también operaciones de Pipeline.
Soporta múltiples maestros y operaciones con alto ancho de banda.
Este sistema se utiliza típicamente con un procesador y la interfaz de prueba sin embargo, es común añadir como buses maestros accesos directos a memoria (DMA) o algún tipo de procesador digital (DSP).

Advanced Peripheral Bus (APB).

Se ha diseño con el objetivo de minimizar al máximo la complejidad de su interfaz y reducir al mínimo su consumo energético.
Su uso habitual es de interfaz de cualquier periférico que requiere un ancho de banda pequeño.
Este bus sigue usándose hoy en día en gran medida a diferencia del ASB.
Las principales ventajas aportadas de incorporar el bus APB en los diseños son las siguientes:

- Uso en dispositivos de baja velocidad como bus periférico.
- Es síncrono.
- Soporta un solo maestro y no tiene estructura pipeline.

Es típico el uso combinado de varios buses dentro del mismo chip, para ello con el fin de poder sincronizar los datos entre buses se utilizan puentes (bridge) que permiten “traducir” la información de un protocolo a otro.
En la siguiente imagen se muestra una organización típica en un SoC, en el cual podrían verse envueltos dos de los buses mencionados.



Como se ha mencionado anteriormente el bus APB realiza las funciones de bajo consumo con dispositivos periféricos, por otro lado, la arquitectura de comunicación utilizada para conectar el procesador con memorias y DMA se realiza con un bus de mayores prestaciones como podría ser el caso de AHB o el ASB.

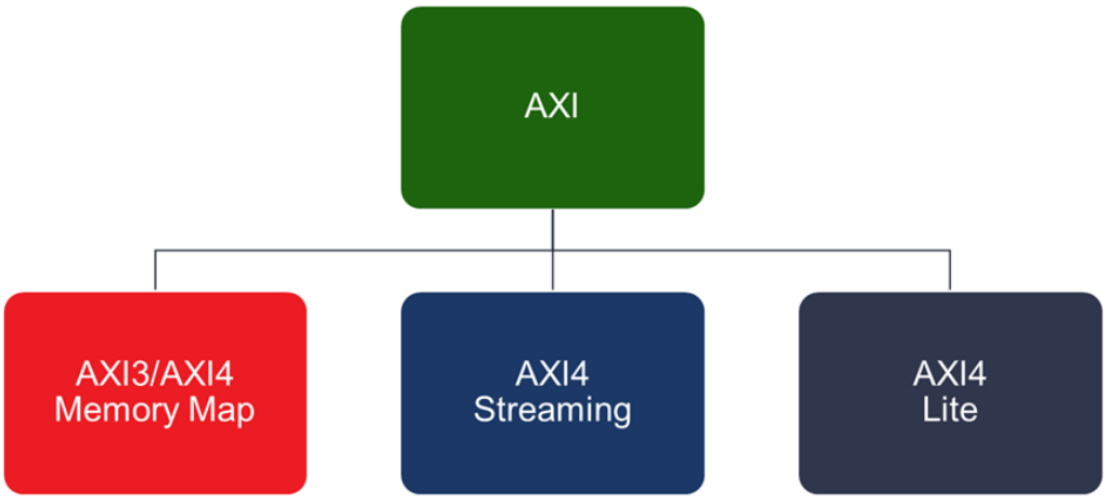
Advanced eXtensible Interface (AXI).

Unos de los buses más extendidos y conocidos de la arquitectura AMBA es el famoso bus AXI. Introducido en la versión 3.0 y ampliado en las sucesivas versiones 4.0 y 5.0. goza de un particular aprecio a la hora utilizarse en el diseño de SoCs.

Estos días prácticamente todos los IPs utilizados en Xilinx hacen uso de esta arquitectura. Los procesadores Zynq, Zynq MP, Microblaze y Versal todos ellos utilizan interfaz AXI.[20]

Como se ha mencionado anteriormente el AHB es un bus de un único canal soportando múltiples maestros y esclavos para el intercambio de información. Con un árbitro seleccionando el maestro más válido y un decodificador jerarquizando la lista de esclavos disponibles.
El bus AXI nace con la misma filosofía sin embargo a diferencia de los anteriores presenta varios canales dedicados a las operaciones de lectura y escritura.[21]
Esta especificación define tres protocolos concretos: [20] [22]

- AXI4: Es una interfaz de direcciones y datos mapeados en memorias de alto rendimiento. Tiene la capacidad de realizar accesos en modo ráfaga (burst) a dispositivos mapeados en memoria. Su uso es habitual para memorias de alto rendimiento.
- AXI4-Lite: Es una derivación simplificada del protocolo AXI4. Su interfaz es más sencilla y carece de capacidad para realizar acceso en modo ráfaga. Su uso típico es para comunicaciones simples asignadas en memorias de bajo rendimiento, en registros de estado y control.
- AXI4-Stream: Es un protocolo unidireccional orientado de maestro a esclavo, reservado para la transmisión de datos a alta velocidad.



1.4. Aplicaciones de los SOCs FPGA en la Industria y aportación a los ODS.

PENDIENTE ACABAR ESTE APARTADO ESTADO DEL ARTE.

2. Entorno de Trabajo

En este apartado se va a abordar el entorno en el cual se han desarrollado los objetivos de este TFG, centrándose en describir las herramientas software que han permitido el desarrollo de las distintas etapas del proceso.

2.1. Herramientas de desarrollo SW/HW Xilinx

Los fabricantes de FPGA ofrecen además de sus dispositivos, herramientas diseñadas y orientadas para la programación y despliegue de diseños en sus propios productos.
Para este proyecto al utilizarse una FPGA de la empresa Xilinx se utilizará el entorno de desarrollo de dicha empresa.

Vivado Design Suite aparece en 2012 como un entorno cuyo propósito era el desarrollo integrado. Recientemente la compañía ha lanzado una nueva plataforma, Vitis, que posteriormente también incluyó Vivado. Ambas herramientas permiten crear diseños que se ejecutan en FPGA.
La aparición de Vitis en escena ofreció la posibilidad por un lado de acercar la programación de FPGAs a mayor cantidad de programadores alejando esta última de los lenguajes de bajo nivel como Verilog y VHDL. Es decir, con sus nuevas funciones de HLS permite la programación de bloques con C/C++ que pueden posteriormente integrarse a bajo nivel con Vivado.
En contrapartida su uso para la programación secuencial en C/C++ de los procesadores presentes en el diseño.

En base a lo mencionado anteriormente, las herramientas de Xilinx utilizadas para este proyecto son las siguientes:

- *Vivado*: Utilizado para la programación en VHDL del hardware y la integración de un SoC incluyendo un procesador softcore.

- *Vivado IP Packager :* Herramienta disponible dentro de Vivado, permite el desarrollo y empaquetado de módulos IPs. Su interfaz es similar a la disponible en Vivado. Sin embargo, incluye la posibilidad de generar una estructura de documentos exportable a otros proyectos en el formato IP.

- *Xilinx Vitis:* Antiguo “Xilinx SDK”, se programa en C/C++ para ejecutar dentro de un procesador en un diseño creado en Vivado. Este código habitualmente acaba, parcialmente, por configurar y controlar ciertos elementos del hardware.

3. Motores sin escobillas (Brushless)

En el siguiente apartado y coincidiendo con uno de los objetivos del TFG, se va a realizar una introducción al funcionamiento de los motores brushless. Con el fin, de transmitir aquellos conceptos necesarios para su control y alimentación.

3.1. Control y alimentación de los motores sin escobillas.

Los motores sin escobillas se caracterizan por tener una alimentación diferente a los motores de CC con escobillas.
Mientras que para estos últimos la excitación del motor se realiza aplicando una diferencia de potencial en sus bornas. Los motores sin escobillas “brushless” tienen la peculiaridad de necesitar un control en la alimentación de sus bobinas con el objetivo de hacer girar su rotor.

4. Desarrollo y creación de un SOC en FPGA

El siguiente apartado y núcleo de este proyecto aborda las fases realizadas desde el diseño hasta su consecución de un SoC en un FPGA con el uso de las herramientas de Xilinx.

4.1. Planificación.

El desarrollo de un IP, su integración en un SoC y posterior programación a alto nivel es una tarea compleja la cual requiere de una planificación minuciosa en el diseño de las distintas fases del proceso. Esto es debido a la combinación de programación Hardware con programación Software. Los errores pueden resultar en cuellos de botella muy grandes durante las fases de pruebas acarreando retrasos altos.

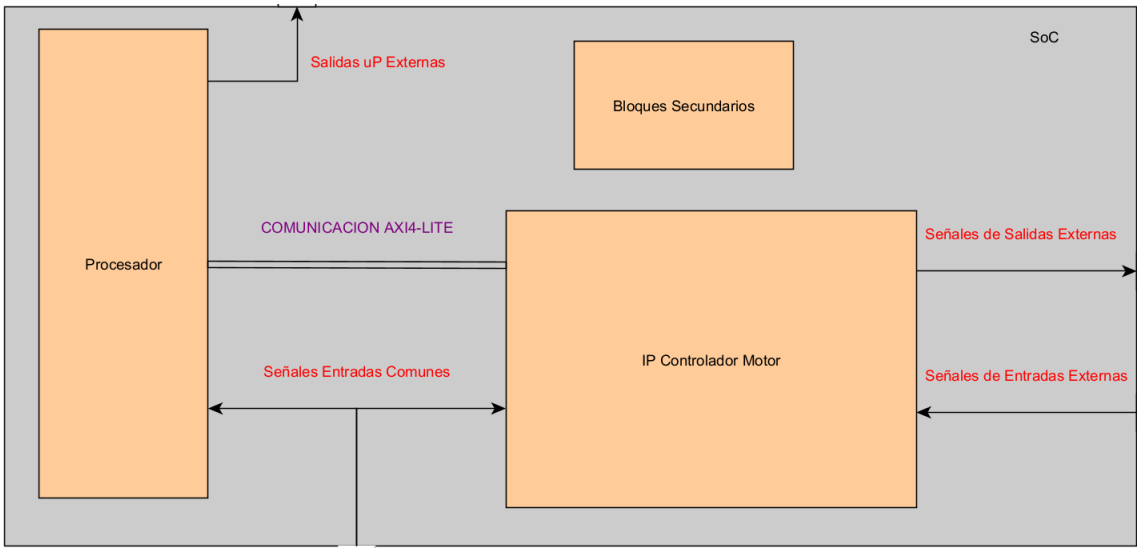
Las fases del proceso desde su inicio hasta consecución son las siguientes.

1. Fase de diseño general.
2. Desarrollo IP
    a. Diseño
    b. Programación Hardware
    c. Integración y empaquetado del hardware en IP
3. Desarrollo e integración del IP en un SoC.
4. Desarrollo Software.
5. Despliegue.

4.2. Fase de diseño general.

El objetivo de esta fase es planificar correctamente cada parte a fin de poder realizar correctamente la integración del sistema.
Esta tarea es compleja debido a la gran cantidad de elementos que intervienen desde la programación individual de módulos, el uso de IPs externos y la alta cantidad de señales.

Cada fase del proyecto debe tener su propia etapa de diseño, sin embargo, con el objetivo de abarcar todo el proyecto el diseño general de este mismo resulta en el siguiente.



Como se puede apreciar la imagen del diseño general está simplificada con el objetivo de no cargar el esquema con elementos menores. Se han seleccionado varios bloques principales, con las entradas y salidas de mayor importancia.

Bloques

- IP Controlador Motor: Este bloque representa el IP que gestionará el control del motor sin escobillas, en su interior se concentra toda la lógica programada en hardware necesaria para generar las señales de control del motor.
- Procesador: Este bloque representa el procesador (Microblaze) embebido en la FPGA.
- Bloques Secundarios: Este bloque representa todos aquellos elementos secundarios necesarios para el correcto funcionamiento de los dos bloques mencionados anteriormente, a fin de simplificar la imagen, como lo son elementos intermedios de comunicación AXI, elementos de memoria y generadores de reloj.

Señales

- Señales de Salidas Externas: Estas señales salen directamente del IP sin interaccionar con demás elementos del SoC son las señales de control del motor además de señales que facilitan tareas de depuración durante las pruebas posteriores.

- Señales de Entradas Externas: Estas señales al igual que las anteriores descritas entran directamente al IP sin interaccionar con elementos del SoC. Son señales de los

sensores HALL del motor. Serán filtradas dentro del IP a fin de aislar este último en mayor medida del exterior.

- Salidas uP Externas: Esta salida principalmente corresponde a la interacción del procesador con el exterior. En este caso será comunicación serial UART.
  Su salida no es directa deberá pasar por elementos como los bloques secundarios.

- Señales Entradas Comunes: Estas señales hacen referencia a la señal de reloj interna de la FPGA y el RESET general. Al igual que las señales de salida del procesador su interacción no es directa con los bloques procesador e IP, sino que deberán pasar a través de bloques secundarios.

Los elementos del SoC generados y la programación sw del procesador dependerán en gran medida de las funcionalidades del bloque IP.

4.3. Desarrollo IP

Este apartado contiene el procedimiento seguido para el desarrollo de un IP desde su concepción inicial, programación, simulación y posterior empaquetado.

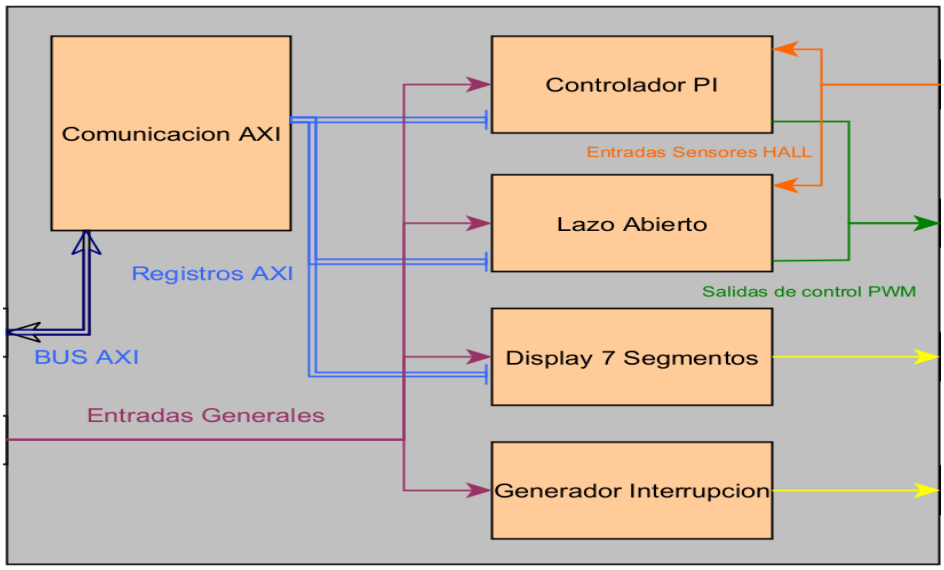
4.3.1. Diseño.

El objetivo principal del IP a desarrollar es el control de la alimentación de un motor sin escobillas.
Las funcionalidades que se definen a continuación van a determinar las características del IP desde sus registros, entradas y salidas.

Funcionalidades:

- 2 modos de funcionamiento:
  - *Modo Controlador PI*: El IP realiza el control del motor sin intervención externa a excepción de configuraciones básicas: Sentido de giro, constantes, Referencia (Velocidad deseada).
  - *Modo Lazo Abierto*: El IP realiza el control de la alimentación del motor, sin realizar control sobre su velocidad. Esto conlleva que se deba proporcionarle el ciclo de trabajo de los PWM para regular la velocidad de forma manual.
- *Salida por 4 display de 7 segmentos*: Se pretende que la velocidad calculada en RPM pueda ser mostrada a través de 4 displays de 7 segmentos.
- *Generador de interrupciones:* Con el objetivo de mejorar las prestaciones del IP es apropiado añadir un módulo de interrupción que permita avisar a otros elementos del sistema el momento en el que se dé una actualización de la velocidad en los sensores.

En base a las funcionalidades concebidas, se definen 4 módulos. Un módulo de control (Controlador PI), un módulo de alimentación (Lazo Abierto), módulo de salida de 4 displays de 7 segmentos y un módulo generador de interrupciones.



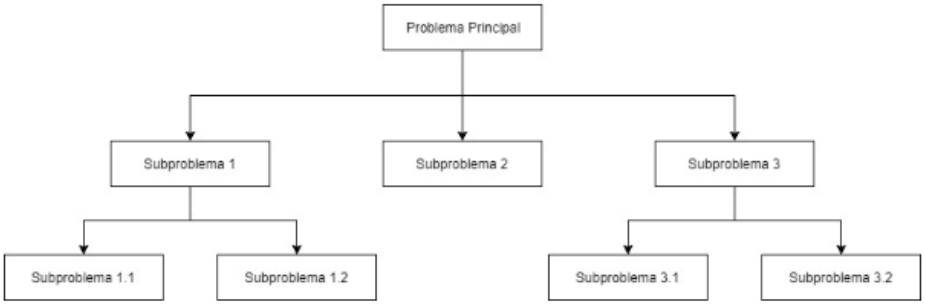
Las entradas generales corresponden a las señales generales de reloj (CLK) y señal de Reset General, se aprecia a su vez que el bus AXI y el módulo de comunicación AXI se comunica con 3 módulos del IP.
El módulo de comunicación AXI y la gestión de sus registros con los demás módulos del IP se gestionará en el apartado de integración del IP.

Quedando definidas las funcionalidades del IP se puede empezar a realizar la programación en VHDL de los 4 módulos concebidos.

4.3.2. Programación Hardware.

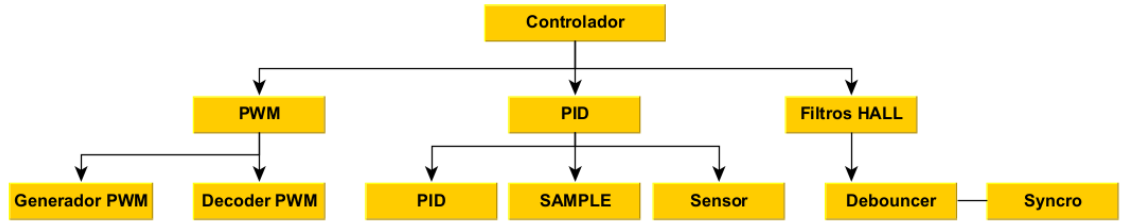
La estrategia empleada para programar los módulos mencionados en el apartado anterior es mediante un acercamiento Top-Down.

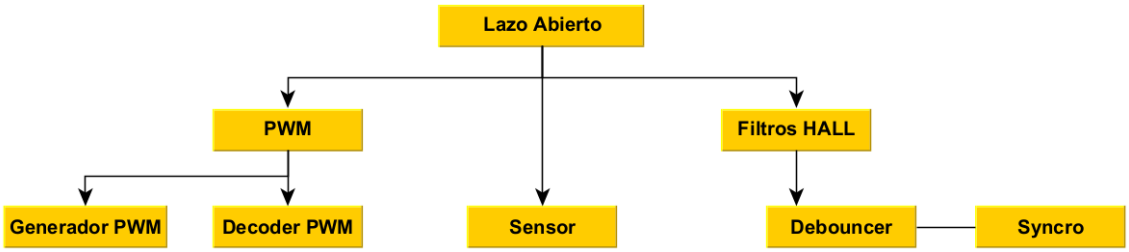
Top-Down: La metodología Top-Down se basa en el paradigma de “Dividir y Vencer” el cual se enfoca en dividir un problema principal en un conjunto de sub-problemas y a su vez dividir estos mismos en sub-problemas.
El uso de esta metodología permite centrarse en tareas individuales para posteriormente interconectarlas entre ellas con el fin de resolver tareas de mayor complejidad.

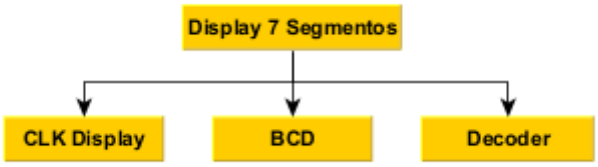


Esta filosofía permite el desarrollo de módulos con alta cohesión con el problema que buscan resolver, pero con alta independencia con respecto a otros módulos. Esto último permite su reutilización en otros proyectos o partes del sistema.

En base a esta metodología el planteamiento de programación de los 4 módulos Controlador, Lazo Abierto, Display e Interrupción es el siguiente.

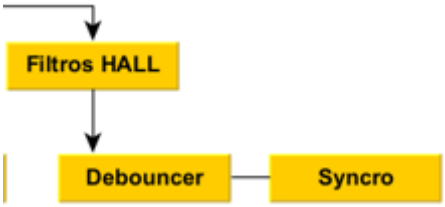






Tanto en el Controlador como en el Lazo Abierto, se puede observar en ellos que se repiten los mismos módulos a excepción del PID y SAMPLE. El enfoque orientado a programación modular va a permitir reutilizar los submódulos programados para ambos diseños, sin realizar ninguna modificación.
A continuación, se van a describir las tareas que realizan cada submódulo junto a sus simulaciones. Esta fase programación y simulación se realiza en Vivado.

4.3.2.1. Filtros HALL

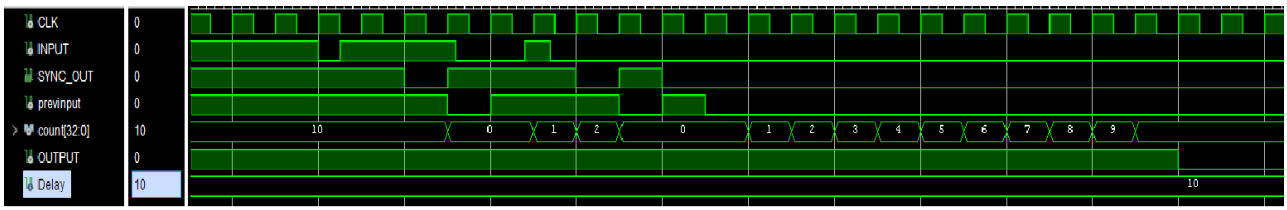


La tarea de este módulo consiste en filtrar las señales asíncronas entrantes procedentes de los sensores HALL.
El submódulo Syncro se ha posicionado en paralelo a la lógica desarrollada para el Debouncer debido a que ambos se encuentran declarados dentro de la misma arquitectura y no son independientes uno de otro.

La tarea del sincronizador permite solventar el problema de las señales asíncronas entrantes. En un sistema síncrono, se asume que la frecuencia de las señales entrantes a los registros flip-flop es menor que la señal de reloj. Sin embargo, cuando las señales entrantes al flip-flop violan su tiempo de setup o hold no se puede predecir correctamente el valor de salida capturado, pudiendo entrar en un estado de meta-estabilidad.
Un método para incrementar la resistencia a estos fallos son los dispositivos llamados sincronizadores. Los más sencillos consisten en crear una cadena de biestables en la cual la salida de un flip-flop es la entrada del siguiente. La longitud de la cadena no es mayor de 2 o 3 biestables.
Un aspecto negativo de esta técnica es que aporta retraso al sistema.

Una vez sincronizada la señal se pasa su valor al submódulo Debouncer (antirebotes). La señal de los sensores HALL al pasar de valor activo a nulo puede sufrir rebotes debido a ruido o el tiempo en el que tarda en estabilizarse, los cuales pueden acarrear mal funcionamiento en el sistema.
Para ello el detector se encarga de fijar mediante un biestable el valor de la entrada y comparar durante x ciclos de reloj. En caso de mantenerse el valor de la entrada estable esta es pasada al registro de salida del filtro en caso contrario se aloja el nuevo valor de entrada y se reinicia la cuenta de ciclos de comprobación.

Este módulo permite configurar los x ciclos de comprobación, estando definido este valor como genérico en su declaración.

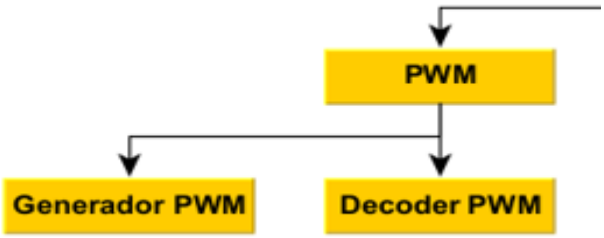


En la siguiente imagen se aprecia como ante una entrada asíncrona, se sincroniza (SYNC_OUT) con 2 ciclos de delay. Por otro lado, gracias al registro (previnput) la entrada previa es comparada con la actual, iniciando el contador en caso de no coincidir. Si el contador no llega a los X ciclos definidos, en este caso 10, se reinicia. Finalmente, la señal de salida cambia al no recibir ningún cambio.

4.3.2.2. PWM

En el apartado XXXXXX se expuso los distintos métodos de control y alimentación un motor Brushless. Para el caso de este IP, mediante la realimentación de la posición del rotor detectada a través de los sensores HALL, se activan las distintas partes del puente inversor con el fin de realizar la correcta excitación de sus devanados. El grado de excitación de estos últimos vendrá determinado por el ciclo de trabajo de la señal PWM de los transistores.

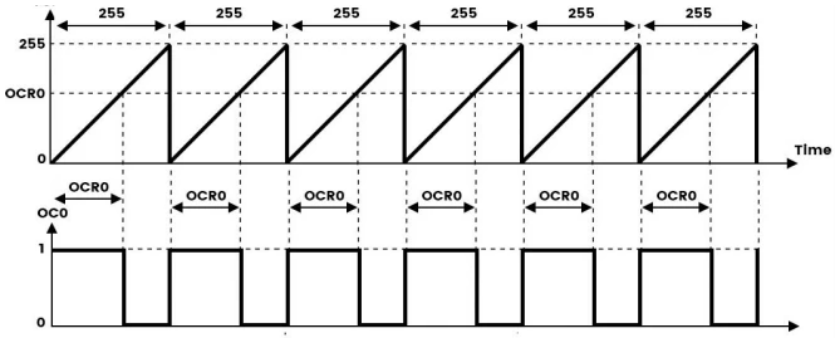
Con estas premisas se puede resolver la cuestión del control de la alimentación mediante la creación de dos módulos.
Un módulo que genere señales PWM y un módulo decodificador que permita determinar que transistores activar y desactivar en función de la realimentación de los sensores HALL.



4.3.2.2.1. Generador PWM

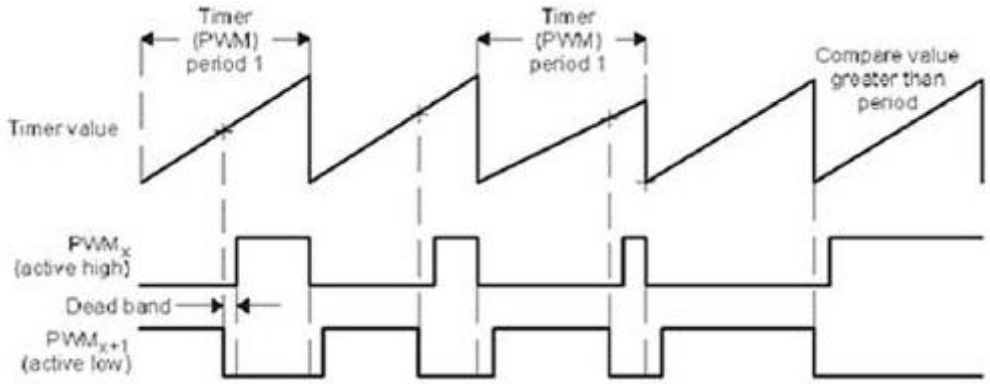
El siguiente módulo se encarga de generar una señal PWM en función del ciclo de trabajo asignado.

Su funcionamiento es el habitual para la generación de dichas señales. Un contador inicia una cuenta, al llegar está a su máximo o mínimo sea la cuenta ascendente o descendente se reinicia el contador. El valor del ciclo de trabajo entra en acción cuando el contador llega al valor estipulado por este en la cuenta. En su caso si inicia con valor lógico 1, convertirá su valor al opuesto es decir 0. Este valor se mantiene hasta llegar al final del contador y este se reinicie volviendo a dar una salida a 1.

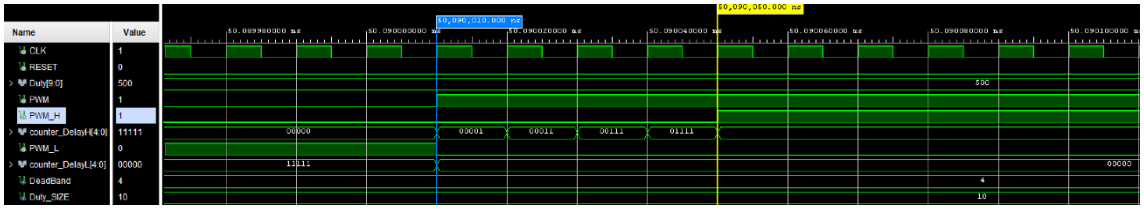


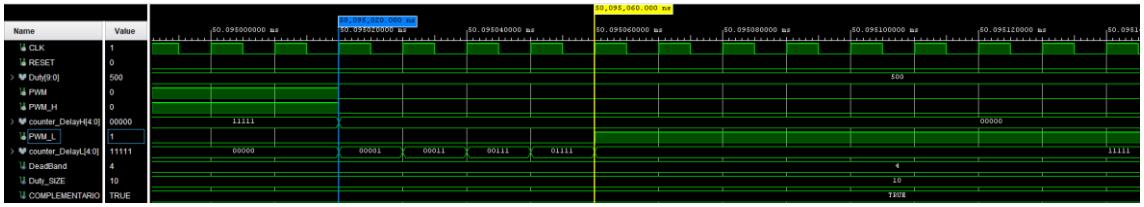
Sin embargo, para el caso de este generador de PWM, se le ha programado la capacidad de poder generar una señal complementaria.
Como se expuso en el apartado XXXXX se puede realizar el control del puente generando una señal de PWM en los transistores superiores. Cuando la señal del transistor superior está apagada se puede generar una señal opuesta en el transistor inferior de la fase del inversor.

La activación complementaria no se puede realizar de forma directa al apagar la señal superior ya que se debe respetar tiempo de "DeadBand". Este tiempo de seguridad permite cerrarse al transistor superior antes de la activación de su opuesto inferior evitando cortocircuitos.

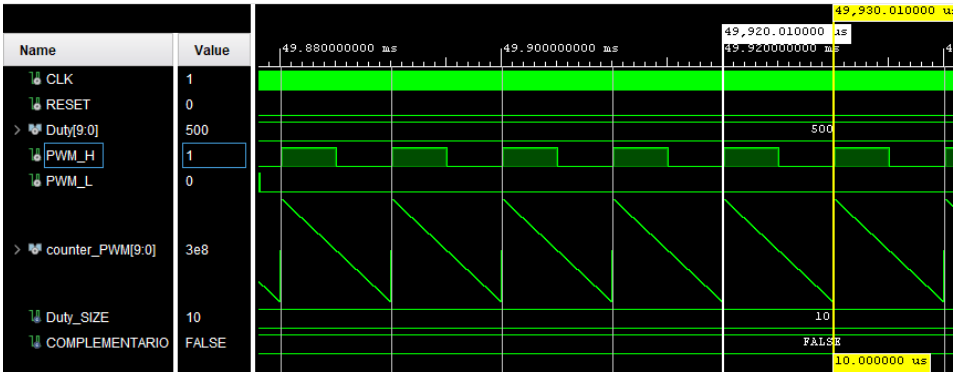


Por lo tanto, el módulo programado permite generar una señal de PWM única o con su correspondiente señal complementaria.
Se puede configurar el Deadband para que su duración sean los x siglos deseados.
Por último la frecuencia de PWM puede ser configurada para funcionar en un rango de funcionamiento de 40KHz-100KHz.
La frecuencia, Deadband y el tipo de señal simple o con su parte complementaria son definidas como genéricos en la declaración del componente.





El PWM mostrado en las dos imágenes anteriores corresponde a una frecuencia de 100KHz y para un ciclo de trabajo de 50%. Con modo complementario activado y un deadband de 4 ciclos se observa como las salidas PWM_H (PWM alto) y PWM_L se encuentran espaciadas por cuatro ciclos de reloj.

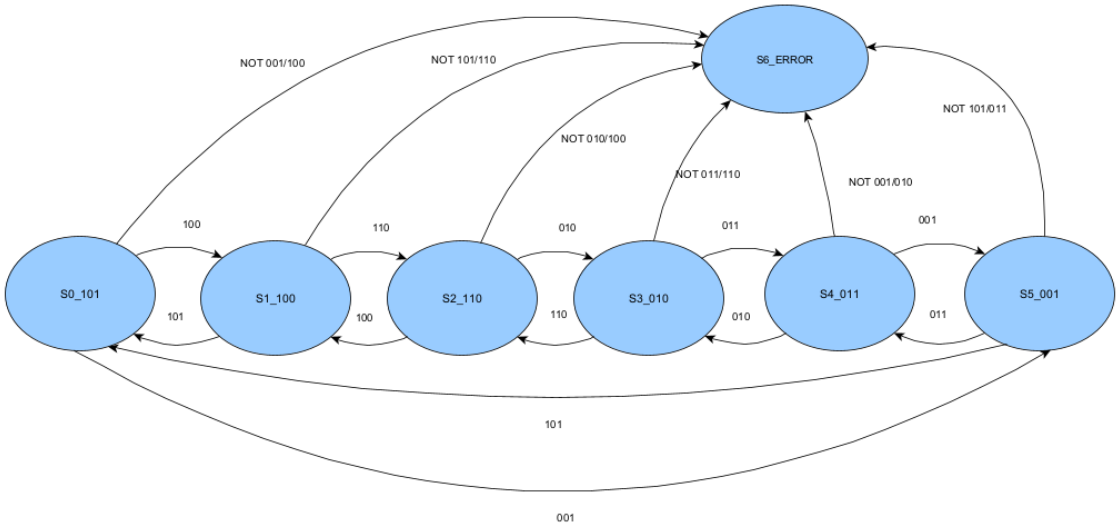


Con el modo Complementario desactivado se aprecia como la única señal en realizar el PWM es la salida PWM_H, la salida de PWM_L permanece desactivada de forma constante.

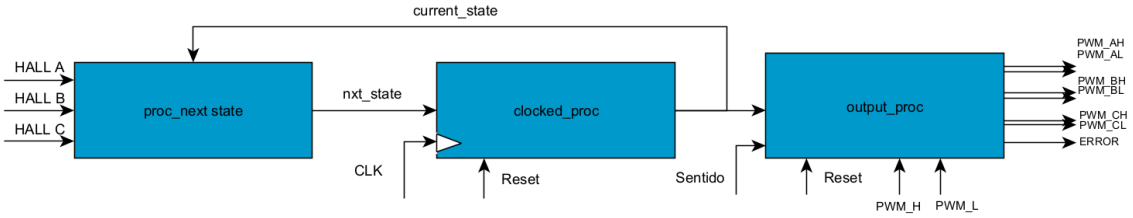
4.3.2.2.2. Decoder PWM

Con respecto a lo expuesto en la introducción el siguiente módulo descrito tiene la tarea de determinar que transistores del inversor se deben activar en función de la señal captada a través de los sensores HALL.
En el apartado XXXXX se introdujo la combinación necesaria de alimentación de las 3 fases del motor para poder realizar un giro horario u antihorario.
Cada ciclo eléctrico se compone de 6 combinaciones, por lo tanto se ha planteado este decodificador como una máquina de estado la cual en función de sus tres entradas(HALL A, HALL B y HALL C) determina cuales de sus 6 salidas (PWM_AH, PWM_AL, PWM_BH, PWM_BL,

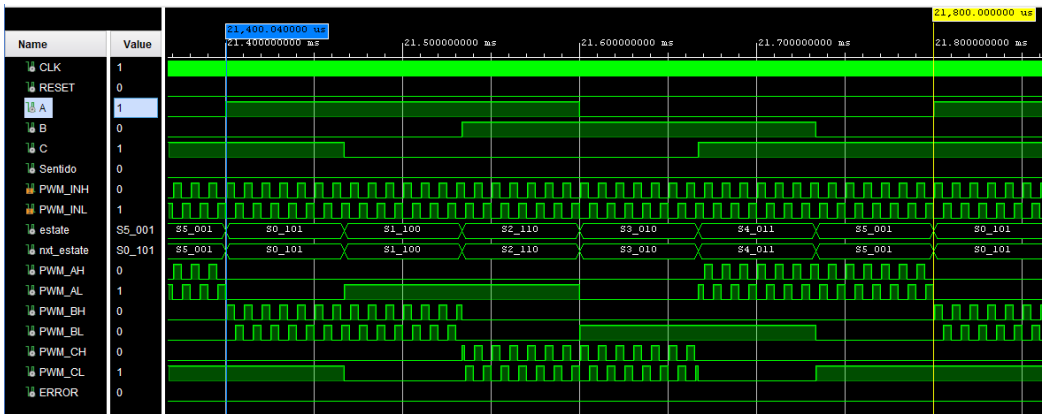
PWM_CH y PWM_CL) se activan. Esta máquina de estado es de tipo Mealy ya que el valor de las salidas no solo depende del estado en el que se encuentran, sino de sus entradas sentido (horario y antihorario), Reset y PWM_H y PWM_L.

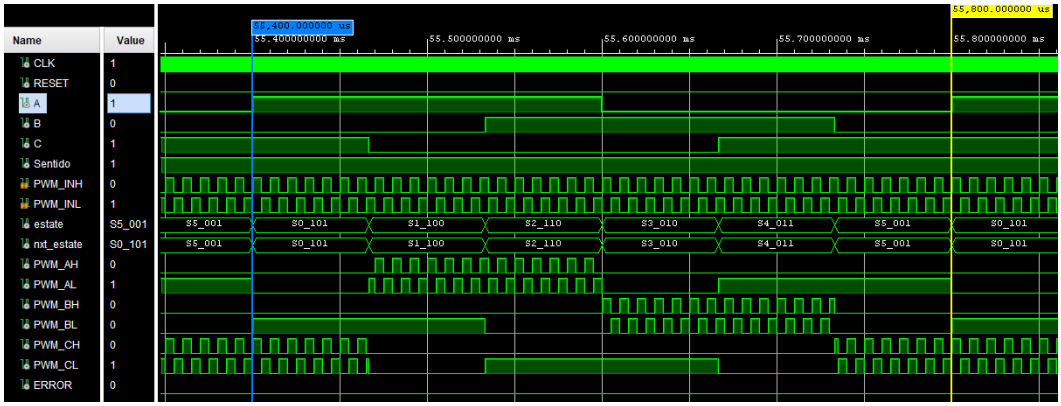


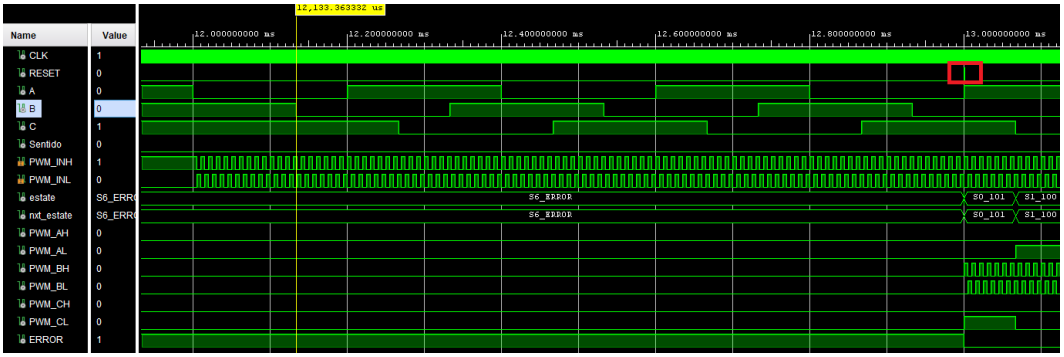
Para el diagrama de flujo de estados mostrados se ha señalizado únicamente los cambios de estado en función de las entradas de los tres sensores HALL (A, B, C).
Se puede observar como el proceso de determinación del siguiente estado va ligado al valor de los sensores HALL. Los sensores HALL emiten una señal desfasada 30^0 o 60^0 entre ellas por lo tanto el cambio entre un estado y otro se debe al cambio de una única señal. Si se diera el caso de un cambio de dos señales, esto repercutiría en un error debido a que o se están perdiendo lecturas o los sensores no funcionan correctamente. La máquina de estado refleja este caso trasladándose a un estado pozo llamado S6_Error y anulando las salidas. Solo se puede salir de este estado poniendo a RESET el sistema.



En el diagrama de bloques del módulo se aprecia como la salida depende del estado actual y las entradas de Reset, Sentido, PWM_H y PWM_L. Estas dos últimas señales son generadas por el módulo *Generador PWM* mencionado en el apartado anterior.







Como se puede apreciar en las dos primeras simulaciones, a la salida se aplica el valor en ese momento de las entradas PWM_H y PWM_L y según el valor del sentido horario u antihorario las salidas de los 6 estados de un ciclo eléctrico varían en función del valor de dicha entrada.

En la última simulación, se observan los sensores HALL en una combinación incorrecta lo que redunda en un estado de error. Pese a que las entradas PWM y de sentido estén activas el estado de error anula cualquier salida y activa la salida de error. Finalmente, gracias al Reset, se reinicia el sistema estableciéndose el estado correcto.

4.3.2.3. Controlador PI

Los algoritmos de control de PID se basan en tres aspectos básicos, los modos proporcional, integral y derivativo.
Cuando se utiliza dicho algoritmo es necesario definir que parámetros y en cual de sus combinaciones se van a utilizar. Generalmente existen 3 tipos de algoritmos el P, PI y PID. Estos son ampliamente utilizados en microprocesadores y chips DSP. Sin embargo, en los últimos tiempos su incorporación a las FPGAs se ha vuelto muy extendido.
La forma simple de la ecuación de un PID es la siguiente:

$$U(t) = K_p(e(t) + \frac{1}{T_i}\int e(t)dt + T_d\frac{d_e(t)}{dt})$$

La constante Kp es la ganancia proporcional la T_i la integral del tiempo y T_d es la derivación del tiempo, e(t) representa el error de la señal y U(t) es la salida del controlador

Dicha ecuación del PID digitalizada se puede expresar de la siguiente forma.

$$U_n = K_p e_n + K_i \sum_{i=0}^{n} e_i + K_d(e_n - e_{n-1})$$

$$AU_n = U_n - U_{n-1} = (K_p + K_i + K_d)e_n - (2K_d + K_i)e_{n-1} + K_d e_{n-2}$$

$$U_n = U_{n-1} + K_0 e_n + K_1 e_{n-1} + K_2 e_{n-2}$$

Los valores de K_0, K_1 y K_2 son los siguientes.

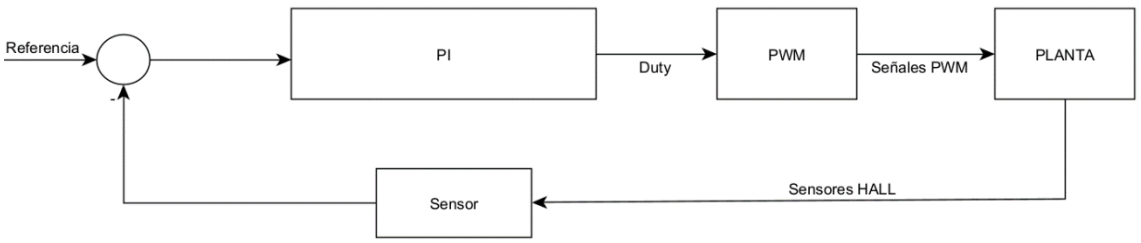
$$K_0 = K_p + K_i + K_d$$

$$K_1 = -(2K_d + K_i)$$

$$K_2 = K_d$$

El controlador que se ha realizado en este proyecto es un controlador de tipo PI. Es decir solo se tendrá en cuenta la parte proporcional e integral aplicada al sistema.

Por lo tanto, la funcion del controlador a programar será la de estimar que ciclo de trabajo es el adecuado en funcion del error entre la referencia y la realimentacion. Este ciclo de trabajo se pasará posteriormente al módulo PWM para su posterior generación.

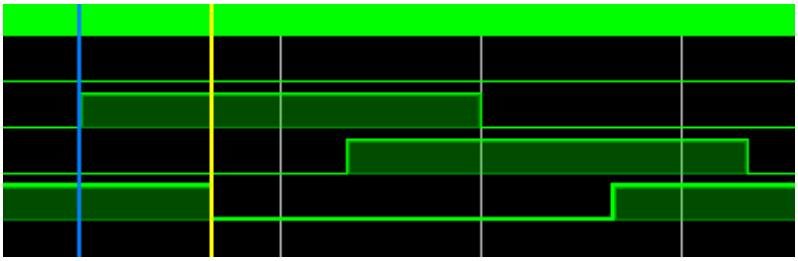


La estimación de las constantes del PI se realizará en el apartado de pruebas de campo. Mediante el procedimiento de sintonía experimental de controladores.

De la FIGURAAA mostrada se aprecian por lo tanto tres funciones en las cuales dividir el controlador. Sensor, PID y Sample.

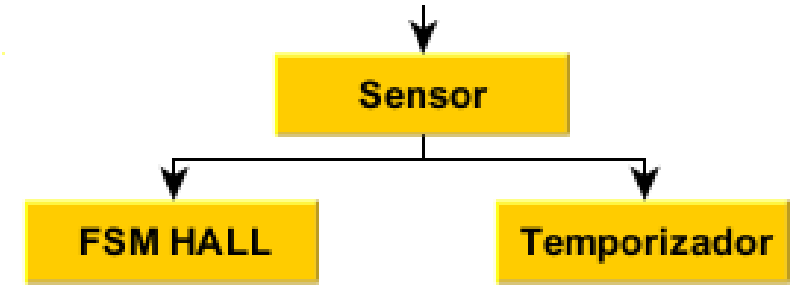
4.3.2.3.1. Sensor

El siguiente submódulo será el encargado de establecer la velocidad del motor. Sin embargo, dicha tarea no es sencilla. El sensor solo recibe tres señales desfasadas de los sensores HALL. Para poder convertir la velocidad en RPM con respecto a la medida entre dos pulsos se realiza de la forma siguiente.



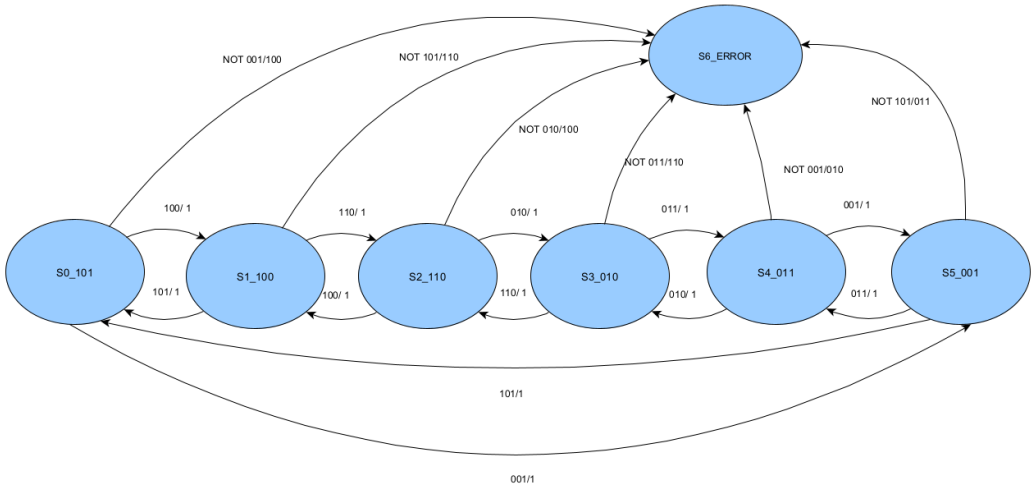
$$RPM = \frac{1}{t_{pulso} * N_{pulsos/vuelta}} * 60$$

Realizar el cálculo en RPM es complejo y da poco margen de valores a la hora de calcular el error con la referencia.
Se ha optado por lo tanto en realizar el control a partir del tiempo que hay entre pulsos de los sensores HALL. La referencia se convierte por lo tanto de RPM->t pulso.
Las tareas del sensor se pueden dividir en FSM HALL y Temporizador.

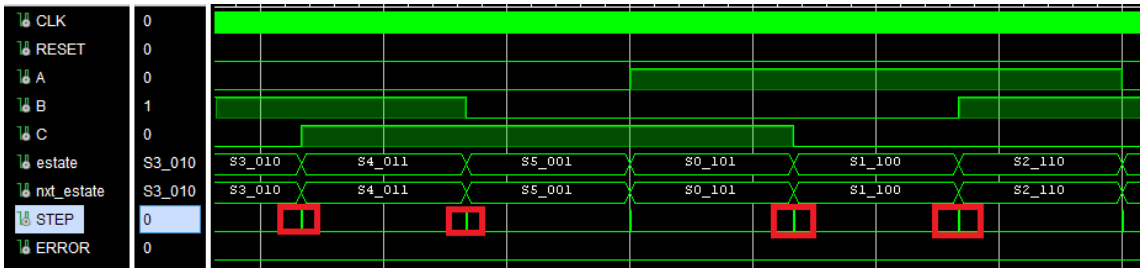


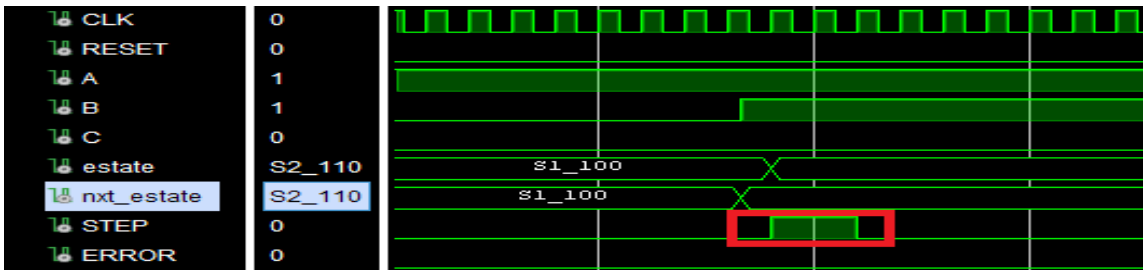
4.3.2.3.1.1. FSM HALL

Este submódulo comparte numerosas similitudes con respecto al descrito en el Decoder PWM. Sin embargo, su única tarea será indicar cuando se produce un cambio en sus estados, esto permitirá estimar posteriormente el tiempo transcurrido entre pulsos.



Al igual que para su homólogo en el Decodificador de señales PWM esta maquina de estados consta de un estado pozo “S6_Error” al cual se transita en caso de no darse correctamente las secuencias de transición.

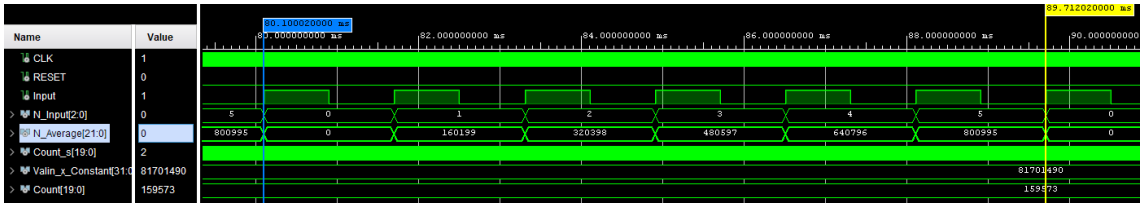


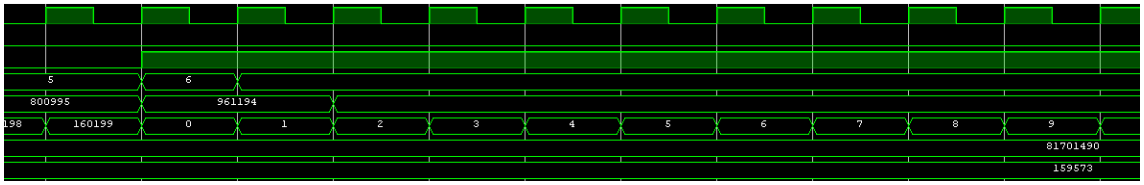


En la primera imagen se aprecian varios pulsos consecutivos tras cambiar alguna de las 3 señales de los sensores HALL. La segunda imagen muestra la secuencia durante el cambio de estado en el momento en el cual se actualiza al nuevo estado.

4.3.2.3.1.2. Temporizador

Este submódulo realiza la tarea de estimación del tiempo transcurrido entre pulsos. Durante las primeras fases, programación y pruebas, se comprobó irregularidades con alimentación a ciclo de trabajo constante del motor en el resultado del tiempo entre pulsos. A fin de obtener una muestra con mayor continuidad se optó por realizar una media de los pulsos calculados por ciclo eléctrico. Es decir, un contador realiza la estimación de tiempo entre pulsos y posteriormente y almacena en un acumulador. Al realizarse 6 pulsos, correspondientes a un ciclo eléctrico, se realiza la media de los 6 valores de tiempo acumulados.





En la primera imagen se aprecia como van acumulándose en la señal N_Average las cuentas realizadas por el contador Count_s. La segunda imagen corresponde al instante del sexto pulso detectado, en este instante se suma el ultimo tiempo medido y se procede a realizar la media de las seis medidas.
Las divisiones en binario no son tarea sencilla, una estrategia para resolverlas es buscar multiplicar el valor a dividir por una constante que nos permita posteriormente dividir por una potencia de dos. Dividir por una potencia de dos consiste en realizar un desplazamiento a la derecha de N bits igual a la potencia.
Para el caso de la media se desea realizar una división entre 6.

$\frac{1}{6} = 0.1\hat{6}$ su equivalente aproximado $\frac{85}{2^9} = 0.166015$

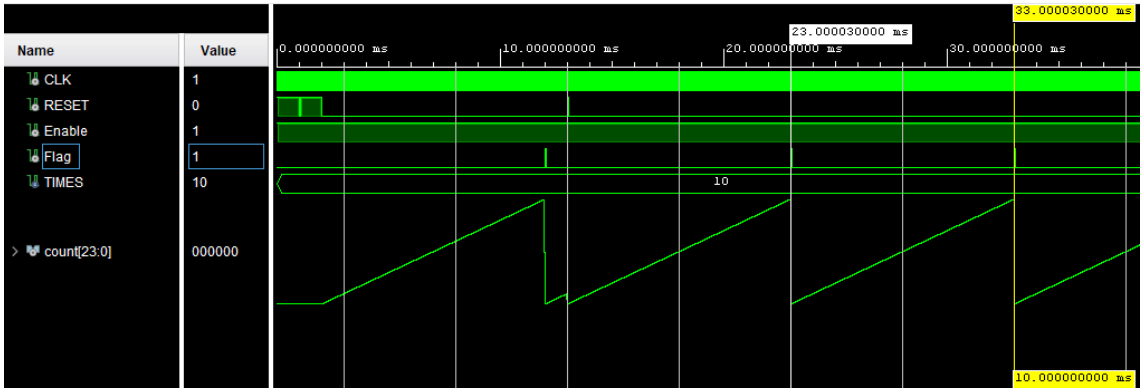
En la simulación se aprecia que el valor 160199 se repite 6 veces.
El resultado de la media es 159573. El error relativo entre el resultado teórico y aproximado es el siguiente.

$$e_r = \frac{160199 - 159573}{160199} = 0.0039$$

Con este margen se estima poder realizar la división sin perder demasiada precisión en el error.

4.3.2.3.2. SAMPLE

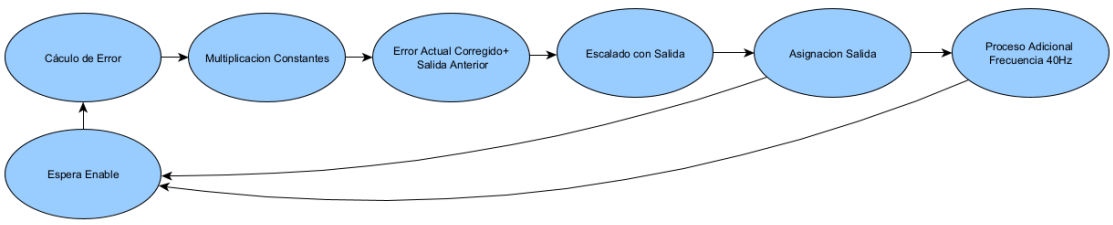
El siguiente submódulo permite configurar el tiempo de muestreo en un margen entre 1ms-100ms. La configuración del tiempo de muestreo permite establecer según los requisitos del proyecto el muestreo y reacción ante nuevas medidas captadas por el sensor.
Su configuración redunda en un simple contador que al llegar al número de ciclos deseado genera una señal de aviso que permite avisar al controlador de realizar otro calculo en base a las nuevas medidas capturadas por el módulo sensor.



La configuración del tiempo de muestreo es definida como genérico en la declaración del componente.

4.3.2.3.3. PI

El módulo PI gestiona el cálculo del error por las constantes de un controlador PI. Al activarse la señal de enable se inicia un proceso secuencial con cada ciclo de reloj que realiza las tareas de cálculo del controlador. Dicho proceso secuencial solo puede ser interrumpido en caso de activarse la señal de Reset del sistema.



Cálculo del error: El cálculo del error se realiza restando el tiempo medido por el sensor por el tiempo de referencia. En un sistema convencional el error es medido gracias a la resta de la referencia por el valor realimentado en este caso se realiza dicha operación al revés para evitar tener el error invertido. El tiempo medido es inversamente proporcional a la velocidad, para

un tiempo captado por el sensor de valor alto, significa que la velocidad es lenta ya que el tiempo entre pulsos es mayor y en caso de velocidades altas el tiempo entre pulsos es menor. Por lo tanto, si se realiza la resta de forma convencional con un error negativo la velocidad es inferior y con un error positivo la velocidad es superior.
Con el fin de evitar esta inversión en el sentido, se invierten los signos en la resta obteniendo por lo tanto a error positivo, velocidad menor que la referencia y a error negativo velocidad mayor que la referencia.

```
elsif Enable='1' AND Flag="0001" then
        Error<=unsigned(Sensor(19 downto 5 ))- signed(Set_Point(19 downto 5));
    Flag<="0010";
```

A su vez se aprecia en la resta como se escala los valores del sensor y de referencia al dividirlos por una potencia de 2.

Multiplicación de constantes: Se realiza la multiplicación del error por la constante proporcional, en el caso de la corrección proporcional. Para el caso de la constante integrativa se realiza la multiplicación de su constante por el error actual sumado al error anterior.
El error actual será a su vez almacenado a final de ciclo en la variable del error anterior.
No se realiza un sumatorio de los errores anteriores debido al riesgo de desbordamiento de la variable optando por lo tanto por almacenar únicamente el error anterior.

```
if ExternalP='0' then
    Pout<=conv_integer(signed(Error))*KP;
else
    Pout<=conv_integer(signed(Error))*conv_integer(unsigned(Proportional));
end if;
if ExternalI='0' then
    Iout<=conv_integer(signed(Error)+signed(SumarError))*KI;
else
    Iout<=conv_integer(signed(Error)+signed(SumarError))*conv_integer(unsigned(Integral));
end if;
```

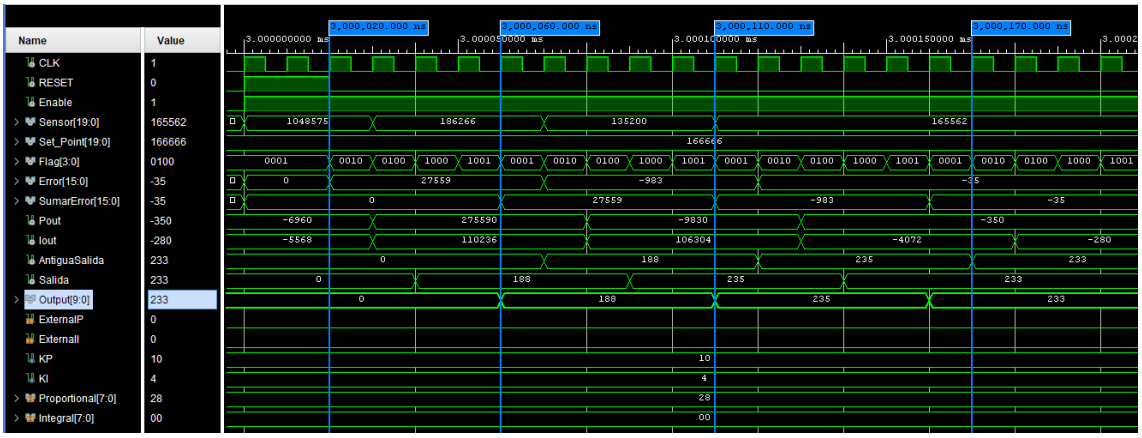
Escalado con Salidas: Se establece un valor máximo y mínimo en la salida calculada que permite en caso de estar fuera de los márgenes establecer un ciclo de trabajo del 100% o del 0%.
El cálculo realizado por el PI permite determinar el ciclo de trabajo para una frecuencia de 100KHz. En caso de seleccionarse 50KHz o 40KHz las dos últimas etapas permiten realizar el cálculo para adaptar el valor del duty a dichas frecuencias.

```
if Duty_SIZE=10 then
    Output<=std_logic_vector(conv_unsigned(Salida ,10));
                Flag<="0001";
elsif Duty_SIZE=11 then
    Output<=std_logic_vector(conv_unsigned(Salida*2 ,11));
                Flag<="0001";
elsif Duty_SIZE=12 then
    Output_s<=(std_logic_vector(conv_unsigned(Salida*5 ,13)));
                Flag<="1011";
```

Cuando se tiene seleccionado una frecuencia de 100KHz el duty tiene un rango de 0-1000, 50Khz duty 0-2000 y para 40KHz el duty es de 0-2500.

Cuando es seleccionado una frecuencia de trabajo de 40KHz entra en acción la última etapa que (Proceso adicional frecuencia 40Hz) permite realizar la división a la mitad necesaria para tal caso.

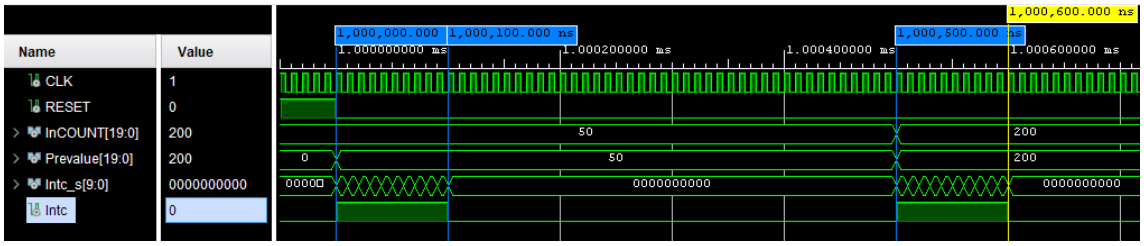
Durante la fase de *Multiplicación de constante* mostrada en XXXXXXX se puede apreciar que la multiplicación del valor proporcional e integral dependen de los valores “Externali” y “Externalp”. Dichas entradas permiten optar por constantes definidas por valores genéricos declarados en el componente o entradas que se pueden variar dinámicamente.
Se ha programado de tal manera esta parte del código a fin de poder variar las constantes posteriormente a través de los registros del bus AXI en el IP.



La siguiente simulación muestra 4 instantes calculados por el controlador PI en el cual se modifican los valores introducidos por la entrada Sensor para una referencia (Set_Point) fija.
En primer lugar se debe precisar que el controlador se encuentra utilizando los valores genéricos KP y KI y no los introducidos por las entradas proporcional e integral. Esto se debe a que las entradas ExternalP y ExternalI se encuentran desactivadas.
La variable Flag muestra codificado el flujo secuencial mostrado en la figura XXXXXX.
Pout y Iout recogen la multiplicación de los errores por sus constantes respectivas. Finalmente se suma la Antigua salida con la suma de las constantes por sus errores escalado entre 2048.
La señal SumarError recoge el valor anterior del error y lo almacena para su posterior uso en el siguiente cálculo.
Podemos apreciar claramente un cambio de tendencia en el crecimiento de la salida cuando el error pasa de ser positivo a negativo.

4.3.2.4. Generador Interrupción

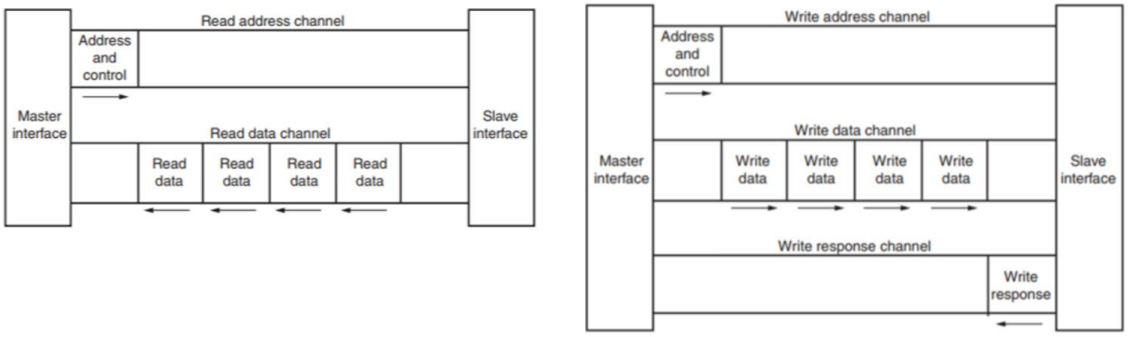
Como se describió al inicio de esta sección una de las funcionalidades deseadas para el IP es la de generar una señal de interrupción.
Las señales de interrupción pueden ser de 2 tipos por nivel lógico u flanco tanto d activo o inactivo, ascendente u descendente. Para este caso concreto se desea generar una interrupción de flanco de subida.
El siguiente módulo genera un pulso de 10 ciclos de duración para y posteriormente se deshabilita.
La activación de la salida se realiza cuando el reset se encuentra desactivado y a la señal de entrada se presenta un valor distinto al almacenado en el registro interno de valores de entrada.



En la siguiente simulación, cuando entra un valor nuevo a la entrada no registrado en la señal Prevalue se activa la interrupción. La duración es de 10 ciclos y se gestiona realizando un desplazamiento de 0 del menos significativo al más significativo, en un vector de 10 bits de longitud, siendo la salida Intc el valor del bit más significativo.

4.3.2.5. Displays 7 Segmentos.

Debido a la temática del TFG, se va a profundizar en el funcionamiento básico del protocolo AXI-Lite.
Al igual que su homólogo AXI4 el protocolo AXI4-Lite consta de cinco canales. Dos canales de lectura (Read Address y Read Data) y tres canales de escritura (Write Address,Write Data y Write Response).

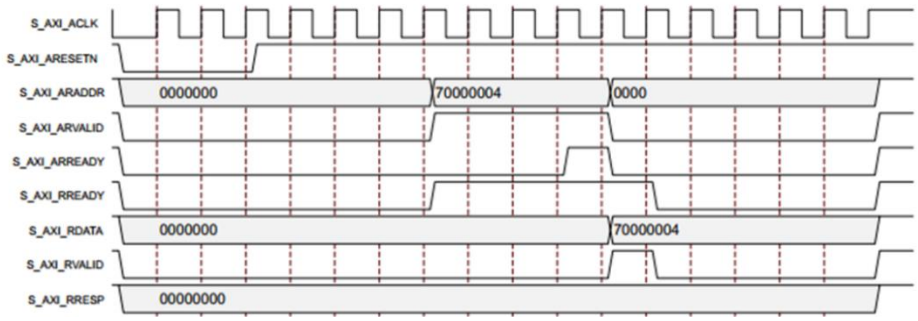


La figura anterior muestra el esquema de comunicación del protocolo AXI4, el protocolo Lite es idéntico con la diferencia principal como se ha mencionado anteriormente de no poder realizar transmisión en ráfaga.
A pesar de ser un derivado del bus AXI4 para poder realizar comunicación entre ambos se necesitan realizar conversiones para permitir comunicación entre AXI4-AXI4Lite.
Se puede resolver dicha problemática con el uso de un puente (bridge) entre ambos protocolos o se puede definir un esclavo AXI4-Lite que incluya lógica de reflexión de ID. Esto permite evitar el uso de puentes e integrar este en un sistema AXI completo con la garantía de que se accede al esclavo mediante transacciones que cumplen con el subconjunto del protocolo AXI4-Lite.[23]

Global	Write address channel	Write data channel	Write response channel	Read address channel	Read data channel
ACLK	AWVALID	WVALID	BVALID	ARVALID	RVALID
ARESETn	AWREADY	WREADY	BREADY	ARREADY	RREADY
–	AWADDR	WDATA	BRESP	ARADDR	RDATA
–	AWPROT	WSTRB	–	ARPROT	RRESP

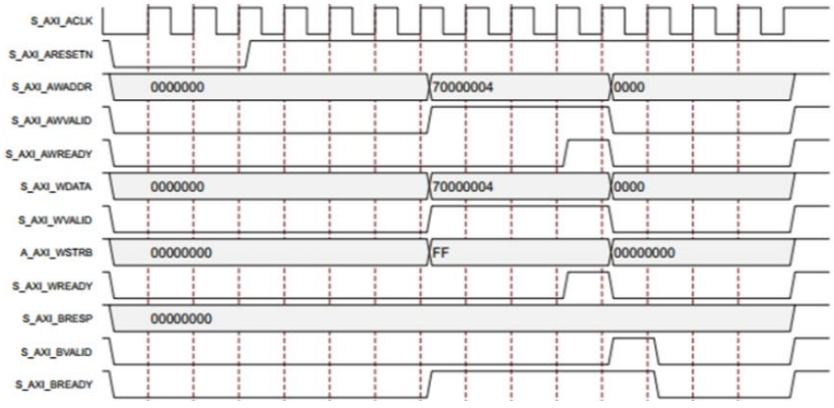
En el cuadro de señales asociadas se pueden apreciar en cada uno de los cinco canales unas señales de VALID y otra señal de READY, ello es debido al proceso de “handshake” (apretón de manos). Este mecanismo doble permite tanto al maestro como al esclavo realizar un control sobre la información que se mueve entre ambos.
La fuente emisora de información genera la señal VALID para indicar que la dirección, dato o control de la información está disponible. Por el contrario, el receptor genera una señal de READY la cual indica que puede aceptar la información. El apretón de manos se realiza cuando tanto la señal de VALID y READY de un mismo canal son asignadas durante un flanco de subida de reloj.[22]

- **Proceso de lectura:**



1. La secuencia de lectura de un registro del dispositivo esclavo comienza por el maestro colocando la dirección del registro en el canal de lectura de dirección (Read Address Channel) a la vez que asigna nivel ARVALID, indicando por lo tanto que la dirección en ese canal es válida.
2. A su vez RREADY indica que el maestro esta listo para leer datos del esclavo.
   El esclavo indica a su vez mediante la señal ARREADY que está disponible para recibir la dirección.
3. A partir del momento en el que ARVALID y ARREADY están activadas en el siguiente flanco de subida de reloj se produce el apretón de manos (handshake). Por lo tanto, el esclavo ya ha recibido la dirección del registro. Una vez ocurrido este proceso tanto maestro como esclavo desenclavan a nivel bajo las señales de ARVALID y ARREADY. El esclavo ya ha recibido la direccion.
4. El esclavo coloca la información del registro requerido en el canal de escritura (Read Data Channel) y activa la señal de RVALID indicando nuevamente que el dato en el canal de lectura de datos es válido. Finalmente, en el siguiente flanco de subida del reloj se desactivan las señales RREADY y RVALID.

- **Proceso de escritura:**



1. El maestro introduce tanto en el canal de escritura y lectura (Write Adress Channel y Write Data Channel) la dirección y dato a transmitir.
   En paralelo activa las señales de AWVALID y WVALID indicando al esclavo la validez de los datos y la disponibilidad de una respuesta por parte del esclavo. El maestro

también activa la señal BREADY del canal de respuesta a la escritura (Write Response Channel).
2. El esclavo a su vez activa las señales de AWREADY y WREADY.
3. Se produce un doble apretón de manos (handshake) tanto en el canal de direcciones como de datos. En el siguiente flanco de reloj se desactivan las señales AWREADY, AWVALID, WVALID y WREADY.
4. Finalmente, el esclavo responde activando BVALID indicando por lo tanto que la transacción ha sido correcta. En el siguiente flanco positivo de reloj ambas señales se desactivan.

REGISTROS IP CREADO

1.1. Mapa de registros AXI4-Lite del IP creado.

Nombre	Tipo	Direcciones Registros		Defecto	Comentario
		Hexadecimal	Binario		
Reg0_Star_Stp	r/w	0x00	00000000	0x00000000	-
Reg1_Const	r/w	0x04	00000100	0x00000000	-
Reg2_ Dut_SP	r/w	0x08	00001000	0x00000000	-
Reg3_Hor_AntH	r/w	0x0C	00001100	0x00000000	-
Reg4_InVel	r	0x10	10000000	0x00000000	-
Reg5_OutVel	r/w	0x14	10000100	0x00000000	-

Descripcion de registros.

Dependiendo del modo de funcionamiento escogido los registros pueden variar en su funcionalidad interna.

Reg0_Star_Stp

Registro de Start Stop dedicado a gestionar el encendido o apagado (r/w).

Star_Stp

Star_Stp [31-0]	Habilita Start/Stop, permite encender o apagar el módulo de control sin utilizar el RESET general. (0x00000000: Stop; 0xFFFFFFFF: Start). Si no se introduce la combinación correcta por defecto se activa el Stop. Valor por defecto: 0x00000000.

Reg1_Const

Modo Controlador

Registro de Constantes dedicado a gestionar la configuración desde SW/HW de las constantes del controlador PI (r/w).

FlagKp	FlagKi	----------	KP	KI

FlagKp	Habilita el uso de la constante KP [7:0] para ser usada por el controlador. Valor por defecto: 0.
FlagKi	Habilita el uso de la constante KI [7:0] para ser usada por el controlador. Valor por defecto: 0.
--------	Espacio sin uso.
KP [7:0]	Constante proporcional. Se activa con FlagKp. Valor por defecto: 0x00.
KI [7:0]	Constante integral. Se activa con FlagKi. Valor por defecto: 0x00.

Modo Lazo Abierto

---------[31:0]	Este registro no tiene efecto sobre el funcionamiento del módulo. Valor por defecto: 0x00000000.

Reg2_ Dut_SP

Configuración Modo Controlador

Registro dedicado a gestionar la configuración de la referencia del controlador PI. (r/w)

----------	SP

SP[19:0]	Tiempo entre pulso con base a la frecuencia de reloj CLK. Valor por defecto: 0x00000.

Configuración Modo Lazo Abierto

Registro dedicado a gestionar la configuración de ciclo de trabajo (Duty) del módulo. (r/w)

----------	Dut

Dut[X:0]	Valor de ciclo en función de la frecuencia de PWM. Según la frecuencia escogida; 100KHz, 50KHz o 40KHz. El tamaño del registro puede ser [11:0], [10:00], [9:0]. 100KHz(0% 0x000000; 100% 0x3E8);50KHz(0% 0x000000; 100% 0x7D0) 40KHz (0% 0x000000; 100% 0x9C4). Valor por defecto: 0x00000.
------- [31:X]	Registro no utilizado.

Reg3_Hor_AntH

Registro dedicado a gestionar la configuración de sentido de giro horario u antihorario. (r/w)

Hor_AntiH

Hor_AntiH [31:0]	Gestión del sentido de giro. (Horario: 0x00000000; Antihorario: 0xFFFFFFFF). Valor por defecto: 0x00000000.

Reg4_InVel

Registro dedicado a gestionar la lectura del tiempo entre pulsos de los sensores HALL. (r)

----------	InVel

InVel [19:0]	Registro con tiempo entre pulsos de sensores HALL. Valor por defecto. (0x0000FFFF).
------[11:0]	Registro no utilizado.

Reg5_OutVel

Registro dedicado a gestionar la comunicación al módulo de la velocidad calculada. (r/w)

----------	OutVel

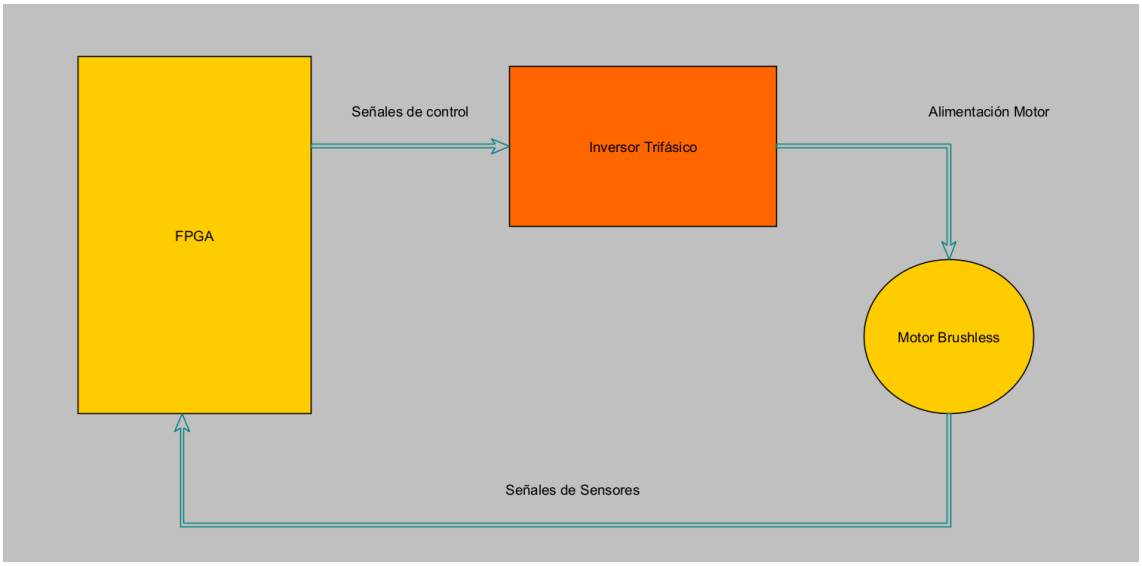
OutVel [12:0]	Registro con velocidad calculada. Valor por defecto. (0x00000000).
------[19:0]	Registro no utilizado.

2. Maqueta Demo Alimentación Motor Brushless.

En esta sección se procede a exponer el diseño y posterior desarrollo de la maqueta realizada para la alimentación de un motor sin escobillas con el fin de poder realizar ensayos y pruebas sobre este mismo.

2.1. Diseño

Como ya se ha podido abordar en la cuestión anterior sobre la alimentación de motores brushless se procede a plantear un esquema general de los elementos presentes en la maqueta a realizar.
Resultando en el siguiente diagrama en el cual se pueden apreciar claramente los principales elementos que intervendrán.



En la siguiente figura se aprecian los tres principales componentes que intervendrán.

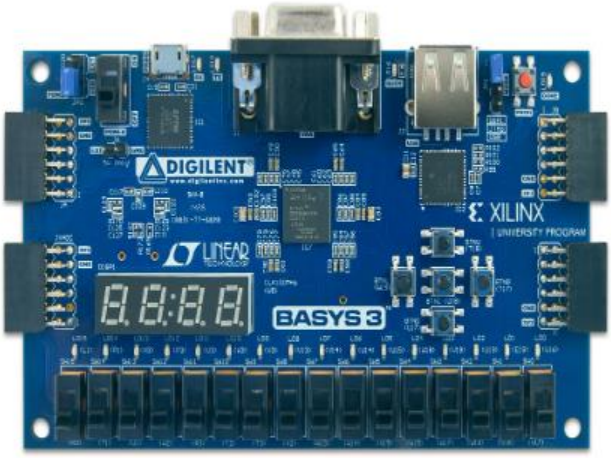
- **Motor Brushless**: Con sus tres fases de alimentación, este motor girará a las revoluciones deseadas en función de la alimentación recibida en sus bobinados.
- **Inversor Trifásico**: La etapa de potencia que gestionará la correcta alimentación del motor sin escobillas a partir de las señales de control (PWM).
- **FPGA**: Será la encargada de realizar las tareas de control del sistema recibiendo las señales de los sensores (efecto HALL) y generando señales de control (PWM) correspondientes para controlar el inversor.

Además de los principales componentes mencionados se deberán añadir otros secundarios que permitan el correcto funcionamiento y relación de los elementos principales entre sí.

2.2. Componentes

En este apartado se enumeran aquellos componentes eléctricos utilizados para el desarrollo de la maqueta de pruebas.

Basys3 Artix-7 FPGA Board.



La Basys3 es una placa de desarrollo diseñada para ser utilizada de forma exclusiva con la herramienta Vivado Design Suite.
Con su arquitectura de Artix-7 desarrollada por Xilinx, esta placa de desarrollo para aprendizajes dispone de una amplia colección de puertos de E/S con los cuales poder cumplir multitud de tareas.

Principales Características de la Artix-7 35T.

- 33,280 logic cells distribuidas en 5200 slices (cada slice contiene 6 LUTs y 8 flip-flops).
- 1,800 Kbits de BRAM.
- 5 posibles administraciones de reloj, cada uno con un su propio PLL.
- 90 DSP slices.
- La frecuencia de reloj alcanza los 450 MHz.
- Convertidor analógico-digital embebido (XADC).

Inversor BOOSTXL-3PHGANINV

El inversor BOOSTXL-3PHGANINV permite trabajar a altas frecuencias de PWM gracias a la presencia en sus tres fases inversoras de drivers LMG5200 Gan de medio puente.
Las principales motivaciones a la hora de elegir este inversor frente a los demás son las siguientes:

- Permite el control PWM del puente mediante señal de 3.3V, esto facilita mucho el montaje gracias a que se evita una etapa de elevación de nivel lógico 3.3V-5V.
- Su amplio rango de alimentación 12V-60V.
- La alta frecuencia de conmutación soportada.

Junto a las mencionadas anteriormente se pueden resumir en la siguiente tabla las principales características del inversor trifásico.

Parametros	Valores Típicos	Comentarios
DC entrada voltaje	48 V (12 to 60 V)	80-V máximo absoluto
Corriente máxima de salida trifásica	$7A_{RMS}$ (10-A_{PEAK}) por fase	-
Potencia máxima de entrada	400 W (a 48 V)	-
Tipo FET de potencia	Tecnología GaN	Módulo de alimentación de medio puente con controladores de compuerta de lado alto y bajo integrados (LMG5200)
Frecuencia de conmutación PWM	40 a 100 kHz	-
PWM deadband	12.5 ns	-
Máxima eficiencia 100-kHz PWM	98.5%	A 400-W potencia de entrada
Sensor y amplificador de corrientes de fase.	5-mΩ shunt por INA240	Amplificador diferencial de detección de corriente con 20 V/V y rechazo de PWM mejorado (INA240).
Rango máximo de corriente en fases.	±16.5 A	Escalado de 0 a 3.3 V; invertido con polarización 1.65-V
Rango precisión de corrientes de fase (–25°C a 85°C)	±0.5% (no calibrado), ±0.1% (calibrado)	Sobre rango nominal ±10 A; calibración única de compensación y ganancia 25ºC
Filtro de entrada y salida EMI	Externo	-
Pila de capas de la PCB	Cobre, cuatro capas de 70-µm	-
PCB tamaño	53.4 mm × 78.9 mm	Dimensions in mil: 2105 mil × 3107 mil
Rango Temperatura	–40ºC a 85ºC	
Alerta PCB exceso temperatura	> 85ºC	Configurable de 70ºC a 85ºC (TMP302B)

Se aprecia que el inversor escogido tiene entre sus características la capacidad de medir la corriente en los devanados junto a filtros contra las interferencias producidas por la acción de las conmutaciones del PWM. Para el caso de esta demo no se hará uso de dichas características limitándose al control del puente mediante las señales PWM.

Motor Brushless Modelo 2163788

Una consideración que se ha tenido en cuenta a la hora de escoger el motor, es el tener integrado en su estructura sensores HALL. Dicha consideración permite reducir etapas y

conexionados en la maqueta al no depender de sensores externos al motor, los cuales además corren el riesgo de introducir ruido al sistema.

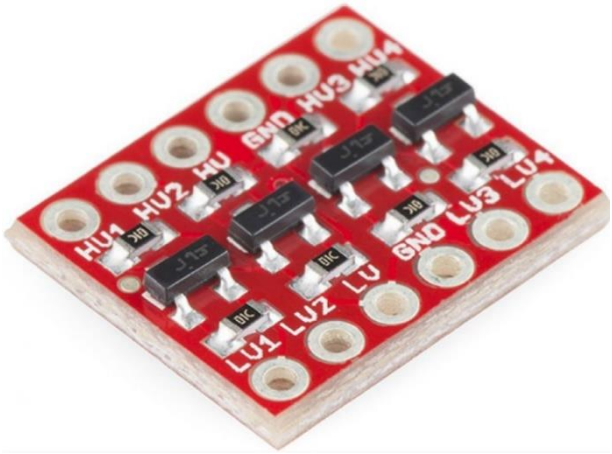
Las características del motor sin escobillas y sus sensores integrados son las siguientes:

Parámetros	Valores
N° polos	8
N° fases	3
Voltaje	24 V
Revoluciones	4800 rpm
Par promedio	0,016 Nm
Máximo Par	0,05 Nm
Constante de par	0,036 Nm/A
Resistencia de línea	5,1 Ω
Inductancia de línea	3,3 mH
Corriente sin carga	150 mA
Máxima corriente de pico	1,4 A
Corriente media	0,44 A
Longitud	20 mm
Inercia del rotor	6 g-cm²
Masa	0,12 Kg
V_{CC} Sensor HALL	+5 to +24 VDC
Ángulo sensores HALL	120°
Salida del eje	0,025 mm
Clase Aislamiento	B
Juego Radial (450 g carga)	0,02 mm
Juego Axial (450 g carga)	0,08 mm
Máxima fuerza radial (10 MM desde zona frontal)	15N
Máxima fuerza axial	10N
Resistencia dieléctrica	600 VCA_{seg}
Resistencia de aislamiento	100 $Mohm_{min}$. 500 VDC

Convertidor niveles lógicos bidireccional.

A diferencia del inversor que nos permite trabajar directamente con las salidas PWM de la FPGA a 3.3V. Las salidas de los sensores HALL son de 5V, se deben tratar dichas señales ya que

la FPGA solo acepta entradas digitales de 3.3V.
El dispositivo mostrado a continuación permite realizar la conversión de niveles lógicos de 3.3V a 5V y vicerversa.



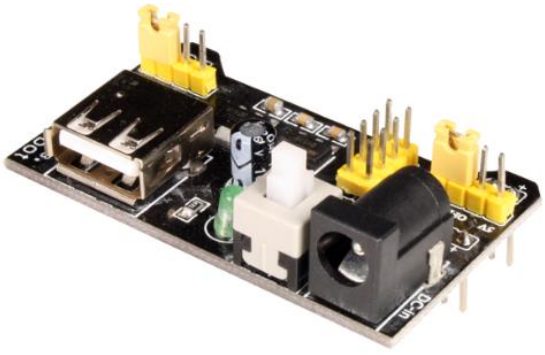
Características:

Parámetros	Valores
Voltaje Alto	5v (2.8V)
Voltaje Bajo	3.3V (1.8V)
N° Canales	4
Sentido	Bidireccional

Fuentes de alimentación.

Se han escogido las siguientes fuentes de alimentación para alimentar los dispositivos mencionados anteriormente.

Fuente de alimentación para board.



Parámetros	Valores
V_{CC} (Jack)	6.5V-12V
N° Salidas	2
Voltaje Salida	5V-3.3V-0(GND)
Corriente máxima	700 mA

Alimentación 12V

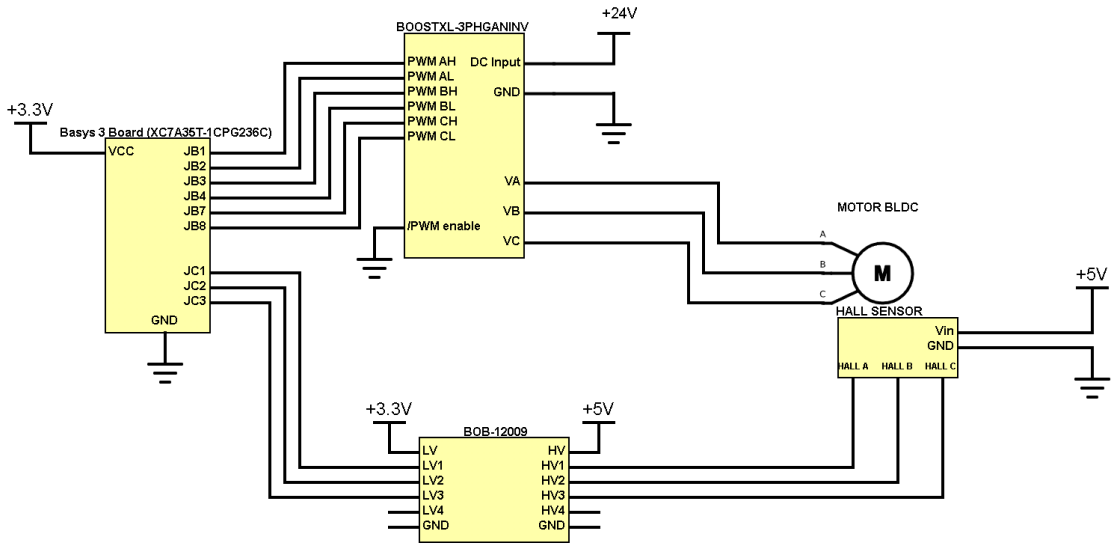
Parámetros	Valores
Entrada AC	220V (50-60 Hz)
Salida DC	1.5-12 V
Corriente máxima	500 mA

Alimentación 24V

Parámetros	Valores
Entrada AC	110-240V (50-60Hz)
Salida DC	24 V
Corriente máxima	2 A.

2.3. Esquema Eléctrico

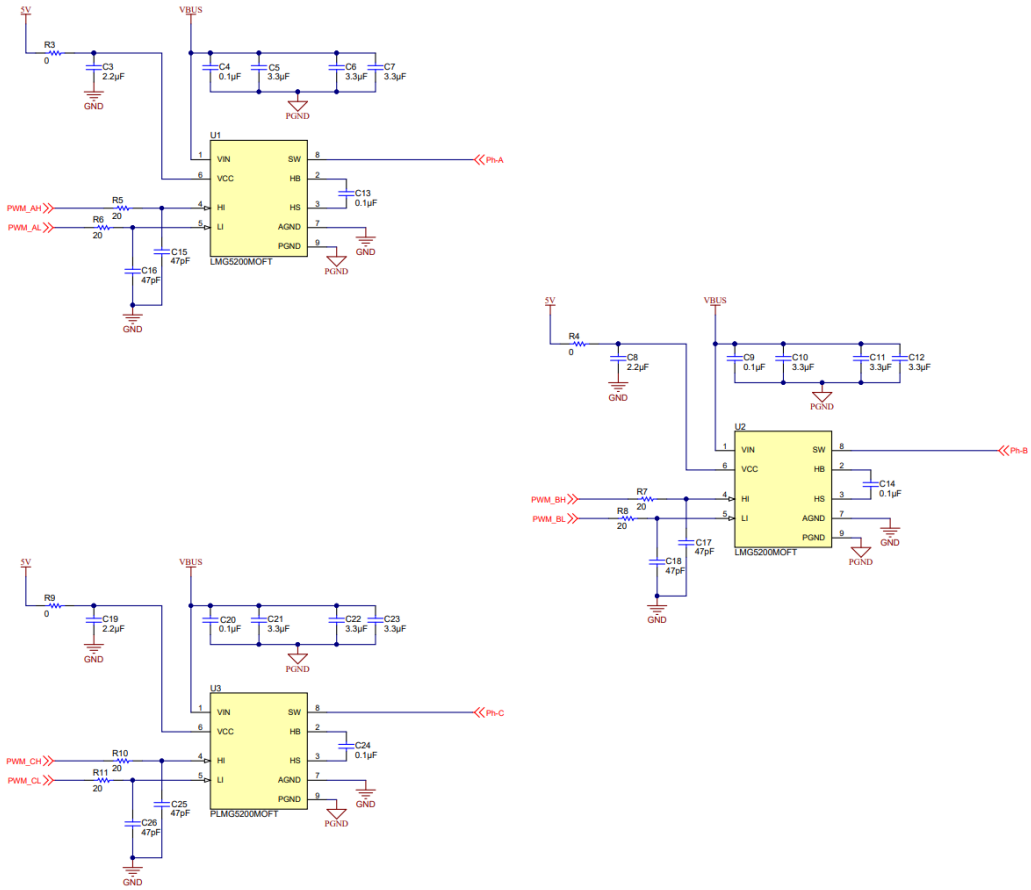
El esquema eléctrico general resultante de la unión de los distintos componentes mencionados, sin incluir las fuentes de alimentación, es el siguiente:



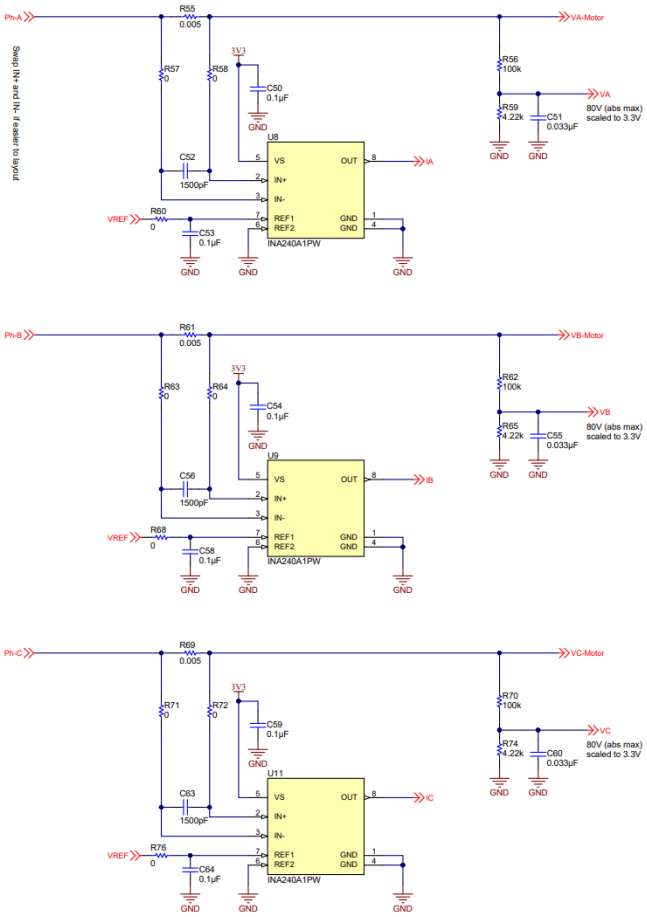
Se incluyen a continuación los esquemáticos de la etapa inversora (BOOSTXL-3PHGANINV) y convertidor de niveles lógicos (BOB- 12009).

- **BOOSTXL-3PHGANINV**

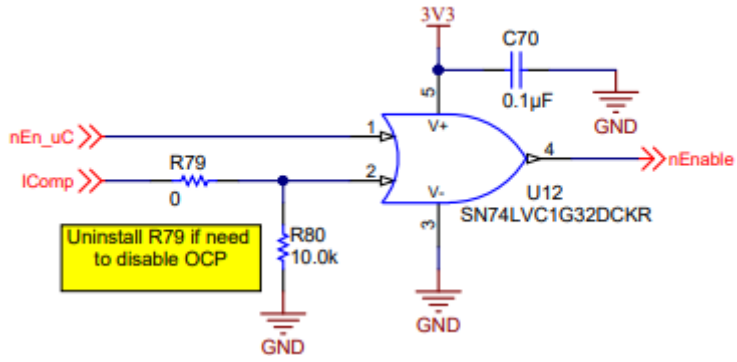
3 Phase Inverter



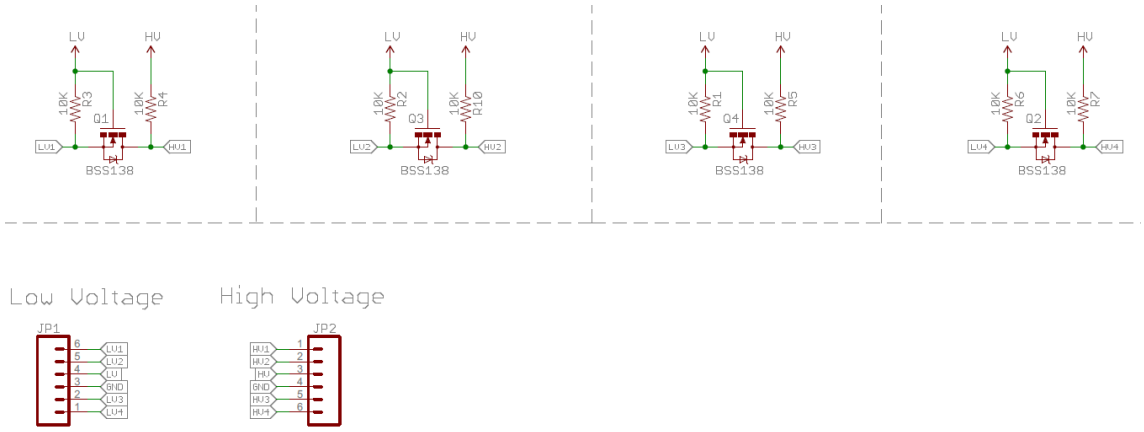
Phase Current/Voltage Sense



PWM ENABLE.



- BOB- 12009 (Niveles Lógicos 3.3V-5V).



4.1. Montaje de la maqueta de pruebas.

En este apartado se va a realizar las imágenes del montaje definitivo de la maqueta de pruebas, se incluirá los planos de perfil y planta realizado con Autocad.
No se realiza este apartado hasta los últimos momentos del TFG cuando se tendrá la disposición de los elementos sobre la maqueta fija y y de forma definitiva.

BIBLIOGRAFIA.

[1] Shivangi Kamat, Sudhamshu M Hosamane, y Anshul Gandhi, «Designing an application-specific System-on-Chip (SoC)», 2016, doi: 10.13140/RG.2.2.27909.76006.

[2] V. S. Chakravarthi, «System on Chip (SOC) Design», en *A Practical Approach to VLSI System on Chip (SoC) Design: A Comprehensive Guide*, V. S. Chakravarthi, Ed. Cham: Springer International Publishing, 2020, pp. 11-40. doi: 10.1007/978-3-030-23049-4_2.

[3] G. Tecnologías, «Sistemas Embebidos SoC - GENERA Tecnologías».

[4] S. Bhunia y M. Tehranipoor, «System on Chip (SoC) Design and Test», en *Hardware Security*, Elsevier, 2019, pp. 47-79. doi: 10.1016/B978-0-12-812477-2.00008-3.

[5] C. M. Maxfield, «Application-Specific Integrated Circuits (ASICs)», en *Bebop to the Boolean Boogie*, Elsevier, 2009, pp. 235-249. doi: 10.1016/B978-1-85617-507-4.00017-6.

[6] M. Maxfield, «EETimes - ASIC vs. ASSP vs. SoC vs. FPGA – What's the Difference?», *EETimes*, 23 de junio de 2014. https://www.eetimes.com/asic-assp-soc-fpga-whats-the-difference

[7] E. Staff, «Fully-programmable SoCs - A new breed of devices», *Embedded.com*, 11 de noviembre de 2016.

[8] D. Oliveira, M. Costa, S. Pinto, y T. Gomes, «The Future of Low-End Motes in the Internet of Things: A Prospective Paper», *Electronics*, vol. 9, n.º 1, p. 111, ene. 2020, doi: 10.3390/electronics9010111.

[9] J. J. Rodriguez-Andina, M. D. Valdes-Pena, y M. J. Moure, «Advanced Features and Industrial Applications of FPGAs—A Review», *IEEE Trans. Ind. Inform.*, vol. 11, n.º 4, pp. 853-864, ago. 2015, doi: 10.1109/TII.2015.2431223.

[10] V. Jayakrishnan y C. Parikh, «Embedded Processors on FPGA: Soft vs Hard», 2019. https://www.semanticscholar.org/paper/Embedded-Processors-on-FPGA%3A-Soft-vs-Hard-Jayakrishnan-Parikh/5155eb3da21a8c1f1e8e246360194cc0556159cb

[11] K. Bouaziz, A. Obeid, y S. Chtourou, «A review on embedded field programmable gate array architectures and configuration tools», *Turk. J. Electr. Eng. Comput. Sci.*, vol. 28, p. 17, ene. 2020, doi: 10.3906/elk-1901-193.

[12] I. Kuon y J. Rose, «Measuring the Gap Between FPGAs and ASICs», *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 26, n.º 2, pp. 203-215, feb. 2007, doi: 10.1109/TCAD.2006.884574.

[13] H. Saidi, M. Turki, Z. Marrakchi, M. Abid, y A. Obeid, «Soft-core embedded FPGA based system on chip», *Analog Integr. Circuits Signal Process.*, vol. 109, n.º 3, pp. 517-533, dic. 2021, doi: 10.1007/s10470-021-01872-5.

[14] J. Patel, Y. Shah, y L. He, «Bridge Design between AXI Lite and AHB Bus Protocol», *J. Phys. Conf. Ser.*, vol. 1993, n.º 1, p. 012008, ago. 2021, doi: 10.1088/1742-6596/1993/1/012008.

[15] L.-B. Chen, J.-C. Ju, C.-C. Wang, y I.-J. Huang, «HPChecker: An AMBA AHB On-Chip Bus Protocol Checker with Efficient Verification Mechanisms», *IEICE Trans. Inf. Syst.*, vol. E93-D, n.º 8, pp. 2100-2108, 2010, doi: 10.1587/transinf.E93.D.2100.

[16] A. Gupta, K. Rawat, S. Pandey, P. Kumar, S. Kumar, y H. P. Singh, «Physical design implementation of 32-bit AMBA ASB APB module with improved performance», en *2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT)*, Chennai, India, mar. 2016, pp. 3121-3124. doi: 10.1109/ICEEOT.2016.7755276.

[17] M. Mitic y M. Stojcev, «An overview of on-chip buses», *Facta Univ. - Ser. Electron. Energ.*, vol. 19, n.º 3, pp. 405-428, 2006, doi: 10.2298/FUEE0603405M.

[18] «Somaraju et al. - 2014 - DESIGN AND IMPLEMENTATION OF THE ADVANCED MICROCON.pdf».

[19] B. N. Manu y P. Prabhavathi, «Design and implementation of AMBA ASB APB bridge», en *2013 International Conference on Fuzzy Theory and Its Applications (iFUZZY)*, Taipei, Taiwan, dic. 2013, pp. 234-238. doi: 10.1109/iFuzzy.2013.6825442.

[20] «AXI Basics 1 - Introduction to AXI».
https://support.xilinx.com/s/article/1053914?language=en_US
[21] «Shrivastav et al. - 2011 - Performance Comparison of AMBA Bus-Based System-On.pdf».
[22] «AXI4-Lite Interface-Introduction to AXI4-Lite».
https://www.realdigital.org/doc/a9fee931f7a172423e1ba73f66ca4081
[23] «AMBA AXI and ACE Protocol Specification Version E».
https://developer.arm.com/documentation/ihi0022/e/AMBA-AXI4-Lite-Interface-Specification/AMBA-AXI4-Lite/Interoperability?lang=en