INTERFAZ SPI PARA CONTROL DEL DAC TLV5627 C/L BASADO EN FPGA

FPGA BASED SPI INTERFACE FOR CONTROLLING THE DAC TLV5627 C/L

Darío Martínez Nava
CICATA - Instituto Politécnico Nacional, México
dmartinezn2016@gmail.com

Antonio Hernández Zavala
CICATA - Instituto Politécnico Nacional, México
anhernandezz@ipn.mx

Recepción: 16/septiembre/2019 **Aceptación:** 30/octubre/2019

Resumen

Al utilizar e interconectar un número cada vez mayor de dispositivos digitales de control (Por la filosofía industria 4.0, las Smart Cities, vehículos inteligentes y por la introducción de la generación eléctrica basada en energías renovables) se vuelve necesaria la implementación de protocolos de comunicación que reduzcan o no incrementen el número de hilos necesarios para comunicarse. Actualmente los protocolos de comunicación serial en uso más importantes entre elementos son el RS-232, RS-485, SPI, I2C y en últimas instancias, el I3C de mayor velocidad que el I2C y UNI/O de un único hilo de Microchip.

El ciclo de *time-to-market* acelerado de las empresas, la reducción de costos de fabricación y mantenimiento, han convertido los FPGA en una en la herramienta de desarrollo de prototipos casi tan importante como los microcontroladores. Con estas dos premisas en mente se presenta la implementación de una interfaz SPI en un FPGA para el control de un Convertidor Analógico Digital (DAC por sus siglas en inglés) ahondando en las ventajas y desventajas que conlleva este desarrollo propio.

Palabras Clave: Diseño Embebido, FPGA, Interfaz SPI, Metodología V.

Abstract

Due to an increasing usage and interconnection of digital control systems (due to industry 4.0 philosophy, Smart Cities, Smart vehicles and renewable energy based electricity generation) the need for communication protocols that reduce, or at least retain the number of wires also increased. Today, the most important serial protocols in use by sensors, controllers and actuators alike are the RS-232, RS-485, SPI and I2C. Recently Microchip has developed an UNI/O interface requiring only 1 wire and an I3C protocol, able to work faster than the I2C and the single wire UNI/O developed by Microchip.

Driven by a time-to-market accelerated cycle, and the need to cut down manufacturing and maintenance costs, many enterprises have chosen an FPGA for fast development and prototyping platform almost as important as a microcontroller. It is with the two previous premises in mind that an FPGA based SPI interface for controlling a Digital-to-Analog Converter is presented, emphasizing on the advantages and disadvantages inherent to its development.

Keywords**:** *Embedded Design, FPGA, SPI Interface, V Methodology*.

1. Introducción

Desde la existencia de las primeras máquinas hubo siempre la necesidad de controlar numerosos parámetros de las mismas mediante palancas, eslabones y engranes para regular su comportamiento. El ejemplo clásico de esto es el gobernador de velocidad mecánico de James Watt para su máquina de vapor [Katsuhiko, 2010], del cual James Clerk Maxwell derivó sus ecuaciones diferenciales dando inicio a la disciplina de control automático [Kang, 2016].

Más recientemente, con el auge de la electrónica y microprocesadores estas conexiones fueron gradualmente reemplazadas por conductores y mandos eléctricos [Franklin & Powell, 1980; Katsuhiko, 1996] ejemplo de los cuales se puede encontrar en la mayoría de aeronaves de última generación tanto militares como civiles conocido como mandos fly-by-wire, de manera muy evidente, casi intuitiva, en los vehículos autónomos, y menos evidentes pero no menos numerosos en los vehículos de la vida diaria. El problema del crecimiento de la interconexión de

dispositivos es que la cantidad de información que debe ser generada, distribuida (mediante algún protocolo de comunicación) y procesada independientemente del modelo de control (Centralizado o Distribuido) [McCarthy, 2014].

El intercambio de información entre los diferentes elementos del sistema se puede realizar esencialmente de dos formas, paralelo o serie. Las evidentes desventajas de la distribución en paralelo radican en la cantidad de hilos conductores de información que requiere, sin embargo, su alta velocidad de transferencia los hace preferibles en ciertas aplicaciones. En el caso de las conexiones en serie, la velocidad de transferencia se ve mermada considerablemente dependiendo del número de bits de la trama a transmitir, sin embargo, economiza el trazado sobre tarjetas de circuito impreso, y pines requeridos para la comunicación, así como la cantidad de cables requeridos.

Toda información transferida en forma seriada requiere, aparte de un protocolo, un hardware que reconozca las diferentes fases y modos de comunicación que se implementarán en el protocolo. Por supuesto que cada quien puede realizar un protocolo propio para intercomunicación, sin embargo, los componentes disponibles en el mercado están sujetos a una serie de estándares previamente especificados. A manera de ejemplo el protocolo I2C creado por Phillips [Anderson, Rutkowski, & Oehler, 2009] y actualmente adoptado, bajo distintos nombres, por numerosas empresas fabricantes de semiconductores.

La información es pues, codificada en una **trama de bits**, común a los dispositivos interconectados. La sincronía es otro factor importante a tener en cuenta, ya que existen protocolos síncronos y asíncronos los cuales contienen o carecen respectivamente de una línea exclusivamente dedicada a transmitir una **señal de reloj**. El hardware debe entonces ser capaz tanto de colocar tramas en un bus, como de leerlas del mismo.

El hardware donde se implementa esta interfaz se conoce como Arreglo de Compuertas Programables en Campo (**FPGA** por sus siglas en inglés) la expresión “Programable en Campo” hace referencia a que los elementos lógicos dentro del **FPGA** no están conectados de forma fija, si no que el diseñador tiene la libertad de conectarlos a voluntad [Moore & Wilson, 2017] después de haber salido éste de la

línea de producción. A manera de contraste, un microcontrolador tiene conexiones internas fijas e inalterables pues la memoria de programa no puede separarse eléctricamente del microprocesador. Actualmente el diseño de hardware a la medida es altamente valorado por Google, Amazon [Morra, 2016] y últimamente Microsoft [Microsoft, 2018] quienes han pasado a abarcar del desarrollo de software, a la construcción de sus propios servidores personalizados. Los beneficios del uso de esta tecnología son 3: reduce costos, aumenta la eficiencia energética y aumenta la velocidad [Microsoft, 2018].

Dado que lo que se realiza es una descripción de hardware se debe conocer a fondo el comportamiento de lo que se desea implementar, pues a diferencia de un programa convencional, escrito ya sea en lenguaje ensamblador, C, C++ o algún otro lenguaje de alto nivel que devuelve una serie de códigos ejecutados secuencialmente uno a uno, la descripción de hardware implementa, mediante lenguaje **VHDL** o **Verilog** [Moore & Wilson, 2017], circuitos concurrentes ya sea de tipo combinatorio o secuencial. Esto hace que distintas secciones de la descripción de hardware pueden estar funcionando al mismo tiempo. De igual forma, la descripción de hardware debe ser "Sintetizada" el equivalente de la "Compilación" de un lenguaje orientado a microcontroladores o microprocesadores, dando como resultado un circuito (**núcleo**) que realiza la función deseada. Aunado a ello, no todas las instrucciones disponibles en **VHDL** son sintetizables.

Existen numerosas empresas dedicadas a la venta de núcleos para acelerar el ciclo de desarrollo de un hardware en particular para acelerar el ciclo de desarrollo. Lo que se entrega, por lo general, es la hoja de datos del núcleo adquirido y un archivo encriptado [ARROW, 2019] que contiene la descripción de hardware, un modelo de simulación, restricciones de tiempo y enrutamiento, interfaz de configuración y control de versiones según las métricas de calidad de Intel disponibles para sus núcleos [Intel, 2019]. Estos núcleos se conocen como núcleos de propiedad intelectual o (núcleos IP por sus siglas en inglés) todo esto por un costo, según se publican en AVNET [AVNET, s/f] y ARROW [ARROW, 2019], que oscila desde los 500USD hasta más de 250kUSD, que deben añadirse al precio a pagar por las suites de desarrollo de alguno de los dos principales productores de **FPGA's** del

mercado Xilinx o Altera, aunque si el diseño no es tan complejo, existen versiones gratuitas desde las páginas de las compañías.

La interfaz propuesta en este trabajo, permite reducir los costos asociados a los núcleos IP mediante el desarrollo propio tanto monetarios como de recursos utilizados en el **FPGA**. Cabe resaltar que, aunado a lo anterior, se propone la filosofía **HUM** (*Do It Yourself*, por sus siglas en inglés) como posible beneficio tanto económico como social para las empresas del giro de desarrollo tecnológico. Proporciona un caso de aplicación tangible para la metodología V proponiendo el caso de una interfaz **SPI** para el control de un Convertidor Digital Analógico (**DAC** por sus siglas en inglés).

El escrito que se presenta consta de seis secciones, en la introducción se presenta la problemática a abordar con el trabajo, así como una brevísima introducción a la interfaz **SPI** y la estructura interna de un **FPGA**. Posteriormente se discute detalladamente la metodología V y cómo esta se aplica al diseño de interfaz **SPI**. Los resultados muestran la interfaz **SPI** implementada y las pruebas de la misma presentando los recursos utilizados del **FPGA** los tiempos de retraso, las señales de comunicación con el **DAC** y el comportamiento del mismo. La discusión aborda la experiencia obtenida en el diseño de la interfaz con un enfoque hacia la industria tecnológica y las posibles bondades e inconvenientes del diseño en **FPGA**. Finalmente, las conclusiones resumen el trabajo presentado y cualidades del mismo.

La interfaz **SPI** [DeLuca & Herold, 1987] es una interfaz síncrona de, formalmente, cuatro líneas y salvo casos muy específicos, de 3 líneas. Las líneas de comunicación se definen, como **MOSI**, **MISO**, **SCK** y **SS** o alternativamente como **DO**, **DI**, **CLK**, y **CS** dependiendo del fabricante las cuales se encargan de las actividades listadas en la tabla 1.

El ***FPGA*** internamente consta de elementos lógicos (LE por sus siglas en inglés), en el caso de los ***FPGA's*** de Altera, o bloques lógicos configurables (CLB por sus siglas en ingles), para el caso de Xilinx. Ambos, sin embargo, internamente están conformados por al menos 3 partes principales, una tabla de búsqueda, lógica de acarreo interna y un registro de salida el cual permite la creación de lógica síncrona.

Cada una de estas celdas lógicas son interconectadas entre celdas adyacentes o más lejanas mediante una malla de interconexión y, finalmente, a una celda de salida también configurable.

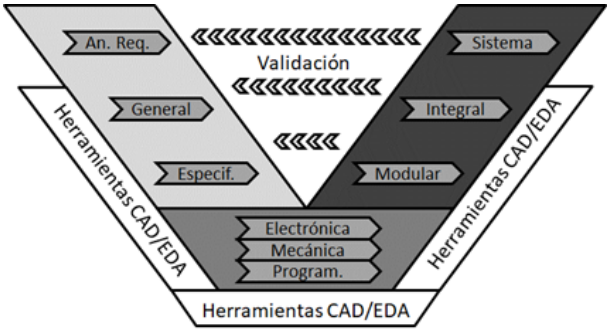
Tabla 1 Funciones de los pines de la interfaz SPI.

Línea	Sentido	Descripción
MOSI/DO	Maestro -> Esclavo	Salida de datos del transmisor (Maestro) al receptor (Esclavo)
MISO/DI	Esclavo -> Maestro	Salida de datos del receptor (Esclavo) al transmisor (Maestro). Utilizado para comunicación bidireccional.
SCK/CLK	Maestro -> Esclavo	Genera un tren de pulsos para el mecanismo de captura de datos del receptor. El Maestro se define como el dispositivo que controla esta línea
SS/CS	Maestro -> Esclavo	El pin habilita/inhibe el funcionamiento del mecanismo de captura de datos del receptor.

Fuente: Elaboración Propia.

2. Métodos

La elaboración de la descripción de hardware utilizando VHDL que se siguió para la elaboración de la interfaz es la Metodología V [Gausemeier & Moehringer, 2002] [Vasić & Lazarević, 2008] [Martinez N, Sosa S, & Hernández Z., 2018]. En este punto se debe hacer énfasis que el flujo de trabajo involucrado en la fase de implementación de la Metodología V corresponde al indicado por el Entorno Integrado de Síntesis 14.7 de Xilinx [Xilinx, 2009] (ISE 14.7 por sus siglas en inglés) por lo que dicho flujo de trabajo se describirá en la respectiva etapa de implementación de la Metodología V. La Metodología V consta de tres fases principales: Diseño, Implementación y Pruebas, figura 1.

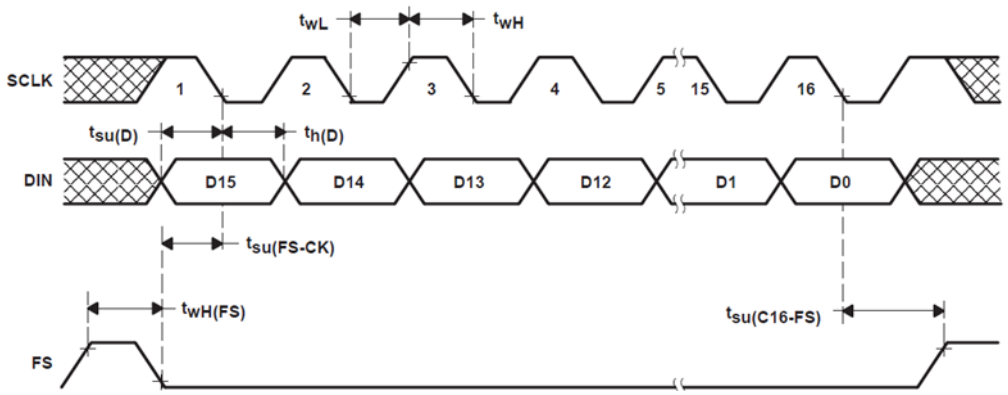


Fuente: Elaboración Propia.

Figura 1 Metodología V detallada.

Fase 1 diseño

- *Etapa de análisis de requerimientos.* Se recopilan los datos para la construcción de la interfaz según el protocolo de comunicación especificado en la hoja de datos del periférico TLV5627 C/L, así como también las tareas básicas a llevar a cabo por la interfaz. En la figura 2 se muestra el diagrama de tiempos para las señales de control.



Fuente: Texas Instruments.

Figura 2 Diagrama de tiempos para las señales de control del DAC.

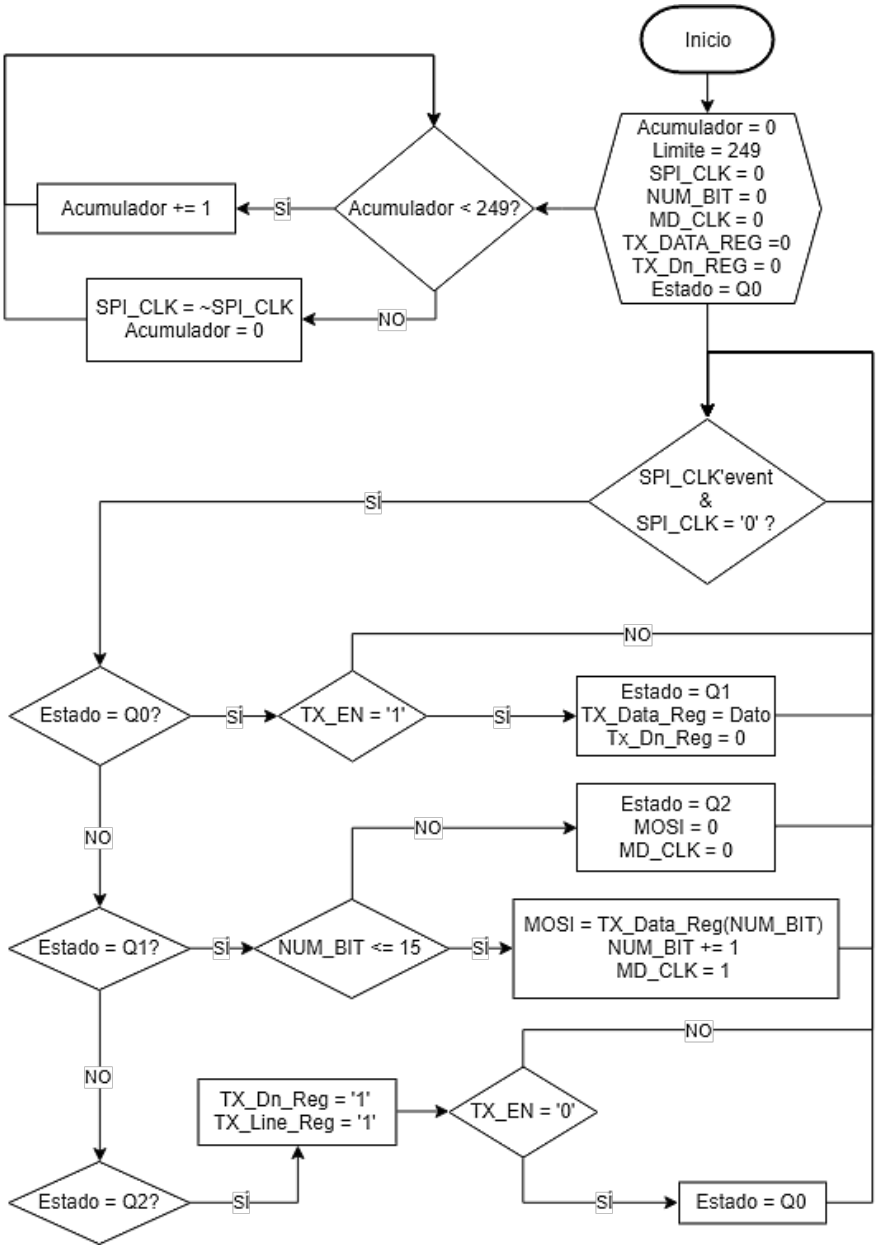
- *Etapa de diseño general.* Se proponen las entradas y salidas de una "caja negra" así como su ancho en bits que, a juicio del diseñador, dotan a la interfaz la funcionalidad planteada (tabla 2).

Tabla 2 Entradas y salidas de los dispositivos a conectar.

Elemento	*Nombre*	*Tipo*	*Magnitud (V)*
Interfaz SPI	DAC_CH_CONF	E	3.3
	DAC_DATA		
	SAM_RDY		
	SYS_CLK		
	RESET		
	DAC_LINE (MOSI)	S	
	DAC_SCLK (SCLK)		
	F_SYNC (FS)		
TLV5627	DIN (MOSI)	E	DVDD(1)
	SCLK		
	FS		
	CS		GND(2)

Fuente: Elaboración Propia con datos de Texas Instruments

- *Etapa de diseño específico.* Habiendo fijado las entradas y salidas, con las señales propuestas en la etapa anterior se procede a proponer una estructura interna basada en un diagrama de flujo como se muestra en la figura 3 que tome las entradas propuestas y genere las salidas deseadas según los datos del cronograma de la figura 3 y agrupándolos en la tabla 3 finalizando así, la fase de diseño.



Fuente: Elaboración Propia.

Figura 3 Diagrama de flujo funcional.

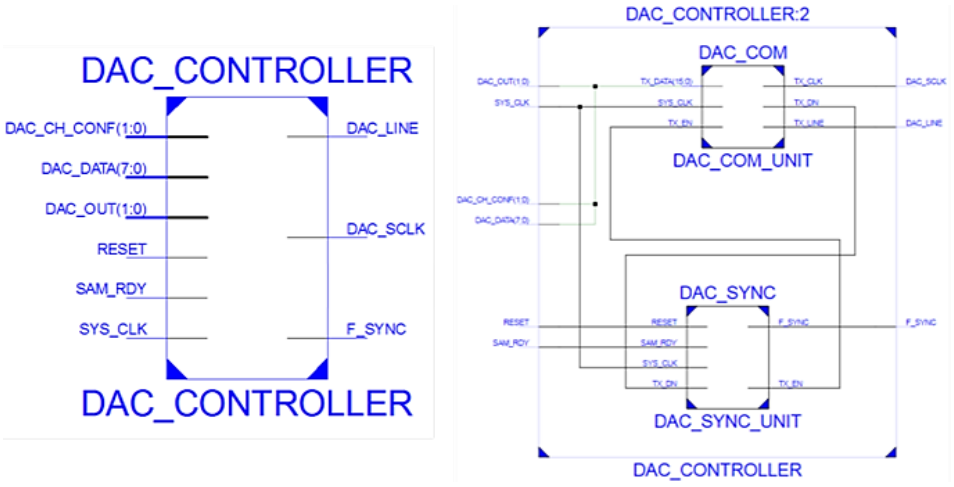
Tabla 3 Requerimientos de tiempo para las señales de entrada.

Parámetro	*Condición*	*Mínimo (ns)*
$t_{su(FS-CK)}$	Tiempo para $FS = L$ antes del primer flanco bajada de SCLK	8
$t_{su(C16-FS)}$	Tiempo para que $FS = H$ desde el muestreo del dieciseisavo bit	10
t_{wH}	Duración de pulso, $SCLK = H$	25
t_{wL}	Duración de pulso, $SCLK = L$	25
$t_{su(D)}$	Datos listos antes de $SCLK \downarrow$	8
$t_{h(D)}$	Persistencia después de $SCLK \downarrow$	5
$t_{wH(FS)}$	Duración de pulso, $FS = H$	20

Fuente: Texas Instruments.

Fase de implementación

- *Síntesis.* Se genera la descripción en VHDL de la interfaz SPI y se sintetiza en el entorno integrado de síntesis, obteniendo un modelo de simulación. Como resultado de esta etapa se pueden observar, la figura 4a para la etapa del diseño general y la figura 4b para la etapa del diseño específico.



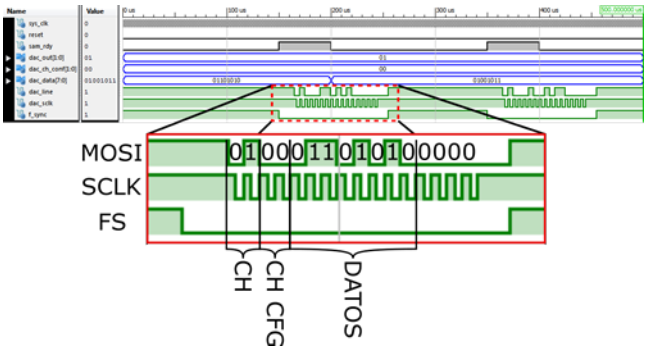
a) General. b) Específico.

Fuente: Elaboración Propia.

Figura 4 Diseño de la interfaz SPI.

- *Simulación.* Se toma el modelo de simulación generado por el sintetizador y se describe entonces un modelo de simulación accionando todas las señales de entrada y monitoreando las señales de salida del modelo sintetizado. En este punto puede ocurrir que el diseñador requiera introducir o pueda eliminar

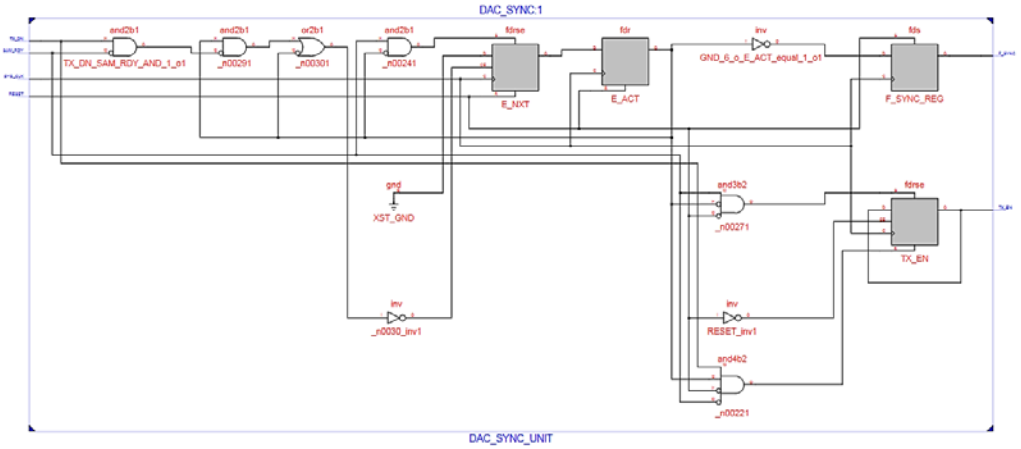
ciertas señales de entrada o salida que no fue posible prever en la fase de diseño, por lo que en caso de ser necesario se regresa a las correspondientes etapas de diseño general/específico y se itera las veces necesarias. En la figura 5 se muestra el cronograma interfaz SPI.



Fuente: Elaboración Propia.

Figura 5 Cronograma interfaz SPI.

- *Implementación:*
  - ✓ <u>Traducción.</u> Genera un archivo de diseño que toma las restricciones físicas del modelo (Pines físicos de entrada/salida deseados, así como la corriente, velocidad de conmutación y tiempos de respuesta) previamente especificadas por el usuario y genera un modelo de diseño basado en los elementos básicos de construcción disponibles en la base de datos asociada al FPGA seleccionado, figura 6.



Fuente: Elaboración Propia.

Figura 6 Traducción de la máquina de estados de la interfaz SPI.

- ✓ Mapeo. Toma el modelo de diseño con los elementos básicos de construcción y selecciona los bloques lógicos configurables y celdas de entrada/salida que cumplan con las restricciones físicas del diseñador, obteniendo la descripción física del hardware a utilizar basado en los recursos disponibles del FPGA seleccionado.
- ✓ Colocar y Conectar. Dado que pueden existir o no numerosos bloques lógicos que, por su cercanía o lejanía, cumplan o incumplan uno o más restricciones dictadas por el diseñador, virtualmente se selecciona un grupo de bloques lógicos configurables, se conecta y analiza. Si se cumplen todas las restricciones del diseñador la etapa termina y se considera el diseño colocado y conectado creando un archivo para la generación del Bitstream, de lo contrario se itera nuevamente seleccionando un nuevo grupo de bloques lógicos configurables.
- ✓ Generación de archivo de configuración. Genera un código de configuración legible por el FPGA para adaptar la lógica interna según lo dicta el archivo de colocación y conexión.
- ✓ Configuración de dispositivo. Se conecta el FPGA y se descarga de la computadora el archivo de configuración, con lo cual el FPGA se considera “programado” y listo para evaluar físicamente el hardware descrito desde el sintetizador.

Fase de pruebas

- *Pruebas de módulo.* Se verifica el correcto funcionamiento de la interfaz y se compara con los resultados de la etapa de simulación en la fase de implementación. Al ser satisfactorio se procede a realizar una prueba de integración, conectando el DAC a ser controlado con el FPGA que contiene la interfaz SPI. De lo contrario se verifica el diseño en específico de la interfaz.
- *Pruebas de integración.* Se verifica la funcionalidad deseada para la interfaz SPI y que efectivamente, se están transfiriendo datos al DAC. Si el DAC no responde por lo general se puede deber a dos cuestiones, no se cableó adecuadamente la conexión entre el FPGA y el DAC o se omitió una señal

de control importante en el diseño general (y con ello el diseño específico de hardware dedicado a controlar dicha señal)

- *Pruebas de sistema.* Se comprueba el comportamiento de la interfaz en situaciones de error.

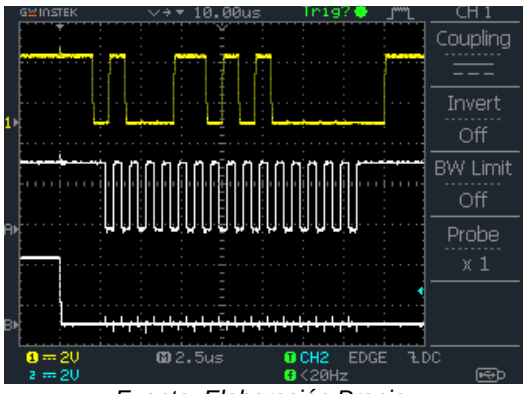
3. Resultados

Se obtienen, como resultado de la metodología V, así como del flujo de trabajo del Entorno Integrado de Síntesis, la tabla de entradas tanto de la interfaz SPI como del convertidor DAC y salidas igualmente, de la interfaz como del convertidor, además de una tabla con los requerimientos en cuanto a la secuencia con tiempos de aplicación de las señales de control y un diagrama para ilustrar la medición de los mismos.

El diagrama funcional de la interfaz SPI, la "caja negra" y arquitectura interna correspondientes al diseño general y específico respectivamente.

El resultado de la síntesis del código de descripción de hardware en VHDL de la máquina de estados, así como la simulación del funcionamiento de la interfaz SPI como parte esencial de la fase de implementación aplicando el flujo de trabajo del Entorno Integrado de Síntesis 14.7.

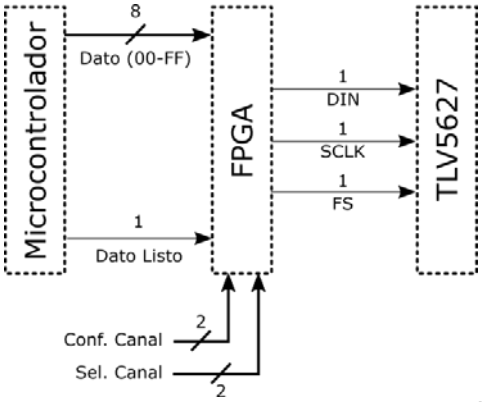
La captura de pantalla del osciloscopio utilizando el dato "01101010" de forma manual (figura 7), muestra las diferentes señales de la interfaz coincidiendo con la respuesta del simulador y verificando el comportamiento físico en el FPGA real para la etapa de pruebas modulares de la fase de pruebas.



Fuente: Elaboración Propia.

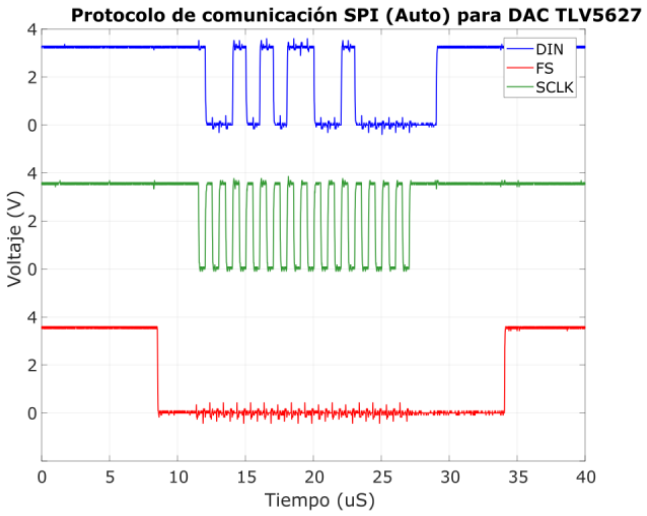
Figura 7 Prueba de implementación modular.

Posteriormente se muestra otra prueba con el DAC conectado y enviando datos de una onda sinusoidal de forma automática mediante un microcontrolador conectado según el diagrama mostrado en la figura 8, comprobando la respuesta inalterada del protocolo que se muestra en la figura 9 mediante datos extraídos del osciloscopio.



Fuente: Elaboración Propia.

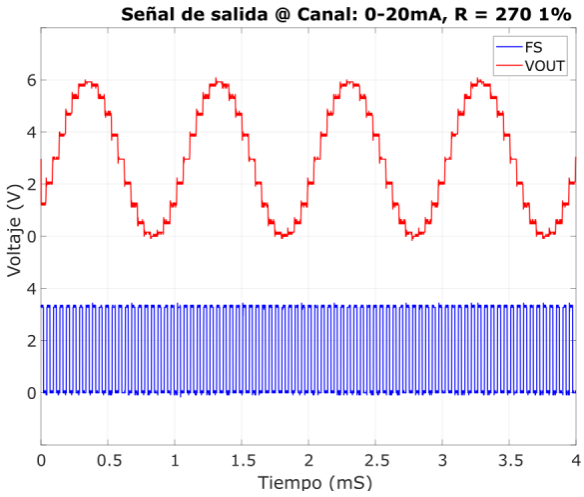
Figura 8 Conexión para onda sinusoidal.



Fuente: Elaboración Propia.

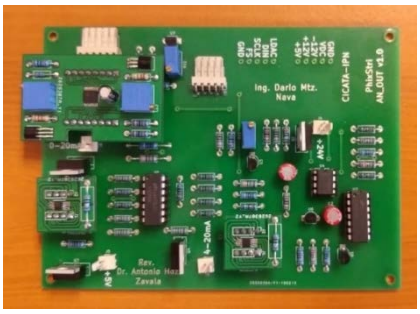
Figura 9 Prueba de interconexión con DAC TLV5627.

Finalmente se muestra la onda sinusoidal resultante (figura 10), enviada del micro controlador al DAC mediante la interfaz SPI en el FPGA adaptada mediante un circuito convertidor de 4 canales seleccionando la salida de 0-20 mA con una resistencia de carga de 270 Ω, figura 11.



Fuente: Elaboración Propia.

Figura 10 Señal sinusoidal comunicada al DAC mediante la interfaz SPI en FPGA.



Fuente: Elaboración Propia.

Figura 11 Interfaz analógica de 4 canales con TLV5627.

En la tabla 4 se pueden apreciar los recursos lógicos del FPGA que se utilizaron de acuerdo a lo que devuelve el sintetizador para implementar la interfaz. Mientras que la tabla 5 muestra el tiempo máximo que tarda una señal en llegar de un lugar a otro dentro de la interfaz SPI diseñada.

Tabla 4 Recursos lógicos utilizados.

Device Utilization Summary			[-]
Slice Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice Registers	35	126,800	1%
Number of Slice LUTs	27	63,400	1%
Number of occupied Slices	14	15,850	1%
Number of bonded IOBs	36	210	17%

Fuente: Elaboración Propia.

Tabla 5 Tiempos de retardo de señal.

En Lógica		En Ruta		Total	Niveles Lógicos
nS	%	nS	%	nS	3
0.125	9.6%	1.180	90.4%	1.305	

Fuente: Elaboración Propia.

4. Discusión

El núcleo aquí presentado se puede utilizar únicamente como transmisor de tamaño reducido para controlar un convertidor DAC y portable a otros dispositivos debido a que se diseñó utilizando lenguaje VHDL, el cual, con algunas modificaciones, puede transmitir tramas de más o menos bits siguiendo la metodología de diseño V. Sus aplicaciones pueden resultar limitadas, sin embargo, esto mismo lo hace extremadamente eficiente en el consumo de recursos del FPGA. Existen métricas para evaluar el desempeño de un diseño digital, tales como Velocidad, Consumo, Tamaño, Costo, Tiempo a prototipo, Tiempo a comercialización, Conformidad, Mantenimiento, Seguridad, Violabilidad y Confiabilidad [Cayssials, 2014].

En el ámbito de los negocios resaltan el ciclo de tiempo a comercialización y el tiempo a prototipo de las empresas tecnológicas los cuales obedecen a la necesidad de innovar más rápido que la competencia para permitir su subsistencia en mercados altamente competidos. Así, los núcleos de propiedad intelectual son ventaja competitiva respecto de los competidores siempre que la empresa tenga la solvencia económica que conlleva el uso y licenciamiento [Kirk, 2007].

Por otra parte, dado que lo que se proporciona al usuario final, en esencia es un código distribuido por internet [ARROW, 2019; Intel, 2019], para las empresas dedicadas al desarrollo de estos núcleos, se reducen enormemente los gastos relacionados con la producción, distribución y comercialización de los mismos. En tanto que los usuarios, en esencia los desarrolladores de aplicación de una compañía de giro tecnológico, solo deben conocer los parámetros configurables del mismo e incluirlo en su diseño.

Un ejemplo más claro se puede dar con las plataformas Arduino, Raspberry PI, y numerosas otras plataformas de código abierto, en las cuales no es necesario tener

nociones de programación para controlar algún dispositivo por la enorme disponibilidad de sus librerías. De esta forma, la empresa de giro tecnológico se libera de tener un departamento de desarrollo de núcleos propios pues el objetivo de la empresa es producir e innovar y no generar capital humano especializado para independizarse del consumo de tecnología ajena a ella. Con esto la compañía ahorra una considerable cantidad de dinero, posiblemente mayor que el mismo costo de adquirir núcleos permanentemente en cada uno de sus nuevos diseños.

Pedagógicamente hablando, cuestionando, exponiendo, debatiendo, esencialmente interactuando y entrando en contacto con lo que se requiere aprender existiendo varias actividades como las planteadas en [Mavrikios, Papakostas, Mourtzis, & Chryssolouris, 2013]. Por ello, mientras más expuesto esté un diseñador a problemas que supongan generar tecnología propia este adquiere un conjunto de habilidades [Mavrikios et al., 2013] que le permiten volverse más eficiente en su trabajo, así como proponer alternativas de diseño e implementación, métodos de trabajo etc. Generando la compañía desarrollos propios se pueden mejorar las habilidades de su plantilla laboral, aunque, con un costo elevado en tiempo y dinero haciendo más grande sus ciclos de tiempo al mercado y tiempo al prototipo, posiblemente arriesgándose al rezago frente a sus competidores. Por lo que, para una exitosa sinergia entre la actualización de la plantilla laboral y el ciclo de tiempo para mercado o prototipo, resulta fundamental balancear la adquisición de núcleos propios y la generación de propiedad intelectual. Además, la filosofía detrás del diseño digital basado en FPGA es la creación de hardware a la medida contrastando contra los núcleos disponibles y a la venta que resultan ser altamente flexibles lo cual, a menudo, se ve reflejado en el tamaño de los mismos a la hora de ser implementados en el FPGA [Intel, 2017] en comparación con un núcleo específicamente diseñado para una función en particular. Haciendo prácticamente inviable la implementación de estos núcleos genéricos en diseños con requerimientos más estrictos.

Es aquí donde la empresa tecnológica tiene el beneficio de gozar con una plantilla que conoce y es capaz de implementar cualquier hardware eficientemente y que además comprende los sistemas lo suficientemente a fondo como para poder

brindar capacitación a los nuevos colegas tanto en las herramientas de trabajo como en las metodologías.

5. Bibliografía y Referencias

[1] Anderson, A., Rutkowski, J., & Oehler, D. (2009). United States Patent Application Publication: Edge Rate Control For I2C Bus Applications (US Pat. Num. US 2009/0066381 A1).

[2] ARROW (2019). Embedded Systems: IP Core (30 de julio del 2019): https://www.arrow.com/en/categories/programmable-devices/embedded-systems/ip-cores.

[3] AVNET (s/f). Programmable Logic: IP Cores (30 de julio del 2019): www.avnet.com/shop/us/c/programmable-logic/ip-cores/?pageNumber=1&pageSize=60.

[4] DeLuca, J. S., & Herold, B. W. (1987). Communication Protocol Between Master and Slave Device With Register Information Sharing (U.S. Pat. Num. 5630152).

[5] Cayssials, R. (2014). Sistemas Embebidos en FPGA. (D. Fernández, Ed.) (1a ed.). Buenos Aires: Alfaomega.

[6] Franklin, G. F., & Powell, J. D. (1980). Digital Control of Dynamic Systems. Phillipines: Adison-Wesley Publishing Company.

[7] Gausemeier, J., & Moehringer, S. (2002). VDI 2206- A New Guideline for the Design of Mechatronic Systems. IFAC Proceedings Volumes, 35(2), 785–790. https://doi.org/10.1016/S1474-6670(17)34035-1

[8] Intel (2017). Intel FPGA 16550 Compatible UART Core: FPGA Resource Usage: www.intel.com/content/www/us/en/programmable/documentation/sfo1400787952932.html#iga1405707226213.

[9] Intel. (2019). Intel IP Quality Metrics: Deliverables (30 de julio del 2019): https://www.intel.com/content/www/us/en/programmable/products/intellectual-property/ip/interface-protocols/m-alt-seriallite3.html.

[10] Katsuhiko, O. (1996). Sistemas de Control en Tiempo Discreto.

[11] Katsuhiko, O. (2010). Ingeniería de Control Moderna (5ta ed.). Pearson.

[12] Kang, C. G. (2016). Origin of Stability Analysis: “On Governors” by J.C. Maxwell. IEEE Control Systems Magazine, 36(5), 77–88. https://doi.org/ 10.1109/MCS.2016.2584358.

[13] Kirk, B. (2007). The True Cost of Free FPGA IP (31 de julio del 2019): https://www.electronicdesign.com/fpgas/true-cost-free-fpga-ip.

[14] Martinez N, D., Sosa S, J. C., & Hernández Z., A. (2018). Empleo de la metodología V en sistemas embebidos: módulo controlador de menús basado en FPGA para una pantalla LCD gráfica. XVII Congreso Nacional de Ingeniería Electromecánica y de Sistemas 2018, 7.

[15] Mavrikios, D., Papakostas, N., Mourtzis, D., & Chryssolouris, G. (2013). On industrial learning and training for the factories of the future: A conceptual, cognitive and technology framework. Journal of Intelligent Manufacturing, 24(3), 473–485. https://doi.org/10.1007/s10845-011-0590-9

[16] Morra, J. (2016). Amazon Plugs Xilinx FPGA into its Cloud. Electronic Design: Technologies, FPGAs: https://www.electronicdesign.com/fpgas/ amazon - plugs-xilinx-fpga-its-cloud.

[17] McCarthy, D. (2014). Choosing between centralized and distributed control system designs. Control Engineering Magazine and Newsletters: www.controleng.com/articles/choosing-between-centralized-and-distributed-control-system-designs/.

[18] Microsoft. (2018). Project Catapult (30 de julio del 2019): https://www.microsoft.com/en-us/research/project/project-catapult/.

[19] Moore, A., & Wilson, R. (2017). FPGA for dummies. (J. Binham, Ed.) (2nd Intel). New Jersey: John Wiley & Sons Inc.

[20] Vasić, V. S., & Lazarević, M. P. (2008). Standard Industrial Guideline for Mechatronic Product Design. FME Transactions, 36, 103–108.

[21] Xilinx. (2009). ISE Design Flow Overview (30 de julio del 2019): https://www.xilinx.com/support/documentation/sw_manuals/xilinx11/ise_c_fpga_design_flow_overview.htm.