

**Universidad Nacional
de San Martín**

PROYECTO FINAL INTEGRADOR

Crono TDC: Diseño e implementación de un Time to Digital Converter en FPGA

Estudiante
Julián Nicolás Rodríguez

Legajo
CyT 7269

Ingeniería Electrónica

Escuela de Ciencia y Tecnología
Universidad Nacional de San Martín

Director Ing. Nicolás Alvarez
Co-Director Dr. Federico Izraelevitch

Fecha
Abril 2023

Agradecimientos

Voy a aprovechar esta sección del informe para agradecer a todas las personas que me ayudaron a avanzar en mi carrera. Si bien el proyecto lo ejecuté individualmente, hubo mucha gente que me ayudó durante estos años para que llegue hasta este punto.

En primer lugar, agradezco a mi universidad. La UNSAM fue el lugar donde desperté mi pasión por la ingeniería y donde pude formarme como profesional. Lugar donde compartí muchos mates en el pastito con gente muy importante para mí, de donde me llevo buenos amigos y recuerdos hermosos. Gracias a mis compañeros y amigos, los cuales me acompañaron durante este camino, dentro y fuera del aula.

Agradezco a mis directores, los cuales confiaron en mí para llevar adelante este proyecto. Mención especial para Nico, que siempre estuvo dispuesto para evacuar dudas y ayudarme en cualquier cosa que se me iba presentando.

Gracias a Belu, mi compañera en esta aventura, por bancarme todos estos años. Tuvimos la suerte de estudiar juntos durante la mayor parte de nuestras carreras, y estoy seguro que sin nuestras tardes y noches de estudio hoy no estaría presentando este informe.

Al pilar fundamental de mi vida, mi familia. Gracias a mi hermano, Diego, me inscribí en esta carrera que hoy en día me apasiona tanto. Gracias a mis viejos, que son mi ejemplo a seguir y me brindaron todo lo que tenían a su alcance para que pueda progresar en esta carrera.

Índice general

1. Introducción	2
1.1. <i>Time to Digital Converters</i>	2
1.2. TDCs en el mercado	3
1.2.1. Medición de tiempo de vuelo	3
1.2.1.1. LiDAR	4
1.2.1.2. PET	5
1.2.2. Física de partículas	6
1.3. Motivación del proyecto	7
1.4. Estructura del proyecto	8
1.5. Flujo de trabajo y herramientas	9
1.6. Resumen	10
2. Arquitectura del core	11
2.1. Arquitectura de TDC	11
2.2. <i>Top level</i> del core	14
2.3. Operación del TDC mediante comandos	15
2.4. Resumen	16
3. Módulos del core	17
3.1. Canales del TDC	17
3.1.1. ISERDESE	18
3.1.2. Decodificador de pulsos	19
3.1.3. RAM	21
3.2. Lógica de procesamiento y control	23
3.2.1. Procesador de comandos	24
3.2.2. Ejecutor de comandos	25
3.2.2.1. Lectura de registros de configuración	27
3.2.2.2. Escritura de registros de configuración	28
3.2.2.3. Medición de los canales de entrada	29
3.2.2.4. Lectura de los canales de entrada	32
3.3. Procesador de tramas HDLC-lite	37
3.3.1. Transmisor HDLC-lite	38
3.3.2. Receptor HDLC-lite	42
3.4. Resumen	47
4. Diseño de PCB	48
4.1. Ancho de banda del sistema	48
4.1.1. Ancho de banda de las señales a medir	48
4.1.2. Ancho de banda mínimo del hardware	49
4.2. Requerimientos	49
4.3. Diagrama en bloques	50
4.4. Alimentación	52
4.5. Configuración de la FPGA	53
4.5.1. Modo de configuración JTAG	54
4.5.2. Modo de configuración Master SPI	54
4.6. Puente UART-USB	56
4.7. Canales del TDC	57
4.7.1. Estándar VITA 57.1	58
4.7.2. Conector FMC	59
4.7.3. Banco 34 y 35	61
4.8. Señales de reloj externas	61
4.9. Especificación CubeSat Kit PCB	62

4.10. <i>Stackup</i>	62
4.11. Resultado final del diseño	63
4.12. Resumen	64
5. Prueba de concepto	65
5.1. Generador de pulsos arbitrarios	65
5.1.1. Prueba de laboratorio	66
5.2. Ensayos con la cadena completa	71
5.3. Utilización de recursos de la FPGA	75
5.4. Resumen	76
6. Discusión y conclusión	78
6.1. Discusión	78
6.2. Trabajo futuro	80
6.3. Conclusión	80
A. Esquemáticos e instrucciones de fabricación	82
A.1. Esquemáticos del PCB desarrollado	82
A.2. Instrucciones de fabricación del PCB	107
B. Diseño de la fuente de alimentación de la FPGA	109
B.1. Diseño de la fuente de alimentación	110
B.1.1. Frecuencia de conmutación	112
B.1.2. Secuenciado	113
B.1.3. Configuración de punto de regulación	113
B.1.4. Elección del inductor	113
B.1.5. Ciclo de trabajo	114
B.1.6. Elección de los capacitores de entrada	115
B.1.7. Elección de los capacitores de salida	116
B.1.8. Estimación del <i>ripple</i> en la tensión de salida	117
B.1.9. Encendido suave	119
B.1.10. Modo de operación	119
B.1.11. Conmutación en fase y desfase	119
B.1.12. Simulación en LTPowerCAD	120
B.2. FPGA Power Distribution System	126
B.2.1. Composición del PDS	127
B.2.2. Simulación en LTSpice	128
C. Notas sobre el lazo de compensación de la fuente <i>buck</i>	130
D. Detalle de las conexiones VITA 57.1	135

Glosario

- AC** Corriente Alterna. 112
- ADC** Conversor Analógico Digital (o *Analog to Digital Converter* en inglés). 54
- AES** *Advanced Encryption Standard*. 52
- AFE** *Analogic Front-End*. 5, 7, 58
- ASIC** *Application-Specific Integrated Circuit*. 12
- BGA** *Ball Grid Array*. 63
- BRAM** *Block RAM*. 22, 23, 76
- CI** Integración Continua. 9
- COTS** *Commercial Off-The-Shelf*. 62
- CRC** Verificación por Redundancia Cíclica (o *Cyclic Redundancy Check* en inglés). 14, 15, 37–40, 44, 45, 47, 79
- CSKB** *CubeSat Kit Bus*. 8, 50–52, 62
- CTR** *Coincidence Time Resolution*. 6
- DC** Corriente Directa. 112, 115, 117, 118, 126, 128
- DGPS** *Differential Global Positioning System*. 4
- DNL** No-Linealidad Diferencial. 12, 78
- EEPROM** *Electrically Erasable Programmable Read-Only Memory*. 135
- ENIG** *Electroless Nickel Immersion Gold*. 63
- ESD** Descarga Electroestática (o *Electrostatic Discharge* en inglés). 57
- ESL** *Equivalent Series Inductance*. 127, 128
- ESR** *Equivalent Series Resistance*. 117, 118, 127, 128
- FMC** *FPGA Mezzanine Card*. 1, 8, 48, 51, 52, 54, 58, 59, 61, 63, 64, 79, 80, 135
- FPGA** *Field Programmable Gate Array*. 1, 2, 6, 8, 12, 13, 17, 18, 22, 23, 47–57, 61–65, 76–80, 109–111, 126–128, 135
- FSM** Máquina de Estados Finitos (o *Finite State Machine* en inglés). 19, 20, 29, 30, 32, 38, 40–43
- GPIO** *General Purpose Input/Output*. 65, 67
- HASL** *Hot Air Solder Leveling*. 63
- HDL** Lenguaje Descriptor de Hardware (o *Hardware Description Language* en inglés). 9, 50, 61
- HDLC** *High-level Data Link Control*. 14
- HDMI** *High-Definition Multimedia Interface*. 66
- HPC** *High Pin Count*. 58
- I2C** *Inter-Integrated Circuit*. 135

- IMU** *Inertial Measurement Unit*. 4
- IO** Entrada/Salida (o *Input/Output* en inglés). 51, 52, 56, 58, 61, 62, 64, 109, 135
- ISERDESE** *Input SERializer DESErializer*. 1, 13, 14, 17–20, 22, 23, 47, 61, 73, 75, 78–80
- JTAG** *Joint Test Action Group*. 50, 51, 53, 54, 135
- LEO** Órbita Baja Terrestre (o *Low Earth Orbit* en inglés). 7
- LFSR** *Linear Feedback Shift Register*. 38
- LiDAR** *Light Detection and Ranging or Laser Imaging Detection and Ranging*. 4, 7, 78
- LoR** Línea de Respuesta (o *Line of Response* en inglés). 5, 6
- LPC** *Low Pin Count*. 58
- LUT** *Look Up Table*. 24, 75–77, 79
- LVDS** *Low Voltage Differential Signaling*. 1, 7, 8, 48–50, 57, 59, 61, 63, 64, 79, 80, 135
- MEMS** Sistemas Microelectromecánicos. 61
- MLCC** *Multi-layer Ceramic Capacitor*. 127
- MRCC** *Multi-Region Clock Capable I/O*. 61
- OBC** *On Board Computer*. 62, 79, 80
- PCB** *Printed Circuit Board*. 1, 8, 9, 48–54, 57–59, 61–64, 79, 80, 82, 107, 127, 135
- PDS** *Power Distribution System*. 51, 126–128
- PET** Tomografía por Emisión de Positrones. 5, 78
- PWM** *Pulse Width Modulation*. 130
- QSPI** *Quad Serial Peripheral Interface*. 56
- RAM** Memoria de Acceso Aleatorio (o *Random-Access Memory* en inglés). 12, 14, 16, 17, 20–23, 34, 35, 52
- RMS** Media Cuadrática (o *Root Mean Square* en inglés). 115
- RTL** *Register-Transfer Level*. 1, 65, 78, 80
- SiPM** Fotomultiplicadores de Silicio (o *Silicon PhotoMultipliers* en inglés). 6, 7, 80
- SoC** *System On Chip*. 6, 66
- SPI** *Serial Peripheral Interface*. 53–56
- SRCC** *Single-Region Clock Capable I/O*. 61
- TAC** Conversor de Tiempo a Amplitud (o *Time to Amplitude Converter* en inglés). 6
- TAP** *Test Access Port*. 54
- TDC** *Time to Digital Converter*. 1–12, 14–17, 21–24, 27, 28, 37, 47, 48, 50, 57–62, 64, 65, 71–73, 75–80, 109, 135
- TMDS** Señal Diferencial de Transición Minimizada (o *Transition Minimized Differential Signaling* en inglés). 66, 67

ToF Tiempo de Vuelo (o *Time of Flight* en inglés). 3–5, 78

TVS Supresor de Voltaje Transitorio (o *Transient Voltage Suppressor* en inglés). 57

UART *Universal Asynchronous Receiver-Transmitter*. 15, 51, 56

USB *Universal Serial Bus*. 1, 51, 56, 57

VDL *Vernier Delay Line*. 11

Resumen

En este trabajo se presenta el proyecto Crono TDC, el cual consiste en el diseño, implementación y validación de un *Time to Digital Converter* (TDC) con la capacidad de medir eventos de hasta 5 ns . Estos dispositivos son utilizados para medir intervalos de tiempo pequeños (en el orden de los nanosegundos o picosegundos). A lo largo del proyecto, se diseñó el *Register-Transfer Level* (RTL) de un TDC basado en múltiples fases de reloj utilizando circuitos *Input SERIALizer DESERIALIZER* (ISERDESE) los cuales son primitivos de la *Field Programmable Gate Array* (FPGA) en la cual se realizó la implementación. Debido a las características propias de su arquitectura, este TDC puede poseer una gran cantidad de canales.

El *core* se diseñó para ser operado con comandos enviados a través de un puerto serie utilizando un protocolo de entramado determinado. Ciertos parámetros del TDC pueden ser configurados con estos comandos, como por ejemplo, la ventana de tiempo de la medición.

Durante el desarrollo del proyecto, se ejecutaron ensayos en los cuales se inyectaron señales conocidas con pulsos de 5 ns en las entradas del TDC y luego se reconstruyeron a partir de la información almacenada en su memoria. En estas pruebas, el TDC se sincronizó con una frecuencia de reloj de 100 MHz , logrando una resolución de $2,5\text{ ns}$ y un error de cuantización de 5 ns .

Además, se diseñó un *Printed Circuit Board* (PCB) con una FPGA XC7A35T-2CSG324I el cual posee 19 canales *Low Voltage Differential Signaling* (LVDS) expuestos en un conector *FPGA Mezzanine Card* (FMC) de alta velocidad, dos osciladores de 100 MHz y una interfaz *Universal Serial Bus* (USB) 2.0. Dicho *hardware* respeta la especificación *CubeSat Kit PCB* [1] lo cual sienta las bases para un diseño de TDC que puede ser utilizado en Cubesats.

1. Introducción

En este capítulo se expone al lector el marco en el cual se desarrolla el proyecto. En sus secciones se incluye un breve marco teórico del dispositivo diseñado, donde se detalla su principio de funcionamiento. También, se comenta la motivación del desarrollo y se explican los objetivos del proyecto, sus entregables y los resultados esperados. Por último, se describe la metodología de trabajo y las herramientas utilizadas a lo largo del desarrollo.

1.1. Time to Digital Converters

Un TDC es un dispositivo que reconoce eventos y provee una representación digital del tiempo en el cual ocurren. El objetivo del mismo es medir el intervalo de tiempo entre una señal de inicio y una señal de fin.

En un sistema digital, el acercamiento más básico para llevar a cabo este cometido es contar la cantidad de ciclos de reloj que transcurren entre el comienzo de la señal de inicio y de la señal de fin. Este proceso se puede visualizar en la Figura 1.1, donde el intervalo de tiempo medido, $\Delta T_{medido} = NT_{clk}$, es un múltiplo del período de reloj utilizado

Pero, en el caso más general, las señales de inicio y de fin pueden ser asincrónicas a la frecuencia de reloj del sistema, por lo tanto, el intervalo de interés puede definirse como $\Delta T = \Delta T_{medido} + \Delta T_{inicio} - \Delta T_{fin}$. Como puede observarse, este tipo de circuito posee un inevitable error de cuantización $\Delta T_{inicio} - \Delta T_{fin}$, cuya cota máxima es $2T_{clk}$. El error de cuantización puede reducirse aumentando la frecuencia del reloj, pero esto no es escalable, dado que al aumentar la frecuencia de reloj aumenta el consumo del dispositivo y, además, las FPGAs poseen una frecuencia máxima a la cual pueden operar.

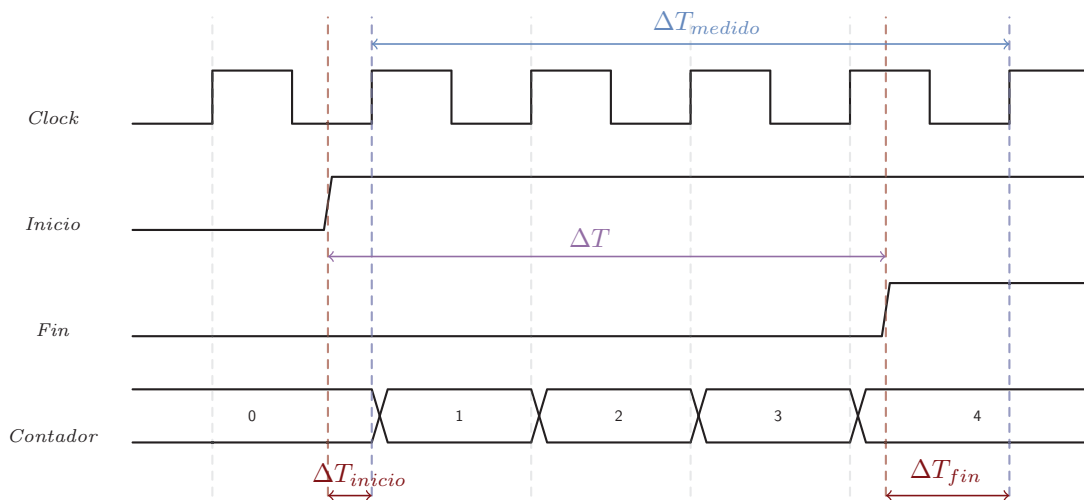


Figura 1.1: Diagrama de tiempos de una implementación de un contador de ciclos de reloj.

Un método viable para disminuir esta incertidumbre y aumentar la resolución es subdividir el período de reloj en intervalos más pequeños. Los sistemas que utilizan esta técnica están compuestos por un contador fino y un contador grueso. El contador grueso es sencillamente un contador de ciclos de reloj, como en la Figura 1.1, y el contador fino cuenta intervalos de tiempo dentro de un mismo ciclo de reloj. Esto se ilustra en la Figura 1.2, donde el contador fino puede realizar hasta 4 cuentas en un mismo ciclo de reloj. El contador fino sirve para estimar en que momento dentro de un ciclo de reloj cambian de estado las señales de inicio y de fin, lo que permite reducir el error de cuantización.

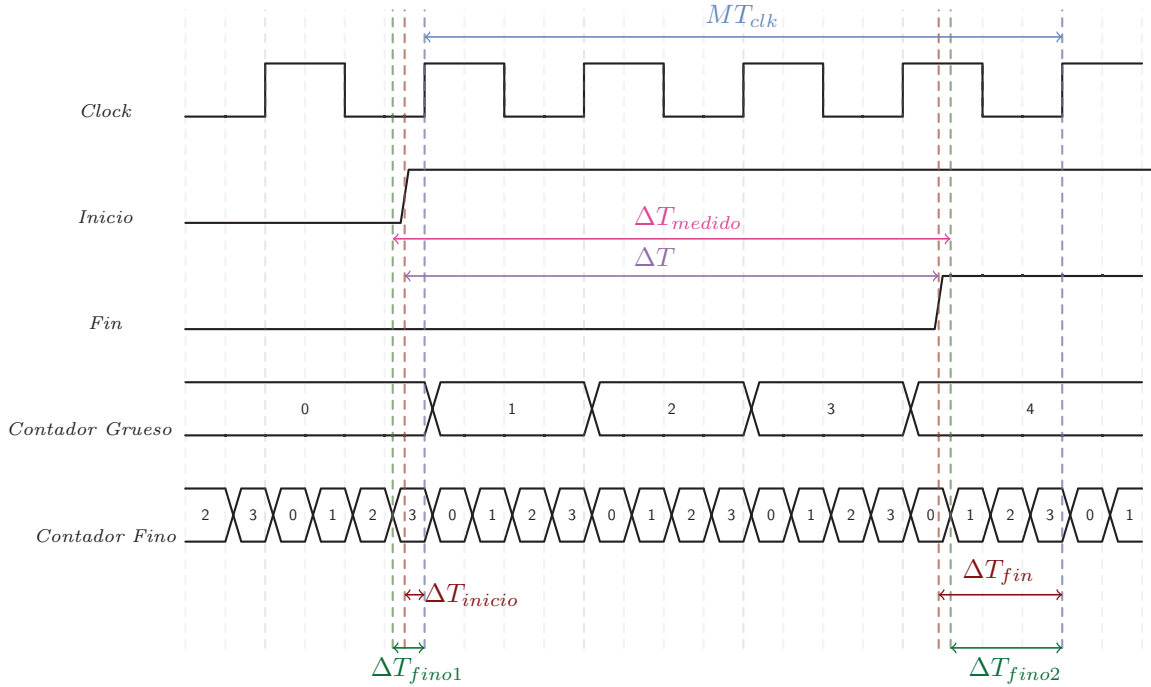


Figura 1.2: Diagrama de tiempos del funcionamiento básico de un TDC, donde se puede observar la operación de un contador grueso y un contador fino.

Utilizando este arreglo de contadores se puede estimar el intervalo de tiempo de interés (ΔT) con la Ecuación 1.1.

$$\Delta T_{medido} = MT_{clk} + \Delta T_{fino1} - \Delta T_{fino2} \quad (1.1)$$

En este caso, la incertidumbre en la medición es dos veces la mínima cuenta del contador fino, como se puede observar en la Ecuación 1.2, donde T_{clk} es el período de reloj y N es la cantidad de subdivisiones en el mismo.

$$\Delta T_{err} = 2 \frac{T_{clk}}{N} \quad (1.2)$$

Existen varias maneras de implementar este tipo de sistemas de medición. En particular, el interés del presente proyecto consiste en utilizar la arquitectura *multi-phase based clock* TDC, en la cual se utilizan distintas fases de una misma señal de reloj para implementar el contador fino. El funcionamiento de dicha arquitectura se describe en la Sección 2.1 y su implementación en la Sección 3.1.

1.2. TDCs en el mercado

Los TDC son dispositivos que se utilizan en diversos campos, como por ejemplo, en física de partículas y en tecnología médica. En esta sección se explorarán tres aplicaciones de estos dispositivos para informar al lector de sus posibles usos. El objetivo de dos de ellas es la medición del Tiempo de Vuelo (o *Time of Flight* en inglés) (ToF) de un haz de luz para determinar una distancia. La otra aplicación, un poco más científica, consiste en la detección individual de fotones.

1.2.1. Medición de tiempo de vuelo

En las mediciones de ToF, se utiliza la velocidad de la luz como referencia para determinar distancias. Estos sistemas constan de un emisor y un receptor de luz, cuyo objetivo es medir el tiempo que tarda un haz de luz en recorrer una distancia determinada. En su forma más simple, estos sistemas

estiman la distancia utilizando la Ecuación 1.3 [2], donde c representa la velocidad de la luz y ΔT es el intervalo de tiempo que tarda la luz en recorrer la distancia Δx .

$$\Delta x = c \frac{\Delta T}{2} \quad (1.3)$$

1.2.1.1. LiDAR

Un caso particular de medición de ToF son los sistemas *Light Detection and Ranging* or *Laser Imaging Detection and Ranging* (LiDAR), utilizados en vehículos autónomos para medir distancias a objetivos cercanos y en sensado remoto para realizar mapeos 3D de la Tierra (típicamente desde un satélite o un vehículo aéreo).

Este apartado se enfoca en introducir los sistemas LiDAR utilizados para mapeos terrestres, los cuales pueden ser utilizados para, por ejemplo, crear mapas topográficos de zonas naturales e identificar daños causados por desastres naturales.

Estos sistemas poseen un escaner láser, una unidad de detección de rango, una unidad de control y monitoreo, un *Differential Global Positioning System* (DGPS) y una *Inertial Measurement Unit* (IMU). Un diagrama conceptual de este tipo de sistemas se puede observar en la Figura 1.3.

El principio de funcionamiento es el siguiente: el vehículo sobrevuela la zona de interés, emite un haz de luz y mide el intervalo de tiempo que tarda este haz en llegar a un objetivo y volver hacia el vehículo. Luego, utilizando la Ecuación 1.3, calcula la distancia entre el vehículo y el objetivo; y con la ayuda de la instrumentación que posee (DGPS e IMU), estas mediciones son georeferenciadas a un sistema de coordenadas.

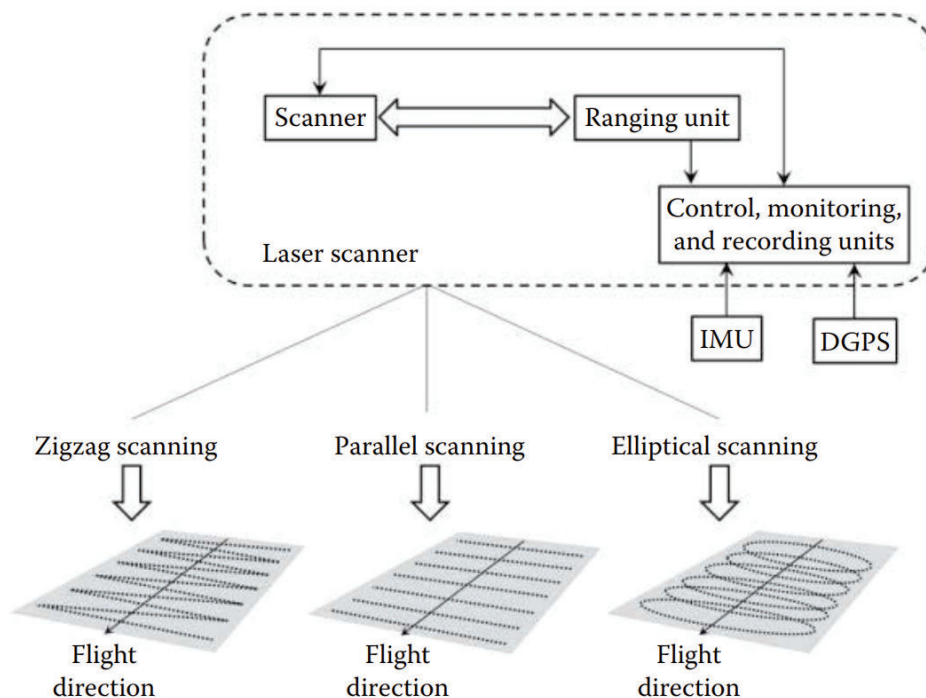


Figura 1.3: Esquema conceptual de un sistema LiDAR aéreo para mapeo terrestre. Extraído de [3].

A modo de referencia, para la realización de mapeos de superficies planas con pendientes de baja a moderada inclinación, se requiere una precisión mínima de 15 cm [3]. Utilizando la Ecuación 1.3, esto se traduce en un intervalo de tiempo de aproximadamente 1 ns . Existen TDC comerciales especialmente diseñados para estas aplicaciones, como por ejemplo el TDC7200 de Texas Instruments [4], el cual posee una resolución de 55 ps .

1.2.1.2. PET

Otro caso de aplicación de los TDC para mediciones de ToF es la Tomografía por Emisión de Positrones (PET) [2]. Esta tecnología se utiliza para determinar la ubicación y el tamaño de tumores.

En PET se suministra al paciente un radiofármaco el cual decae debido a su configuración nuclear inestable. Por lo general, estos radiofármacos son moléculas similares a la glucosa que tienden a acumularse en los tumores dado que las células cancerosas son metabólicamente más activas que el resto.

Durante el proceso de decaimiento de estos radioisótopos, se generan positrones los cuales viajan y colisionan con electrones pertenecientes a átomos que se encuentran cerca del proceso de aniquilación. Esta aniquilación genera dos rayos gamma con una energía de 511 keV y una separación de 180° . El dispositivo posee cristales que absorben y convierten la energía de estos rayos gamma en fotones. Estos fotones luego son convertidos en una señal eléctrica utilizando sensores de fotones con sus respectivos *Analogic Front-End* (AFE), los cuales acondicionan la señal. Estas señales eléctricas luego son procesadas para reconocer cuales fueron los eventos generados por el mismo proceso de aniquilación, los cuales se encuentran en una misma Línea de Respuesta (o *Line of Response* en inglés) (LoR). Un esquema de este proceso se puede visualizar en la Figura 1.4.

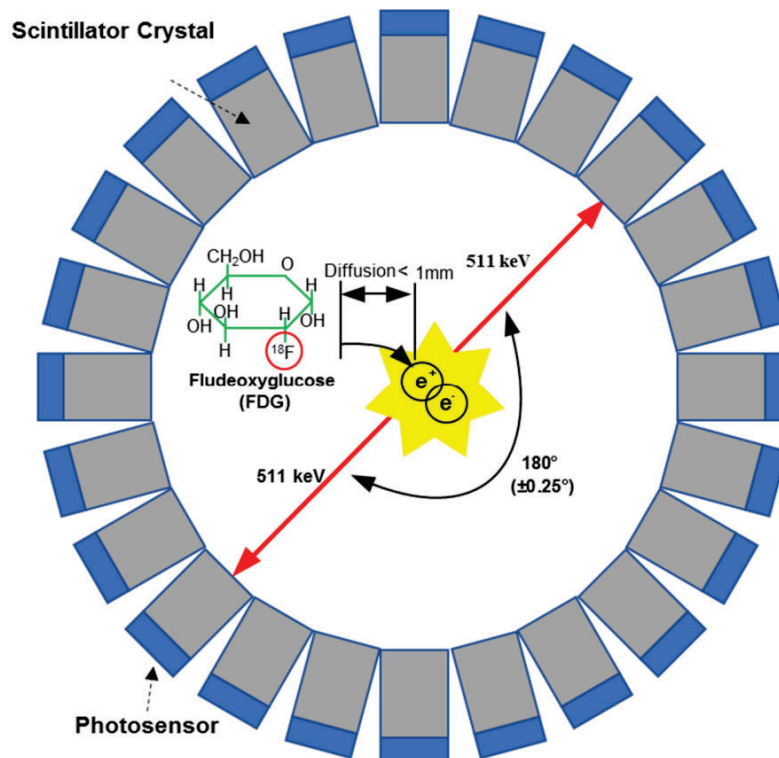


Figura 1.4: Principio de funcionamiento del PET: Un anillo de detectores que detectan un par de rayos gamma con una energía de 511 keV que resultan de la aniquilación de un electrón con un positrón emitido por un radiofármaco. Extraído de [5].

El objetivo del PET es determinar con la mayor precisión posible la ubicación en la cual ocurrió el proceso de aniquilación, dado que allí se encuentra el tumor. Con la medición de ToF se puede determinar en que zona de la LoR es más probable que haya ocurrido el evento de aniquilación. Los TDC en estos dispositivos miden la diferencia entre los tiempos de arribo de cada fotón en los detectores que se encuentran a lo largo de la LoR. Si no hubiera medición de ToF, se debería asignar la misma probabilidad de ocurrencia a toda la LoR.

En la Figura 1.5 se puede observar el efecto de la medición de ToF en estos sistemas. La distancia Δx desde el centro de la LoR a la ubicación del evento de aniquilación se puede estimar utilizando la

Ecuación 1.3.

La mínima variación en los tiempos de arribo de los fotones de una misma LoR se define como *Coincidence Time Resolution* (CTR) y en los equipos utilizados actualmente se encuentra en el orden de los 500 ps [6].

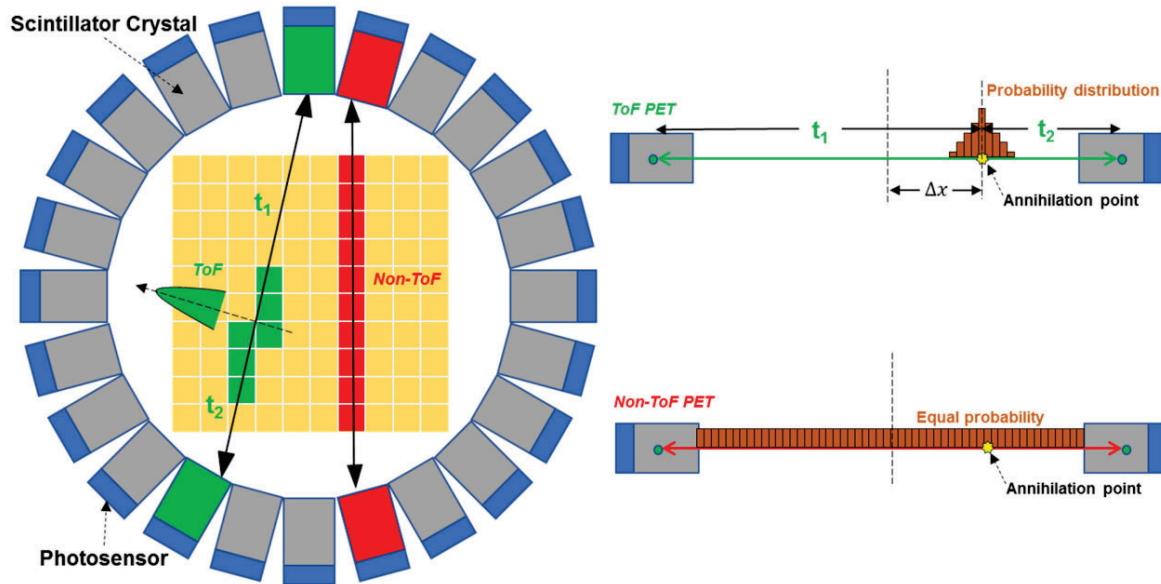


Figura 1.5: Concepto de la medición de tiempo de vuelo en PET. Si no hubiera medición de tiempo de vuelo la probabilidad de ocurrencia del evento de aniquilación a lo largo de una misma línea de respuesta es constante. Pero, al utilizar la tecnología de medición de tiempo de vuelo, se puede asignar una probabilidad mayor a la zona en la cual ocurrió dicho proceso. Extraído de [5].

1.2.2. Física de partículas

Dentro del campo de la física de partículas, se encuentran aplicaciones en las cuales los TDC son utilizados para detectar y contar partículas individuales, como por ejemplo, fotones.

En el artículo [7] se describe el desarrollo de un TDC de 8 canales cuyo objetivo es detectar señales provenientes de la absorción de fotones. En el experimento descrito en dicho artículo, utilizan un dispositivo que genera un pulso de aproximadamente 10 ns al detectar un fotón. El TDC fue implementado en el *System On Chip* (SoC) Zynq-7020 de Xilinx y se obtuvo una resolución de 22 ps.

También, en el trabajo de grado [8] se implementó un TDC en una FPGA Virtex-5 de Xilinx para medir el decaimiento de un muón. Se comparó la medición del tiempo de decaimiento del TDC con una medición realizada con un Conversor de Tiempo a Amplitud (o *Time to Amplitude Converter* en inglés) (TAC), y ambas mediciones resultaron cercanas.

Por otro lado, existe instrumental de laboratorio específico para estos propósitos, como la serie *Time Tagger* de Swabian Instruments [9]. En la nota de aplicación [10] se comenta la utilización de uno de estos equipos (*Time Tagger X*) para contar fotones provenientes de un detector de fotones EOS SNSPD de la empresa Single Quantum [11]. También, en el reporte [12] se utiliza el *Time Tagger Ultra* para medir el ancho de los pulsos provenientes de Fotomultiplicadores de Silicio (o *Silicon PhotoMultipliers* en inglés) (SiPM) y a partir de dicho ancho pueden determinar la cantidad de fotones que impactaron en el sensor. Se puede remarcar en esta última nota de aplicación que el ancho de los pulsos provenientes del sensor SiPM se encuentra entre 1 ns y 3 ns.

Con la información recabada, se puede remarcar que los TDC utilizados en estas aplicaciones son soluciones personalizadas implementadas en FPGAs o instrumental de laboratorio especializado.

1.3. Motivación del proyecto

Los interesados en el proyecto pertenecen a un grupo de investigación llamado LabOSat. La necesidad del grupo de investigación reside en detectar intervalos de tiempo cuya magnitud está en el orden de las decenas de nanosegundos. Dicho grupo diseñó y fabricó un AFE el cual genera pulsos LVDS a partir de la excitación de sensores SiPM [13]. A partir del ancho de estos pulsos se puede inferir la cantidad de fotones incidentes.

El objetivo del proyecto Crono TDC es desarrollar un prototipo de un TDC que pueda medir estos pulsos, priorizando la escalabilidad del sistema en términos de la cantidad de canales de entrada. Estos pulsos se transmiten a través de interfaces LVDS, por lo cual, los canales de entrada del dispositivo finalmente implementado deben poseer esta interfaz. Como referencia, una medición de estas señales se puede visualizar en la Figura 1.6.

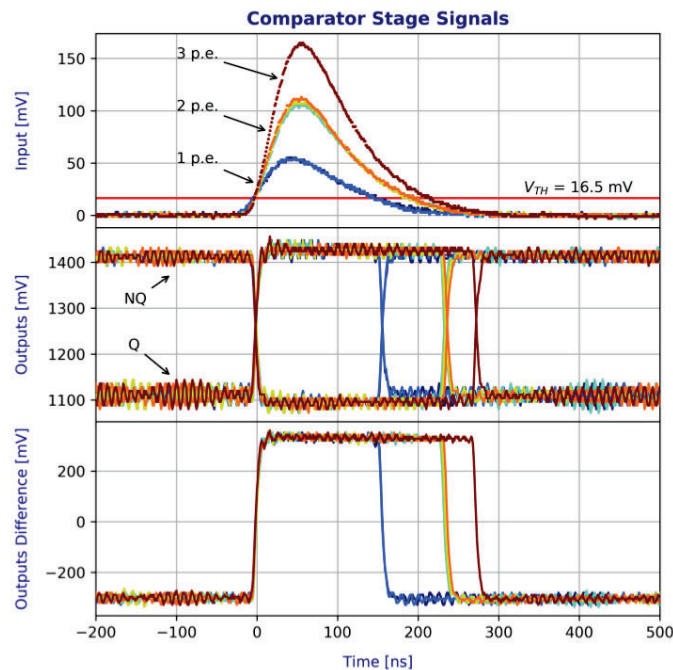


Figura 1.6: Señal de salida del AFE creado por el grupo LabOSat, donde se puede visualizar la respuesta del mismo cuando inciden fotones sobre los sensores. Extraído de [13].

Este laboratorio se dedica a investigar y desarrollar electrónica para aplicaciones espaciales. El principal interés del desarrollo de este TDC es utilizarlo para un futuro instrumento para satélites en Órbita Baja Terrestre (o *Low Earth Orbit* en inglés) (LEO) el cual pueda detectar y contar fotones utilizando sensores SiPM y un AFE diseñado por el mismo laboratorio [13]. Dicho instrumento sentaría las bases de la detección de fotones individuales en LEO y habilitaría el desarrollo de nuevas líneas de investigación, como por ejemplo instrumentos LiDAR para pequeños satélites.

Existen varias publicaciones donde se discute la utilización de tecnología LiDAR en pequeños satélites [14]. Entre las aplicaciones mencionadas, se encuentran mediciones de pequeñas distancias (por debajo de 1 *km*) y largas distancias (en el orden de los cientos de kilómetros).

Los LiDAR de corto alcance poseen un rango de hasta 1 *km*, lo cual es útil para la generación de imágenes 3D. Este tipo de sensado puede ser utilizado para maniobras orbitales en las cuales dos naves espaciales llegan a la misma órbita y se aproximan a una distancia muy cercana y para misiones dedicadas a capturar muestras de cometas u asteroides. En particular, en el artículo [15] se describe un LiDAR de estas características, el cual fue diseñado idealmente para misiones de reconocimiento y toma de muestras de asteroides.

Por otro lado, el rango de los LiDAR de largo alcance puede llegar a los miles de kilómetros

1.5. Flujo de trabajo y herramientas

Tanto en el desarrollo del código Lenguaje Descriptor de Hardware (o *Hardware Description Language* en inglés) (HDL), en la biblioteca de Python para comandar al TDC a través de un puerto serie y en el diseño del PCB se utilizó el *software* de control de versiones Git, utilizando la plataforma Gitlab.

El *core* del TDC [17] y el generador de pulsos [18] se describieron en Verilog. Los bancos de prueba se realizaron con el *framework* Cocotb, donde cada módulo descrito en Verilog posee un *testbench* asociado. Dado que el TDC y el generador de pulsos hacen uso común de varios módulos, se decidió almacenar estos últimos en otro repositorio [19] para poder ser importados como submódulos de Git.

Se crearon *pipelines* de Integración Continua (CI) en cada repositorio de Gitlab donde se ejecutan los *testbenches* de cada módulo automáticamente en cada *commit*. Si alguno de los *testbench* falla, dicho *pipeline* arroja un error, alertando de un problema introducido en dicho *commit*. Para esto, se creó una imagen de Docker [20] con las herramientas necesarias para ejecutar los *testbenches*, la cual es utilizada por el *runner* de Gitlab en cada *job*. Como el proyecto se estructuró en diversos módulos reutilizables, el *pipeline* de CI resultó fundamental para asegurar que al modificar alguno de ellos ningún otro resultaba perjudicado.

Se creó también una biblioteca de Python [21] la cual posee funciones que son útiles para diseñar *testbenches* en Cocotb. Los *testbenches* de los módulos de Verilog creados hacen uso de esta biblioteca, por lo cual, se encuentra instalada en la imagen de Docker base (*crono_hdl_ci* [20]).

Para poseer un ambiente de síntesis e implementación de HDL portable y repetible, se crearon *scripts* [22] (*bash* y *tcl*) para automatizar la generación de los proyectos de Vivado 2018.3. De esta manera, se provee un mecanismo sencillo con el cual se puede replicar el mismo ambiente utilizado durante el desarrollo.

Se creó una biblioteca de Python [23] cuyo objetivo es interactuar con el TDC a través de un puerto serie. Con esta biblioteca se pueden configurar registros del TDC, ejecutar mediciones y leer los datos almacenados. Esta biblioteca posee un ambiente de CI en donde se ejecutan los *tests* unitarios de la misma. En este repositorio también se construye una imagen de Docker en CI la cual contiene este paquete de Python instalado. Esto otorga una gran agilidad, dado que permite tener disponible un ambiente para interactuar con el TDC en cualquier *host* que soporte Docker.

Para diseñar el PCB se utilizó el *software* Altium 22.11.1. Si bien la mayoría de los archivos de este *software* son binarios y no es posible aprovechar al máximo el control de versiones que ofrece Git, se trabajó en un repositorio de Gitlab [24] para llevar adelante un proceso de diseño ordenado.

Las dependencias e interacciones de los repositorios se pueden visualizar en la Figura 1.8.

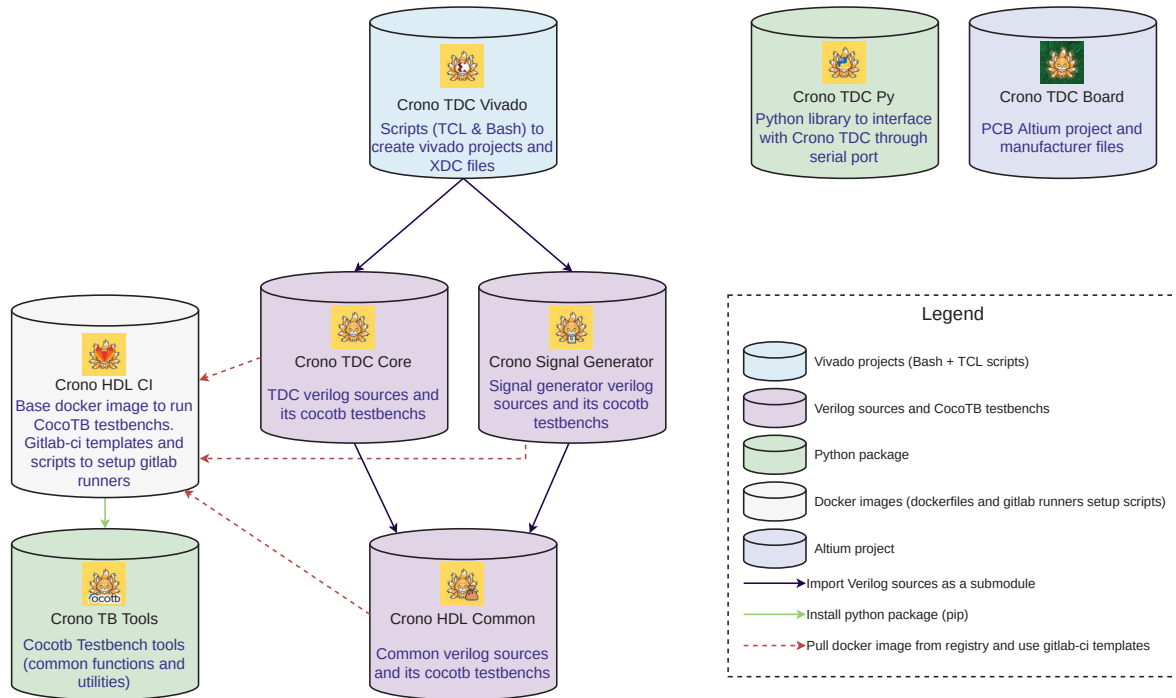


Figura 1.8: Organización de repositorios en los cuales se fue desarrollando el proyecto y sus respectivas interacciones.

1.6. Resumen

En este capítulo, se ha presentado una explicación detallada del principio de funcionamiento de los TDC, que son dispositivos ampliamente utilizados en distintas aplicaciones, como la medición de tiempos en experimentos científicos y en la industria de la comunicación. La comprensión de estos conceptos teóricos es esencial para el desarrollo del proyecto en cuestión. Asimismo, se han mencionado las motivaciones que dieron origen a este proyecto, destacando la importancia del mismo en el ámbito de la medición de tiempos y su posible aplicación en futuros proyectos.

En cuanto a la estructura del proyecto, se han descrito detalladamente tanto el enfoque como los objetivos específicos del mismo, con el fin de proporcionar una comprensión clara del alcance del proyecto y de las metas a alcanzar. Además, se ha presentado la estructura del informe, indicando cómo se han organizado los distintos capítulos y secciones, a fin de proporcionar una visión general del contenido del documento.

Por último, se han introducido las herramientas que se utilizaron durante el desarrollo del proyecto, como los repositorios donde se encuentra el código fuente y el proyecto de Altium.

A continuación, en el siguiente capítulo, se describirá la arquitectura del *core* de TDC diseñado, donde se presenta en más detalle la arquitectura escogida.

2. Arquitectura del *core*

En este capítulo se presenta la arquitectura general del *core* diseñado. En primer lugar, se realiza una descripción de la arquitectura de TDC escogida y se fundamenta su elección. Luego, se presenta el *top level* del *core*, enunciando cada una de sus partes y se describe el caso de uso típico para el cual el TDC fue diseñado.

2.1. Arquitectura de TDC

Existen varias arquitecturas de TDC, una de las más clásicas se denomina *Vernier Delay Line* (VDL) [25] [26], la cual utiliza retardos de propagación de elementos lógicos para medir el intervalo de tiempo entre la señal de inicio y de fin. Un diagrama conceptual de una entrada de este tipo de TDC se puede observar en la Figura 2.1, donde se observan las etapas n_i de las cadenas de retardo, cada una de ellas compuestas por dos *buffers* y un *flip flop*. La señal de inicio se propaga a través de una cadena de retardo (t_{d1}), mientras que la señal de fin se propaga por otra línea de retardo (t_{d2}) oficiando de señal de reloj para los *flip flops*.

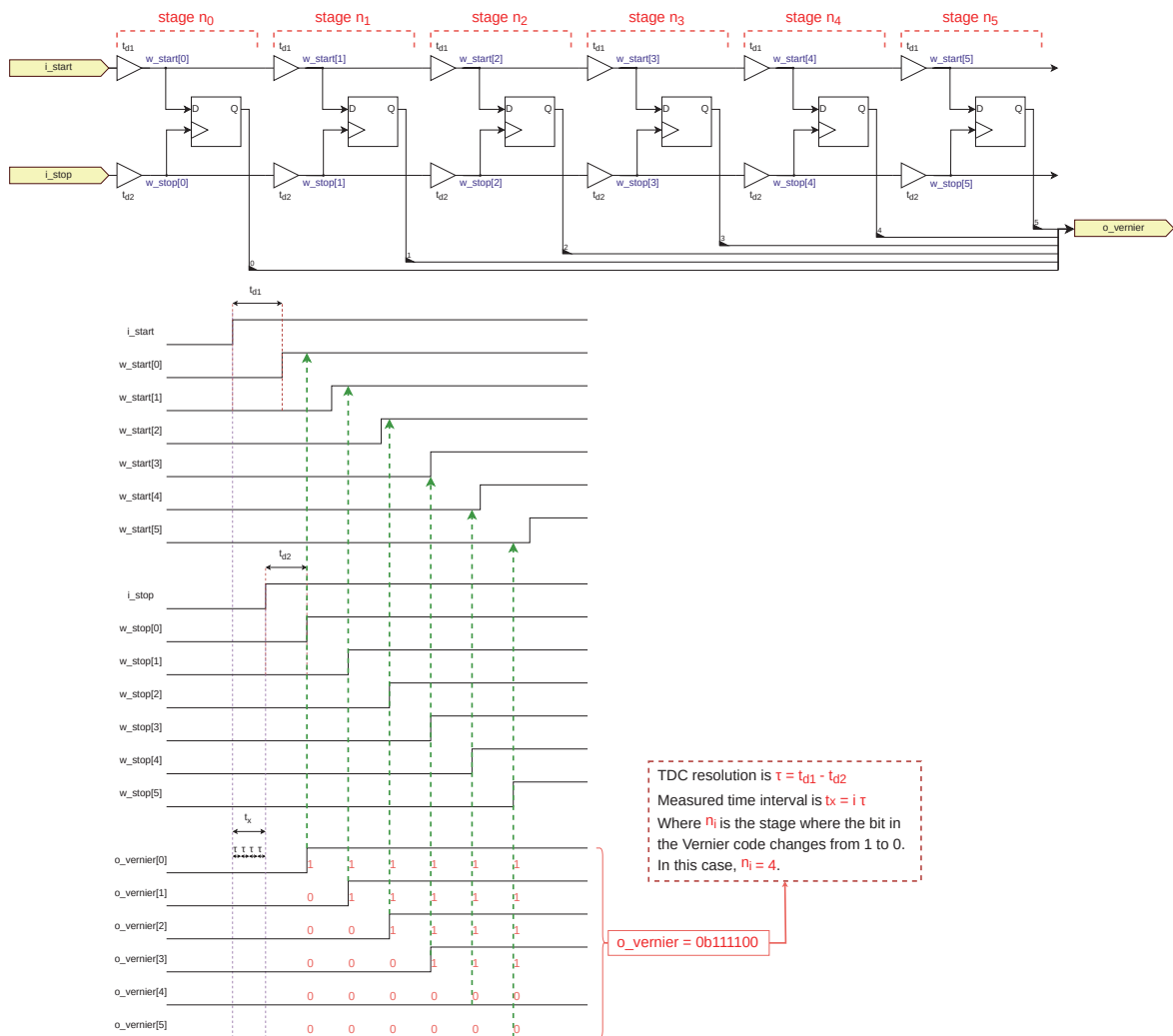


Figura 2.1: Principio de funcionamiento del TDC *Vernier Delay Line*. Adaptado de [27].

Por diseño, el tiempo de propagación t_{d1} es ligeramente mayor al tiempo de propagación t_{d2} [28]. Esto provoca que la diferencia de tiempo entre los arribos de las señales de inicio y de fin en cada una de las etapas decaiga a medida que se propagan a lo largo de la cadena.

La señal de fin oficia de señal de reloj para los *flip flops*, por lo cual, si arriba después de la señal de inicio, el valor de salida del *flip flop* será 1. En el caso contrario, si la señal de fin llega primero que la señal de inicio, la salida de este último será un 0. Esta mecánica produce un “código termómetro” a la salida de los *flip flops*, donde la cantidad de unos indican el intervalo de tiempo transcurrido entre la señal de inicio y de fin (t_x). El valor de este tiempo t_x se puede calcular con la Ecuación 2.1 [27], donde i es la etapa en la cual los bits cambian de 1 a 0 en el código y τ es la resolución del TDC la cual es la diferencia de tiempo entre los retardos de la cadena superior e inferior (ver Ecuación 2.2 [27]). La ventaja de este tipo de arquitectura es que se puede obtener una resolución muy pequeña, dado que τ depende de la diferencia entre los tiempos de propagación de dos elementos lógicos.

$$t_x = i\tau \quad (2.1)$$

$$\tau = t_{d1} - t_{d2} \quad (2.2)$$

Un inconveniente encontrado en este tipo de TDC consiste en los errores ocasionados por la No-Linealidad Diferencial (DNL). Esta última se define como la variación del retardo de cada etapa respecto de su valor ideal [29].

En un *Application-Specific Integrated Circuit* (ASIC), es posible controlar el τ de cada etapa y añadir compensaciones contra variaciones de tensión y temperatura [30]. Por otro lado, en una FPGA, las cadenas de retardos se implementan con *carry chains* porque poseen tiempos de propagación pequeños, del orden de las decenas de los picosegundos. Sin embargo, este tiempo de propagación es sensible a las variaciones de tensión y de temperatura [31]. La variación de la tensión de alimentación puede ser controlada si la fuente fue diseñada apropiadamente, pero controlar la temperatura a la cual está expuesta la FPGA puede no ser sencillo. Dado que el TDC diseñado en este proyecto se pretende utilizar en el espacio, donde el ciclado térmico varía a lo largo de una órbita, este tipo de arquitectura no resulta adecuada para la aplicación.

Por este motivo, se propone otro tipo de arquitectura, basada en múltiples fases de una misma señal de reloj [32]. Como se puede observar en la Figura 2.2, los componentes principales de esta arquitectura son un contador grueso y un contador fino. El contador grueso cuenta ciclos de reloj y el contador fino muestrea la señal de interés múltiples veces en un mismo ciclo de reloj. Los datos proporcionados por estos contadores se guardan en un registro y luego se almacenan en una Memoria de Acceso Aleatorio (o *Random-Access Memory* en inglés) (RAM), para poder ser leídos por otros bloques externos.

Cada registro almacenado en memoria posee N puntos de muestreo de la señal de entrada durante un determinado ciclo de reloj, donde N es la cantidad de fases de reloj utilizadas. A esta mínima unidad de tiempo que puede medir el TDC se la denomina *bin* y está determinada por la Ecuación 2.3, donde T_{clk} es la frecuencia de reloj a la cual están operando los contadores.

Al conocer el tamaño del *bin* del TDC, se puede reconstruir la forma de onda de la señal de entrada, dado que cada bit del contador fino estará equiespaciado un *bin* de distancia. Este proceso se puede observar en la Figura 2.3, donde también se puede ver la incertidumbre que posee este método, la cual se origina por la cuantización misma de la señal y se define por la Ecuación 1.2. Si además se conoce en que ciclo de reloj ocurrió cada muestra del contador fino, no hace falta almacenarlas y procesarlas todas, ya que solamente generan interés las muestras del contador fino en las cuales los bits cambian. Esto permite que la memoria sea mas pequeña, disminuyendo la cantidad de recursos de FPGA necesarios para implementar el TDC, o que puedan almacenarse registros por más tiempo, lo que aumenta el rango dinámico.

$$bin_width = \frac{T_{clk}}{N} \quad (2.3)$$

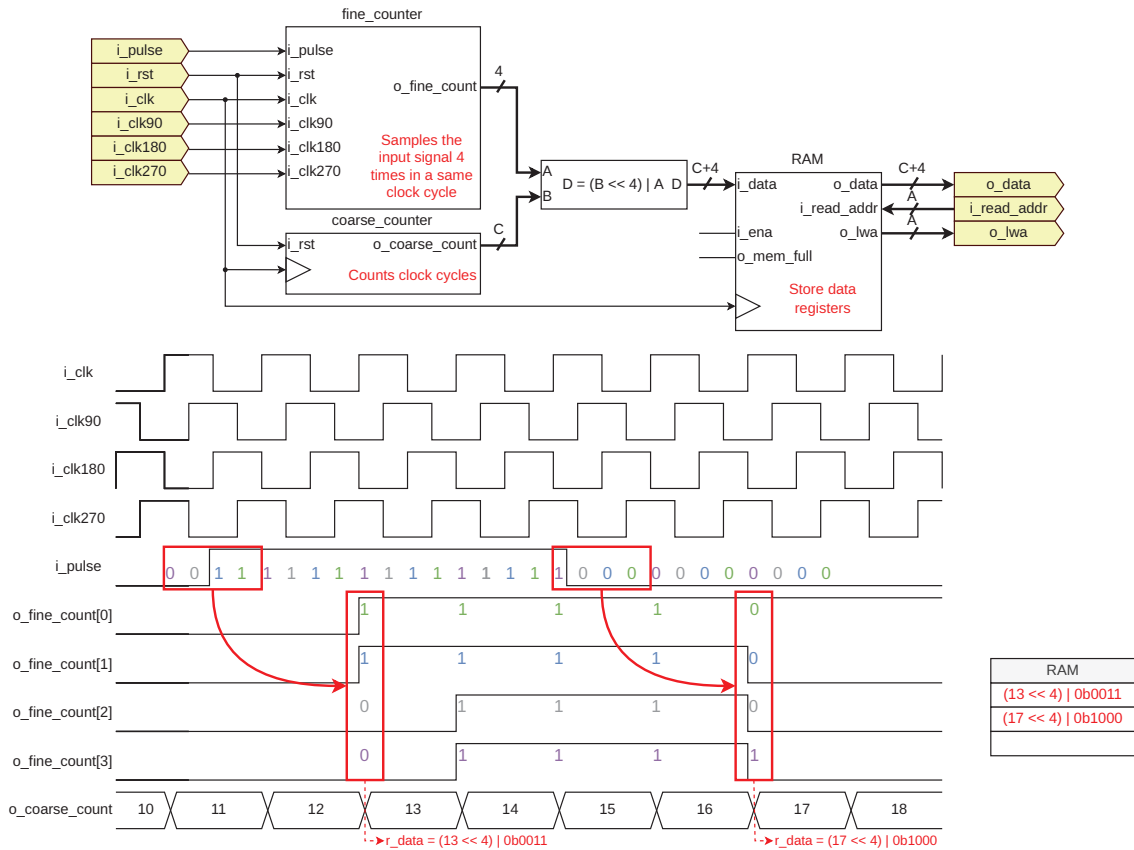


Figura 2.2: Arquitectura de TDC basada en 4 fases de una misma señal de reloj.

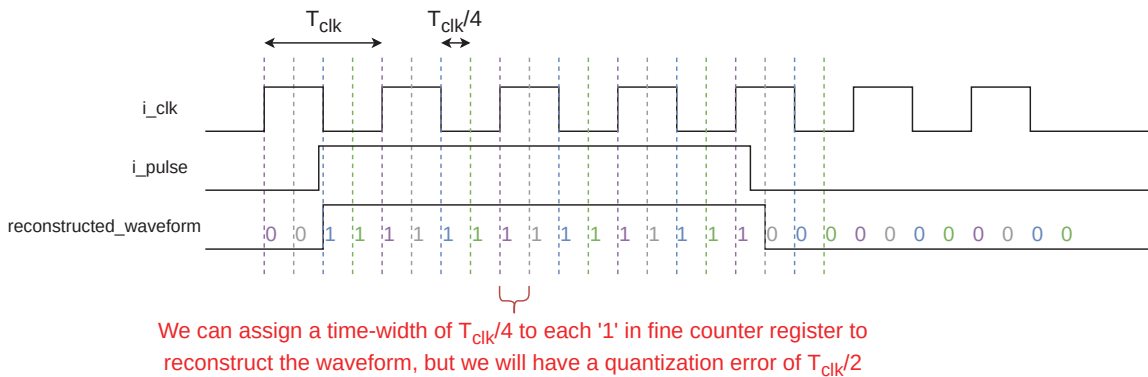


Figura 2.3: Reconstrucción de la señal de entrada a partir de los registros de datos almacenados por el TDC.

Por lo tanto, esta arquitectura permite medir señales cuyas frecuencias sean hasta N veces mayores que la frecuencia de reloj a la cual opera el contador fino. Pero, se debe tener en cuenta el error de cuantización dado por la Ecuación 1.2.

El contador fino se implementó mediante el uso de un bloque ISERDESE disponible en la FPGA, que permite obtener una señal paralela de $N = 4$ bits a partir de la señal de entrada. Como la FPGA seleccionada puede operar con una señal de reloj de 250 MHz [33], se podría lograr una resolución de 1 ns (Ecuación 2.3) con un error de cuantización de 2 ns (Ecuación 1.2). En este sentido, cabe destacar que la magnitud de los eventos temporales a medir en la aplicación se encuentra en el orden de las decenas de nanosegundos [13], por lo que la resolución alcanzada es suficiente para el

propósito del sistema.

2.2. Top level del core

En la Figura 2.4 se encuentra un diagrama del *top level* del core, configurado con 3 canales de entrada.

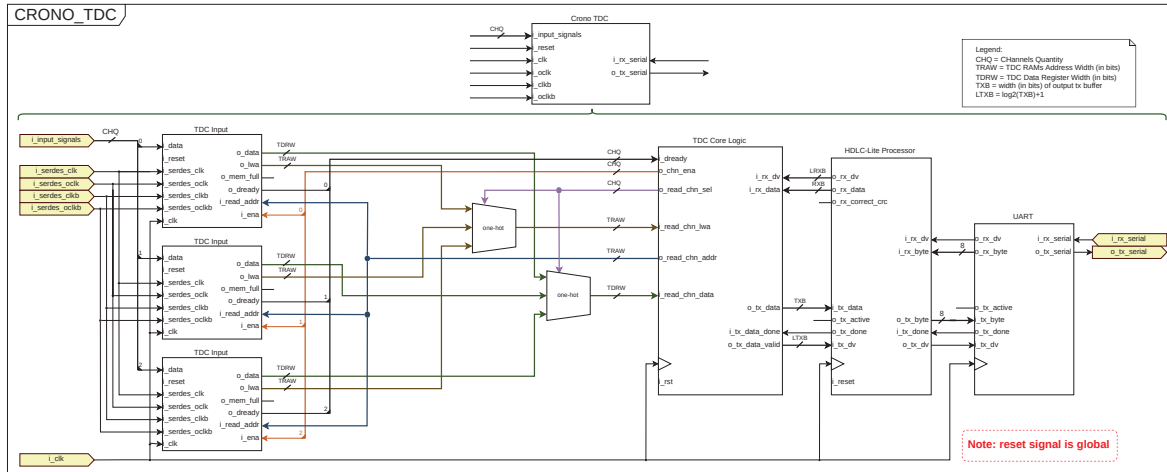


Figura 2.4: Top level del core Crono TDC.

Los bloques *TDC Input* identifican el momento en el cual ocurre un cambio de estado en las señales de entrada y guardan dicha información en una memoria RAM interna. Posteriormente, con la información de dichas memorias, el usuario puede recrear la forma de onda de las señales de entrada. En estos bloques es donde se encuentra la arquitectura de TDC basada en múltiples fases de reloj implementada con un bloque *ISERDESE*.

El core incluye una lógica de control en el bloque *TDC Core Logic* que es capaz de procesar comandos recibidos y ejecutar una acción en respuesta. En otras palabras, el TDC actúa de manera reactiva, donde el usuario debe enviar un comando para que el sistema lleve a cabo una acción. Este bloque está dividido en dos componentes: un procesador de comandos y un ejecutor de comandos.

El procesador de comandos se encarga de procesar los datos enviados por el usuario e identificar si dentro de los mismos se encuentra un comando. Si los datos incluyen un comando conocido, el procesador notifica al ejecutor de comandos para que éste lleve a cabo la acción correspondiente.

Por otro lado, el ejecutor de comandos posee distintas máquinas de estado las cuales implementan las acciones correspondientes para cada comando. Asimismo, cuenta con una lógica que gestiona cual de las máquinas de estado puede enviar datos por la única interfaz de comunicación disponible con el usuario en un mismo momento.

Con el fin de aumentar la confiabilidad de la comunicación con el usuario, se implementó un protocolo que agrupa los bytes intercambiados entre el usuario y el sistema en unidades de datos denominadas tramas. Este protocolo es una adaptación simplificada del conjunto de protocolos *High-level Data Link Control* (HDLC), los cuales son utilizados en redes de comunicación. Los comandos enviados por el usuario y las respuestas enviadas por el TDC están dentro de estas tramas. Este protocolo aumenta la robustez de la comunicación dado que determina el formato de los datos que ambos extremos deben enviar y además posee un mecanismo de detección de errores llamado Verificación por Redundancia Cíclica (o *Cyclic Redundancy Check* en inglés) (CRC).

El bloque *HDLC-lite Processor* es el encargado de procesar todos los bytes recibidos a través de la interfaz de comunicación y de construir las tramas correspondientes. Además, este bloque se encarga de verificar el CRC de las tramas recibidas y de decodificar los datos contenidos en ellas.

Por otro lado, también es responsable de tomar los datos que el TDC desea enviar al usuario, calcular su CRC y codificarlos dentro de la trama correspondiente.

La interfaz de comunicación serie entre el TDC y el usuario se lleva a cabo mediante una *Universal Asynchronous Receiver-Transmitter* (UART). Si bien se ha diseñado un bloque para dicha interfaz, se ha optado por no profundizar en su detalle debido a su simplicidad.

2.3. Operación del TDC mediante comandos

La operación del TDC se basa en la recepción de comandos enviados por el usuario, los cuales son procesados por el sistema para llevar a cabo una operación determinada. La secuencia de operación típica del dispositivo se encuentra representada en la Figura 2.5. Esta figura ilustra los pasos necesarios para utilizar el TDC mediante comandos, lo cual permite al usuario interactuar con el dispositivo de manera efectiva y controlar su funcionamiento.

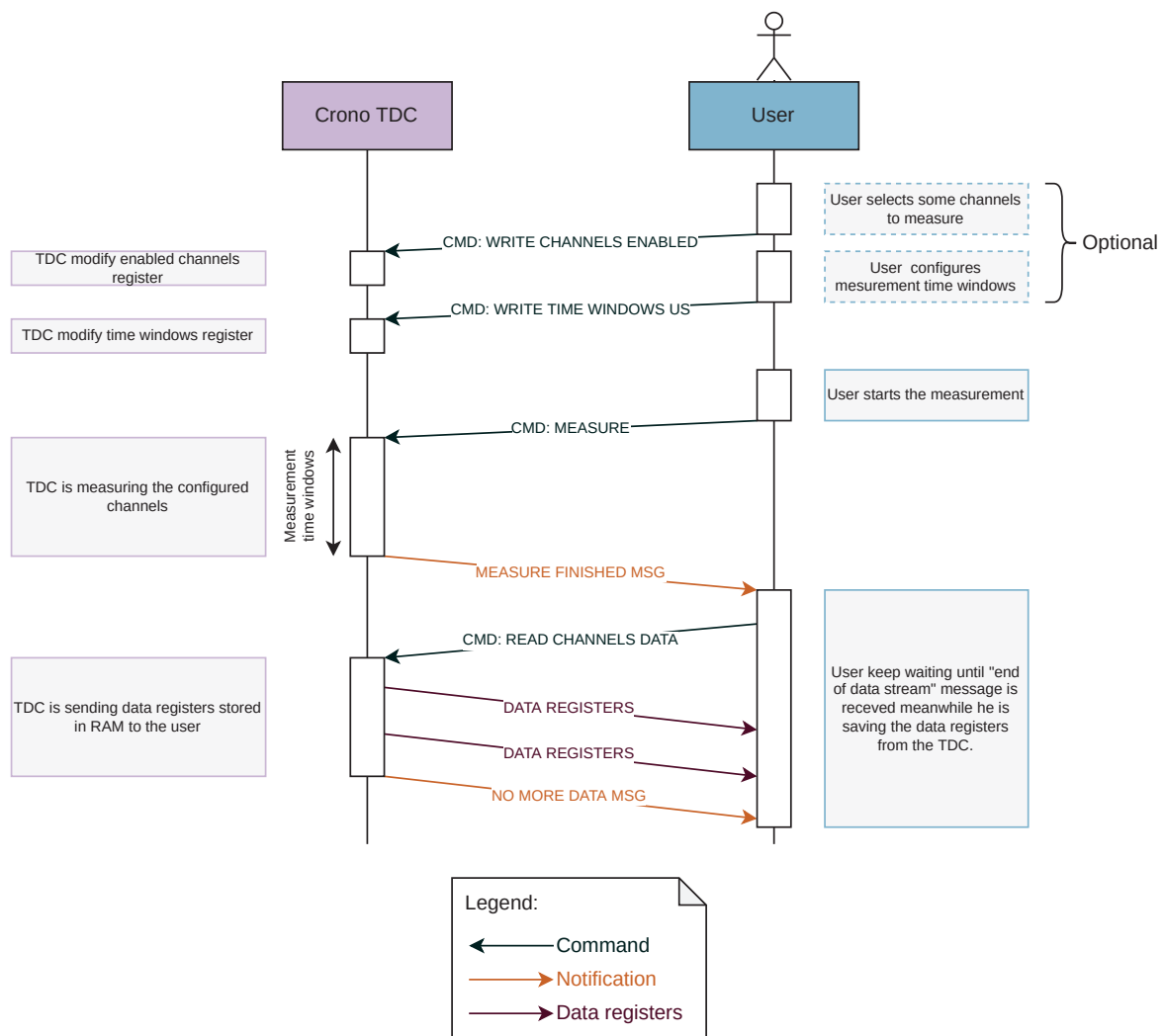


Figura 2.5: Diagrama de secuencia el cual describe una operación típica del TDC.

Antes de dar inicio a una medición, el usuario tiene la opción de configurar ciertos parámetros de la misma a través de los comandos de escritura de registros. Con esto es posible, por ejemplo, definir el tamaño de la ventana de medición y los canales con los cuales se medirá.

Para iniciar una medición, el usuario debe enviar el comando `MEASURE` y el TDC medirá

sobre los canales habilitados durante la ventana de medición configurada. Una vez que la ventana de tiempo expira, el TDC envía un mensaje al usuario indicando que los datos medidos se encuentran almacenados en su RAM.

Cuando el usuario desee acceder a los datos almacenados en el TDC, deberá enviar el comando `READ_CHANNELS_DATA` y el dispositivo procederá a enviar los registros de datos. Una vez que no haya más registros disponibles para ser enviados, el TDC enviará un mensaje de confirmación.

Es importante mencionar que cada vez que se inicia una nueva medición, los datos previamente almacenados en la RAM serán sobrescritos.

2.4. Resumen

En el presente capítulo se ha desarrollado un análisis detallado del principio de funcionamiento de la arquitectura de TDC seleccionada, la cual fue debidamente justificada frente a otra alternativa más clásica.

Asimismo, se presentó el *top level* del *core* diseñado, donde se destacaron los componentes más relevantes que lo conforman, y se describieron brevemente sus respectivas funciones.

Finalmente, se expuso una secuencia de operación característica del dispositivo, la cual brindó una visión general del comportamiento del TDC en su entorno de operación.

A continuación, se seguirá adelante con la explicación del diseño del *core*, brindando más detalles de cada uno de los bloques expuestos en el *top level* (Figura 2.4).

3. Módulos del *core*

En este capítulo se presentará una descripción detallada de los módulos que conforman el *top level* del *core* diseñado. Se iniciará con la composición de los canales de entrada, los cuales tienen la función de muestrear las señales de entrada 4 veces en un mismo ciclo de reloj y discriminar que información es necesaria almacenar en memoria. Asimismo, se abordará la lógica de procesamiento y ejecución de comandos que posee el *core*, que permite al TDC recibir ordenes y ejecutar acciones en respuesta. Finalmente, se explicará el protocolo de entramado utilizado por el TDC para comunicarse a través del puerto serie y se describirá el circuito que implementa la codificación y decodificación de dichas tramas.

3.1. Canales del TDC

Con el objetivo de permitir la implementación de varios canales de entrada en la FPGA, se desarrolló un módulo que puede ser instanciado múltiples veces en un mismo diseño. Este bloque se muestra en la Figura 3.1, y se encarga de implementar una entrada de TDC individual. De esta manera, se logró una mayor flexibilidad en el diseño, permitiendo la adición de múltiples canales de entrada en un mismo sistema, según las necesidades de la aplicación específica.

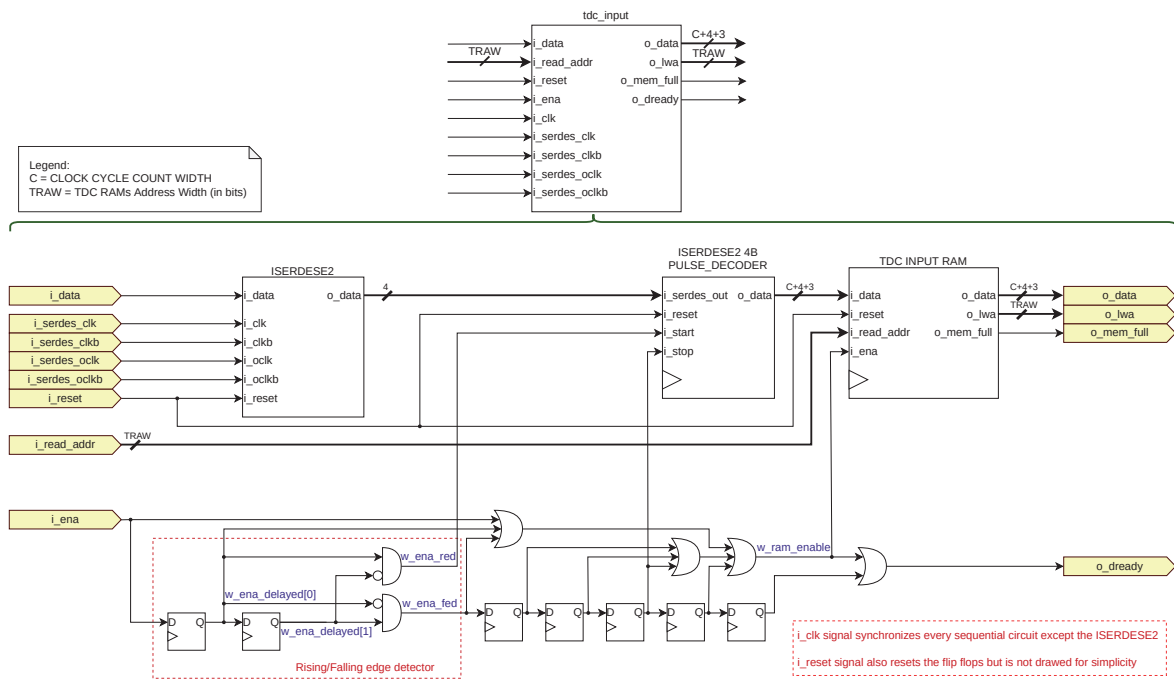


Figura 3.1: Bloque que implementa un canal de entrada del TDC.

Los componentes principales de cada entrada del TDC son: un bloque ISERDESE, un decodificador de pulsos y una RAM. El ISERDESE permite muestrear la señal de entrada 4 veces en un mismo período de reloj, produciendo una señal paralela a su salida. Luego, el decodificador de pulsos reconoce los eventos importantes en la salida del ISERDESE para luego ser almacenados en la RAM. En este capítulo se describirán en detalle cada uno de estos componentes.

3.1.1. ISERDESE

La primitiva ISERDESE de la FPGA permite deserializar una señal de entrada y puede ser configurada en diferentes modos, pero el modo de sobremuestreo es el que resulta relevante para la aplicación en cuestión.

Al configurar esta primitiva en el modo mencionado, se comporta como una matriz de *flip-flops*, estando el primer elemento de cada fila sincronizado con una señal de reloj distinta. Al alimentar a estos *flip-flops* con una misma señal de reloj desfasada 90° se logra muestrear la señal de entrada 4 veces en un mismo ciclo de reloj. Como se puede observar en la Figura 3.2, esto se debe a que cada *flip-flop* se activará en un momento distinto del período de reloj.

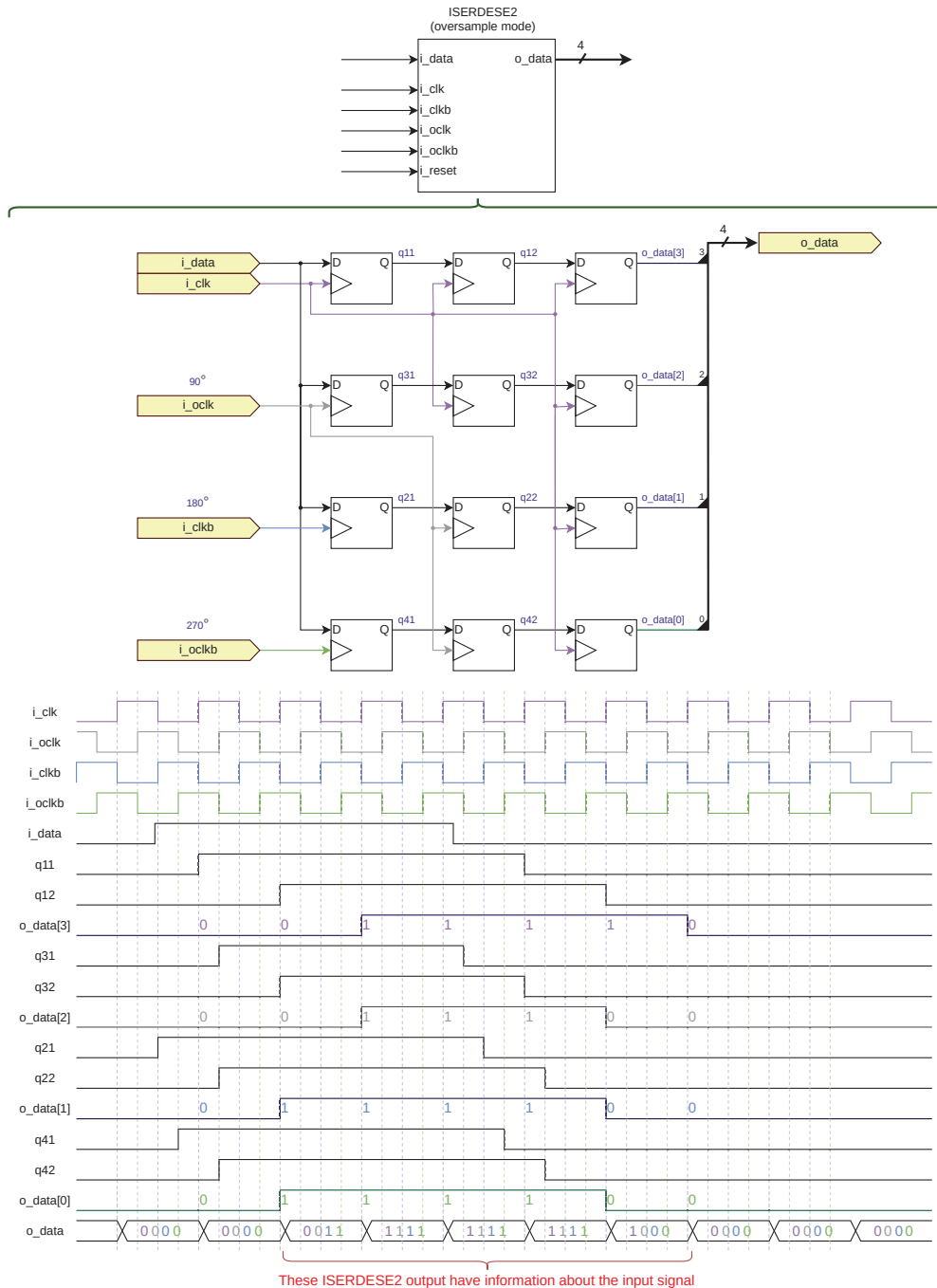


Figura 3.2: Composición y funcionamiento de la primitiva ISERDESE2 de las FPGAs *Series 7* de Xilinx. Adaptado de [34].

3.1.2. Decodificador de pulsos

Se diseñó una lógica la cual procesa la salida del ISERDESE y discierne cuales son los eventos que son necesarios almacenar en memoria y cuales no. Además, esta lógica construye los registros que finalmente son almacenados en memoria, los cuales poseen la salida del ISERDESE y la cuenta de reloj en el momento del evento. Cada uno de estos registros posee un indicador (*o tag*) el cual brinda información adicional del evento en el cual se almacenó el dato para ayudar al usuario a reconstruir la señal.

Como se puede observar en la Figura 3.3, los componentes principales de este módulo son una Máquina de Estados Finitos (o *Finite State Machine* en inglés) (FSM) que procesa la salida del ISERDESE y un contador de ciclos de reloj.

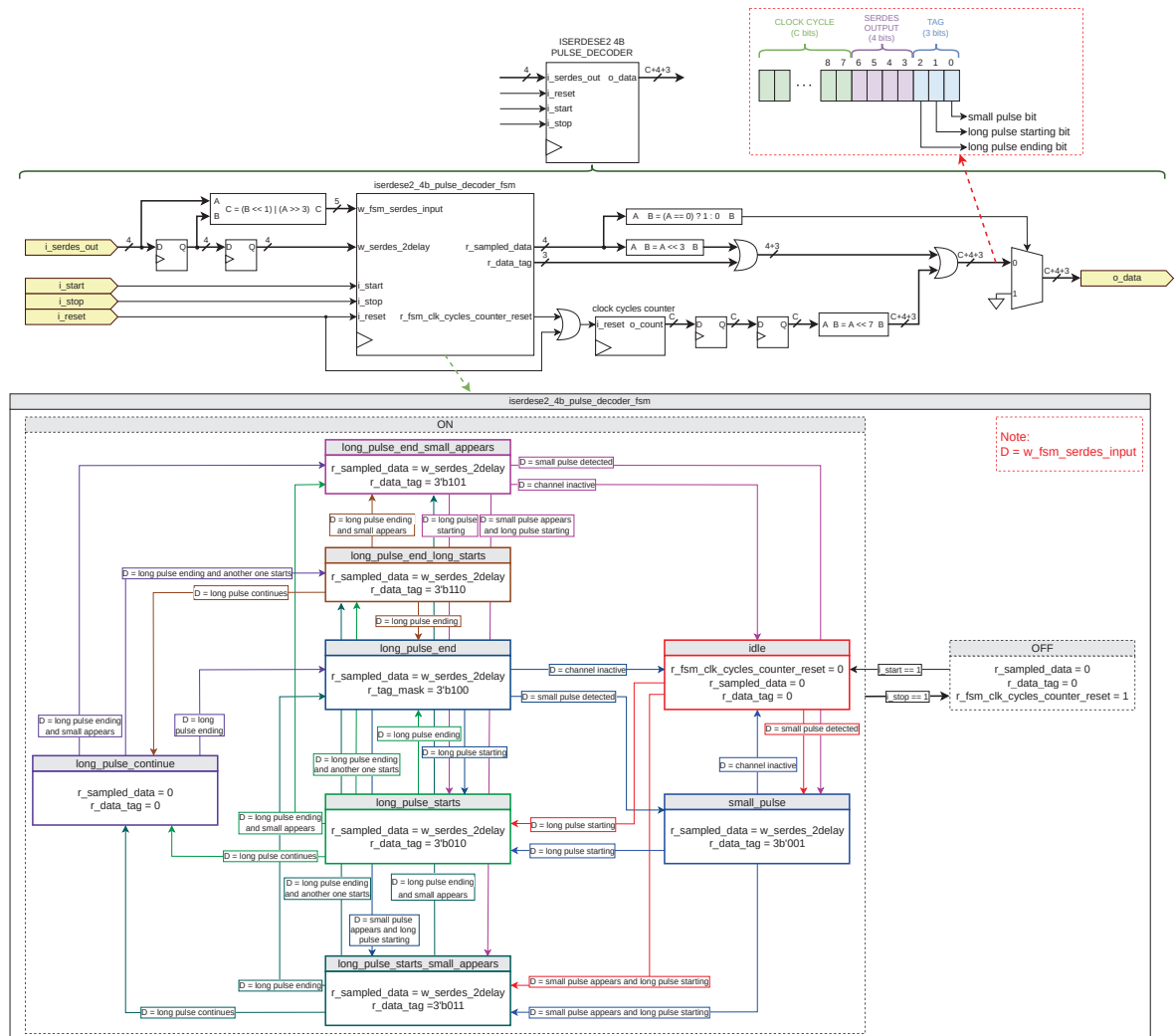


Figura 3.3: Diagrama del decodificador de pulsos.

Los pulsos de la señal de entrada fueron clasificados en pulsos cortos y pulsos largos. Los primeros poseen una duración menor a un ciclo de reloj y los últimos abarcan más de un ciclo de reloj. Cabe destacar que la duración de un pulso largo no tiene porque ser mayor a un ciclo de reloj, basta con que el pulso sea muestreado en más de un ciclo de reloj para que califique como tal. Con esta clasificación y teniendo en cuenta que el ISERDESE puede tomar 4 muestras por ciclo de reloj, se definieron 6 *tags* posibles para cada registro de datos, los cuales se pueden observar en la Figura 3.4. Esta información es almacenada en memoria junto con la cuenta de reloj y la salida del ISERDESE para que el usuario pueda reconstruir la señal sin lugar a ambigüedades.

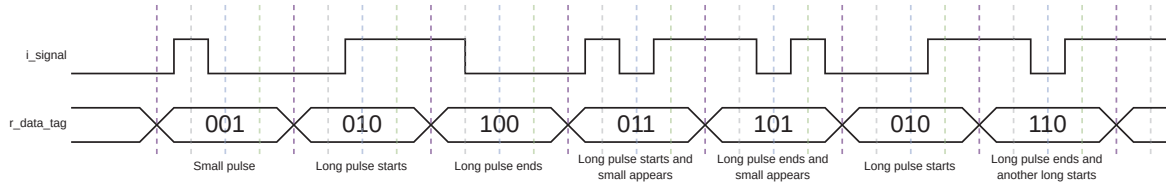


Figura 3.4: *Tags* posibles que pueden ser anexados a cada registro de datos almacenado en memoria por el decodificador de pulsos.

La FSM utiliza la salida del ISERDESE retrasada un ciclo de reloj y el bit más significativo de la salida actual del ISERDESE para decidir el siguiente estado. Como se puede observar en la Figura 3.5, de esta manera la FSM muestrea la señal de entrada con una ventana la cual se traslada 4 *bins* en cada ciclo de reloj. La salida del ISERDESE retrasada posee las muestras $k_i k_{i+1} k_{i+2} k_{i+3}$ de la señal de entrada y la salida actual posee la muestra k_{i+4} .

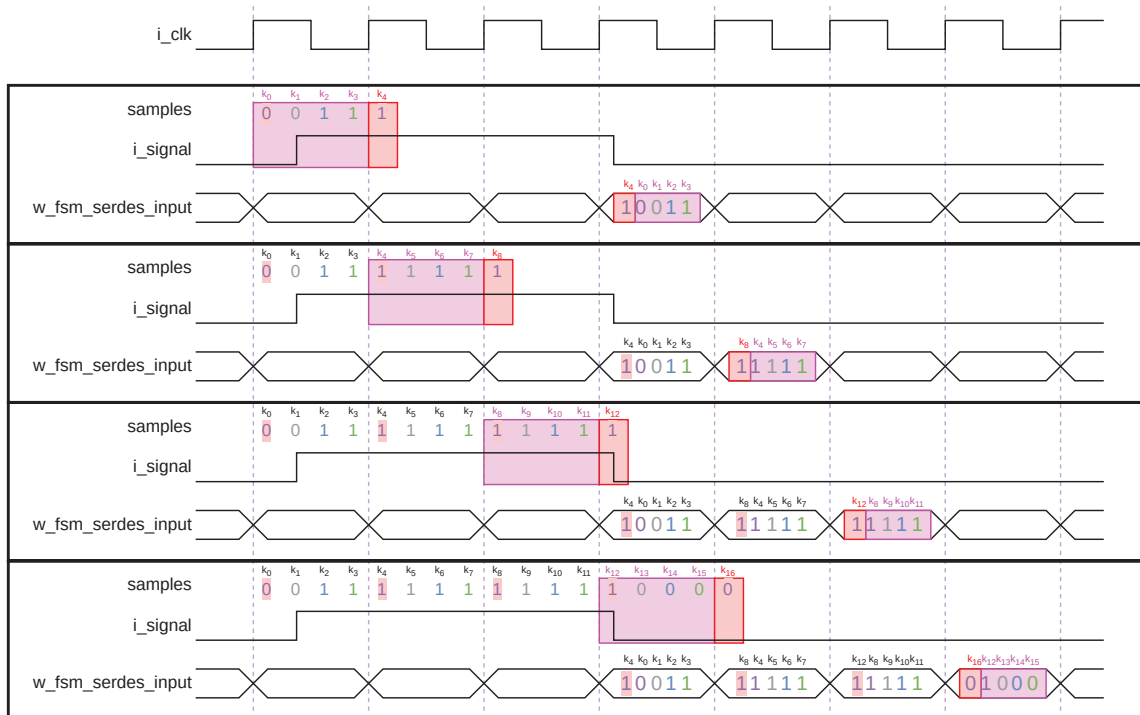


Figura 3.5: Muestras del ISERDESE tomadas por la máquina de estados del decodificador de pulsos para decidir el próximo estado. En rosa, la salida del ISERDESE retrasada un ciclo de reloj. En rojo, el bit mas significativo de la salida actual del ISERDESE.

La muestra k_{i+4} se utiliza para determinar casos en los cuales la información de la muestras $k_i k_{i+1} k_{i+2} k_{i+3}$ no brinden la información suficiente. Por ejemplo, en la Figura 3.6 se encuentra el caso en el cual la muestra k_{i+4} permite a la FSM determinar que un pulso largo está iniciando.

Solamente los registros que proveen información acerca de un cambio en la señal de entrada son almacenados por la FSM en memoria. Sin este mecanismo, la RAM debería ser lo suficientemente grande como para almacenar todas las muestras tomadas por el ISERDESE. Esto puede ser visualizado en la simulación de la Figura 3.7, donde el puerto de salida `o_data` posee un valor distinto de 0 en los casos en los cuales las muestras $k_i k_{i+1} k_{i+2} k_{i+3} k_{i+4}$ muestran un cambio en la señal de entrada. En estos casos, el registro `r_data_tag` también toma un valor no nulo, el cual identifica el tipo de evento en cuestión.

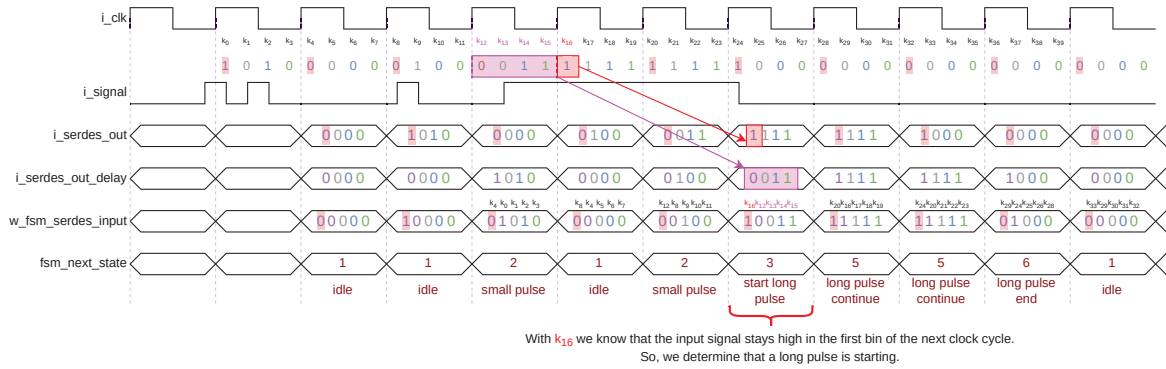


Figura 3.6: Ejemplo de detección de un pulso el cual persiste en un estado lógico alto al avanzar la señal de reloj de un ciclo a otro. Con la muestra k_{i+4} se puede determinar que el pulso es “largo”.

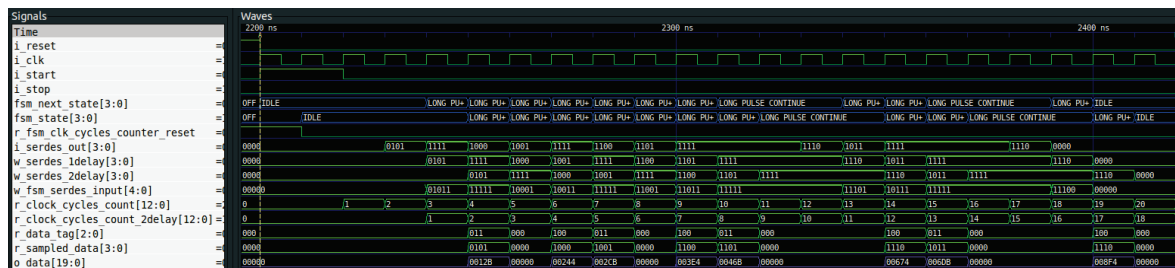


Figura 3.7: Simulación del decodificador de pulsos procesando una serie de salidas ISERDESE.

3.1.3. RAM

En la Figura 3.8 se puede visualizar la memoria RAM ubicada en cada entrada del TDC. La RAM utilizada posee solamente un puerto de lectura/escritura, por lo cual, se diseñó un sencillo controlador para arbitrar en que momento se lee y se escribe sobre la misma. El componente principal de este controlador es un detector de flancos ascendentes y un contador de direcciones de memoria.

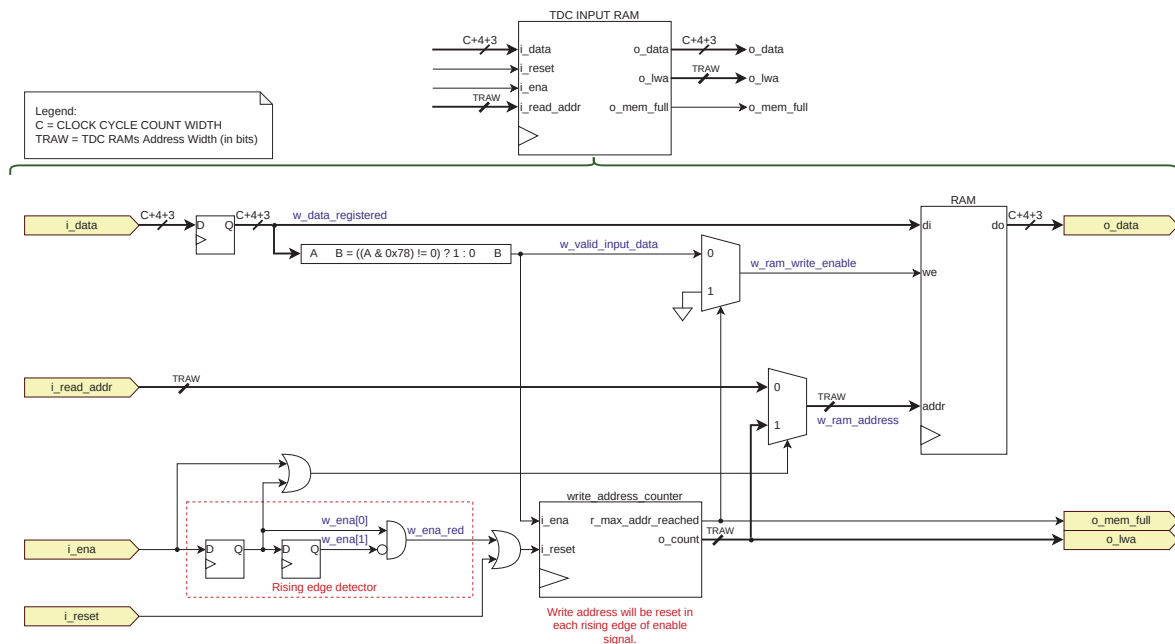


Figura 3.8: RAM implementada en cada canal de TDC.

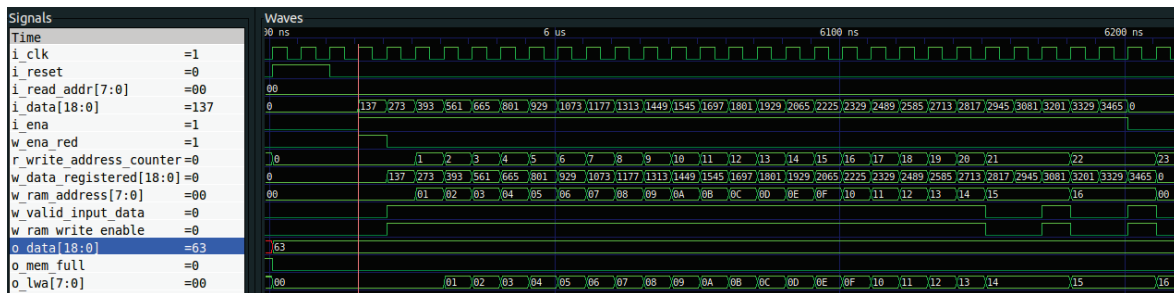
El módulo se activa al detectar un estado lógico alto en el puerto `i_ena` y se desactiva al detectar un estado lógico bajo en el mismo.

Cuando se activa el módulo la memoria se posiciona en un modo de solo escritura. Al suceder esto, el contador de direcciones de memoria de escritura se reinicia y cada vez que se introduce un dato válido a través del puerto `i_data`, éste se almacena en la siguiente posición de memoria disponible. Se considera dato válido el que posee un valor distinto de cero en los bits [6:3], que es donde se encuentra el campo con la muestra del ISERDESE.

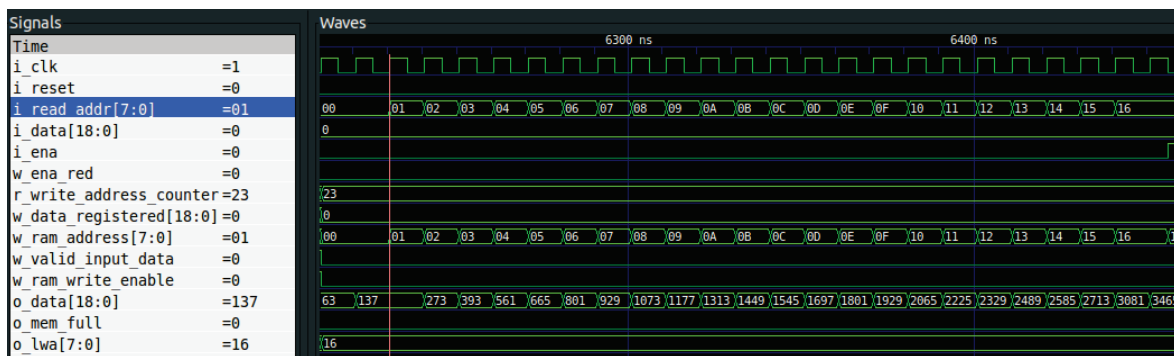
En todo momento, el módulo indica a través del puerto `o_lwa` la última posición de memoria en la que se escribió, lo que permite que una lógica externa sepa hasta qué posición de memoria hay datos válidos. La Figura 3.9a muestra el proceso de escritura de la memoria RAM de entrada.

Cuando el módulo está deshabilitado la memoria se encuentra en un modo de solo lectura. En esta condición, solo se pueden leer los datos almacenados en la memoria utilizando el puerto `i_read_addr` para indicar la posición deseada. En la simulación de la Figura 3.9b se puede observar que los datos previamente escritos en la simulación de escritura aparecen en el puerto de salida `o_data` a medida que se indica una nueva posición de memoria.

Este funcionamiento permite que cada vez que se realice una medición con el TDC, se habilite este módulo para almacenar los registros de datos en la memoria. Al finalizar la medición se deshabilita el módulo para que los datos puedan ser leídos desde la memoria utilizando los puertos `i_read_addr` y `o_data`.



(a) Simulación de una escritura en la RAM de cada canal de TDC.



(b) Simulación de una lectura en la RAM de cada canal de TDC.

Figura 3.9: Simulación de una escritura y una lectura en la RAM ubicada en cada canal del TDC.

En las FPGAs es posible llevar a cabo la implementación de RAM de forma distribuida, utilizando LUTs, o mediante el uso de recursos específicos de la propia FPGA, denominados *Block RAM* (BRAM), los cuales están diseñados para tal fin. Es importante tener en cuenta que el acceso a la RAM distribuida resulta más rápido en el caso de memorias de tamaño pequeño, no obstante, debe considerarse que la implementación de dicha memoria requiere del uso de recursos lógicos de la FPGA que podrían ser destinados a otras partes del diseño. Para mantener al mínimo el uso de recursos lógicos de cada canal, se priorizó utilizar BRAM para implementar sus memorias.

Cada dirección de memoria almacena un registro de datos que está compuesto de C bits para la cuenta de reloj (dato generado por el contador grueso), 4 bits para la muestra del ISERDESE y 3 bits para el *tag*. A medida que la cuenta de reloj se represente con más bits, se podrá medir durante mayor tiempo antes de que ocurra un *overflow*, otorgando un mayor rango dinámico al TDC. Por lo tanto, existe una relación de compromiso entre el rango dinámico del TDC y el tamaño de la RAM de cada canal.

La FPGA seleccionada (XC7A35T) posee 50 BRAM internas de 36 *kb* [35], obteniendo una capacidad de almacenamiento total de 1,8 *Mb* [36]. En la Figura 3.10 se puede observar la relación entre el rango dinámico, la cantidad de canales que pueden ser implementados y el tamaño de la cuenta del contador grueso. Para mayor flexibilidad, el tamaño de la cuenta del contador grueso y el tamaño de la RAM de cada canal se diseñó para que fuera configurable.

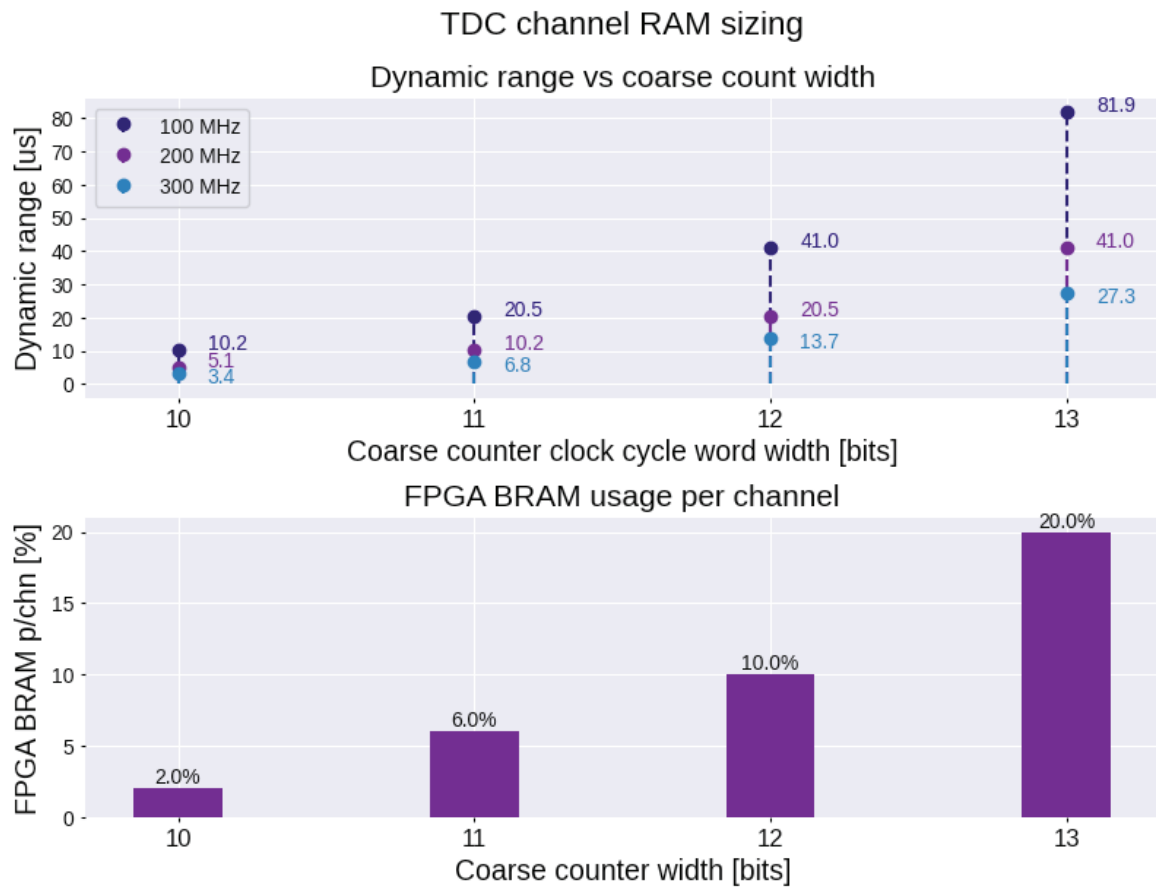


Figura 3.10: Relación entre el rango dinámico del TDC, el largo de la cuenta del contador grueso y la cantidad de canales que pueden implementarse en la FPGA XC7A35T.

3.2. Lógica de procesamiento y control

El módulo *TDC Core Logic* se encarga de procesar los datos enviados por el usuario y detectar si conforman un comando. Si un comando es recibido, entonces el módulo ejecuta las acciones correspondientes en respuesta. Como puede observarse en la Figura 3.11, el módulo se dividió en dos: un procesador de comandos y un ejecutor de comandos. En los siguientes apartados se describirán en detalle cada uno de estos bloques.

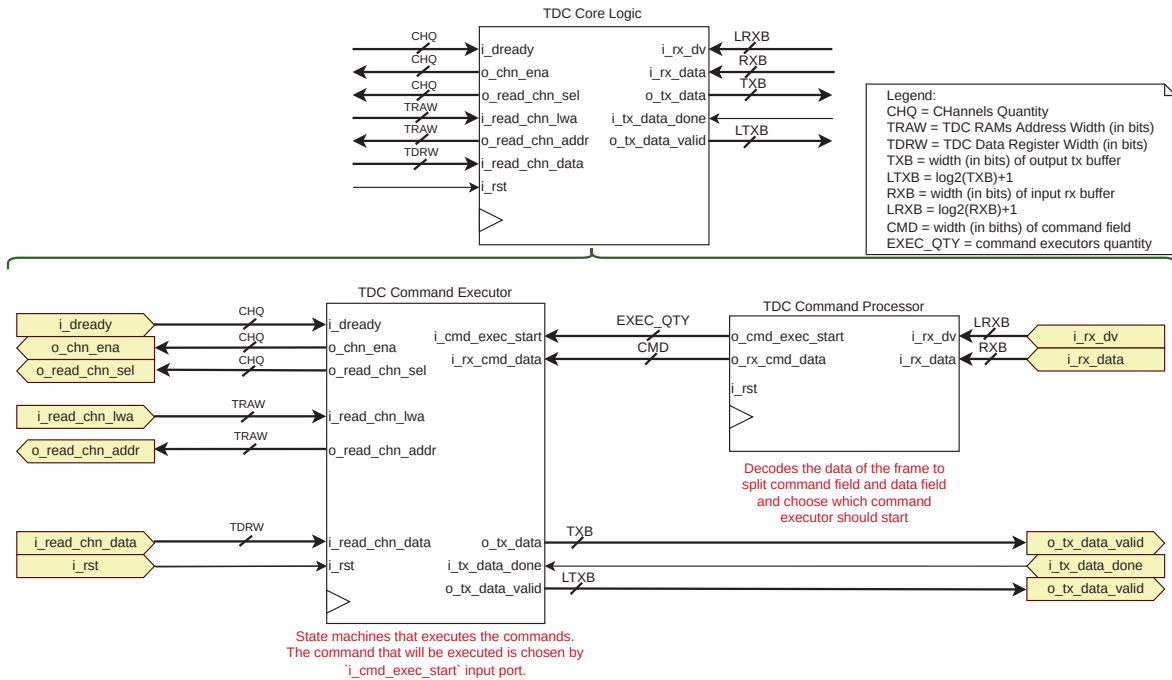


Figura 3.11: *TDC Core Logic*: Lógica de procesamiento y ejecución de comandos.

3.2.1. Procesador de comandos

Los comandos son enviados por el usuario dentro de una trama de un protocolo conocido como HDLC-lite, el cual se detalla en la Sección 3.3. Como se observa en la Figura 3.12, los comandos poseen un campo de datos, el cual puede ser utilizado por el usuario para enviar alguna configuración al TDC, y un campo de identificación único.

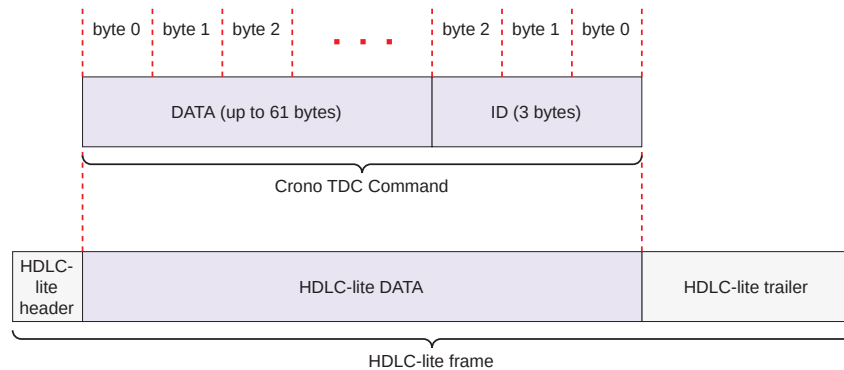


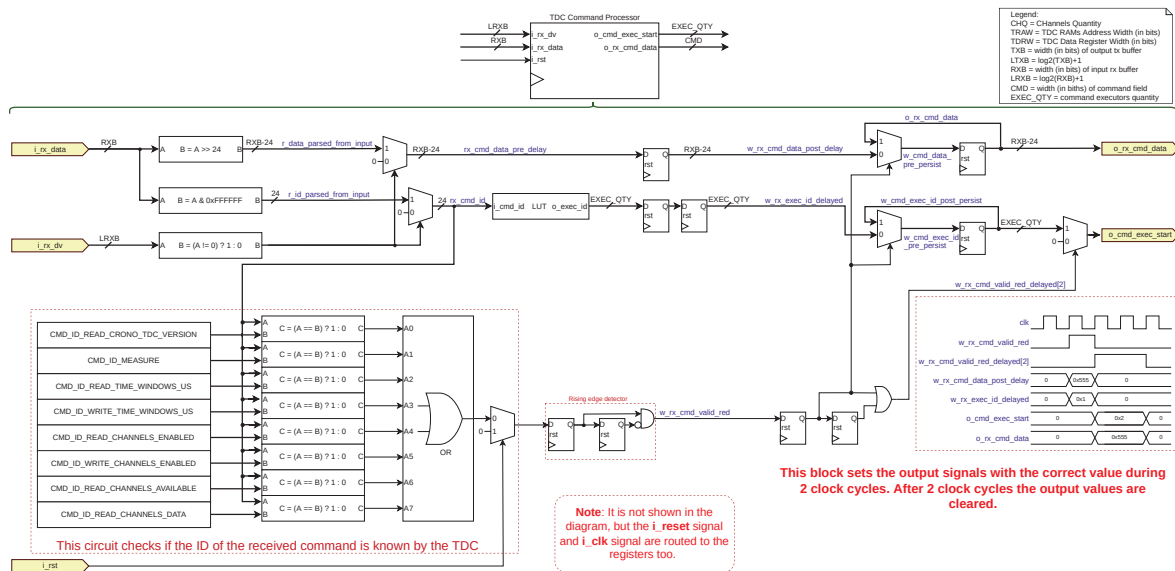
Figura 3.12: Formato de la trama de comandos que debe enviar el usuario.

El objetivo del procesador, cuyo circuito se puede observar en la Figura 3.13a, es verificar si el TDC recibió un comando, y si este es el caso, entonces debe informar al “Ejecutor de comandos” del suceso, indicándole cual fue el comando recibido.

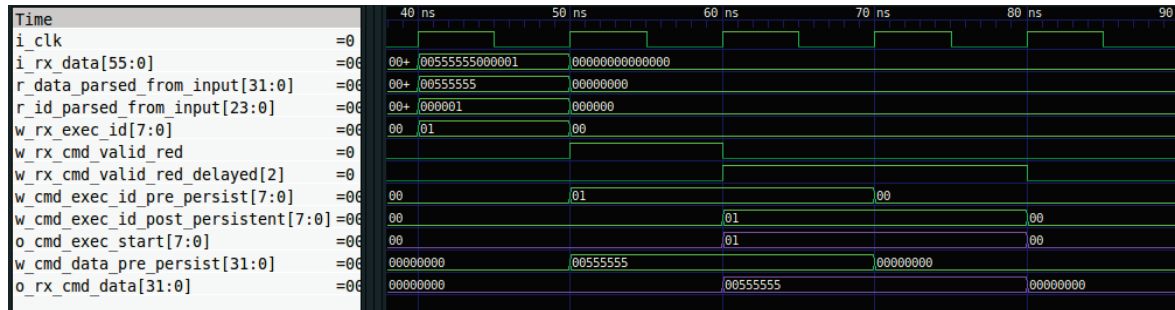
Al recibir una trama de comando, el circuito extrae con un *barrel shifter* el campo de datos (`r_data_parsed_from_input`) y con una serie de compuertas AND el identificador (`r_id_parsed_from_input`). Luego, el identificador es introducido dentro de una *Look Up Table* (LUT) que aparea un identificador de comando con un identificador de ejecutor, el cual determina qué ejecutor debe activarse para determinado comando. Al mismo instante, el identificador del comando es comparado con todos los identificadores de comandos conocidos por el TDC. Si el identificador del comando recibido es igual

a, al menos, uno de los identificadores de comandos conocidos entonces la señal `w_rx_cmd_valid_red` toma un estado lógico alto. Esta misma señal es prolongada durante los dos ciclos de reloj siguientes, generando la señal `w_rx_cmd_valid_red_delayed[2]`, la cual habilita el paso del identificador hacia el puerto de salida (`o_cmd_exec_start`).

El funcionamiento de este circuito se muestra en la Figura 3.13b, donde se puede visualizar como el *core* extrae el identificador `0x1` y el dato `0x555555` de la trama de comandos y luego los procesa. El mismo detecta que el comando es válido y en el puerto de salida `o_cmd_exec_start` se expone el identificador del ejecutor correspondiente.



(a) Diagrama del módulo *TDC Command Processor*.



(b) Simulación del módulo *TDC Command Processor* decodificando el comando con identificador `0x1` y cuyos datos son `0x555555`.

Figura 3.13: Composición del procesador de comandos y una simulación donde se observa el procesamiento de un comando.

3.2.2. Ejecutor de comandos

Este módulo, que puede observarse en la Figura 3.14, es responsable de ejecutar las acciones correspondientes cuando se recibe un comando. Cada comando posee un ejecutor asociado, por ejemplo, si el usuario envía un comando para modificar un registro de configuración, entonces existe un ejecutor cuyo objetivo es escribir dicho registro. Dado que hay varios comandos, y por lo tanto varios ejecutores, este módulo se encarga de administrar qué ejecutor debe activarse en cada instante de tiempo.

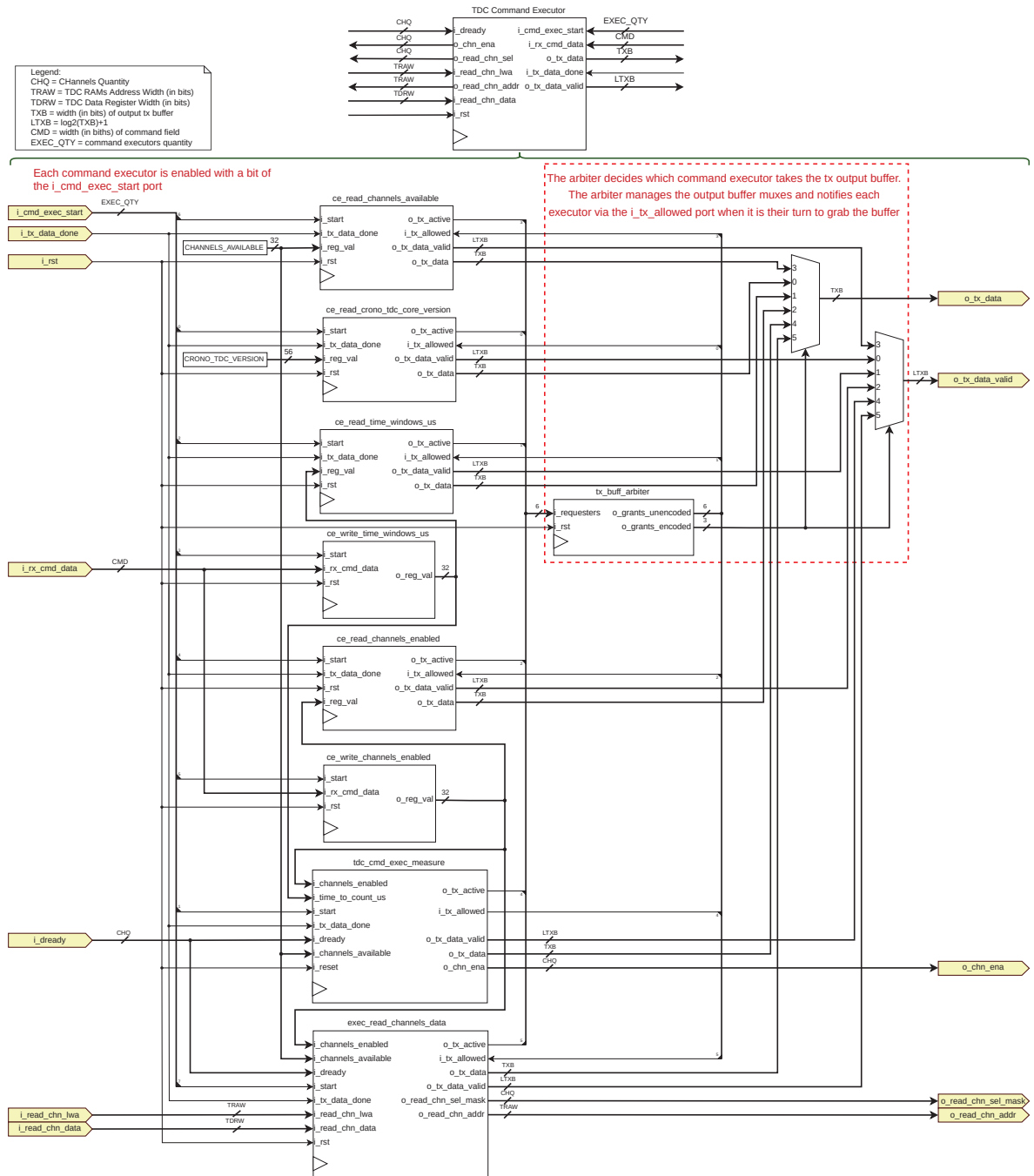


Figura 3.14: Diagrama del módulo *TDC Command Executor*.

Cada bit del puerto `i_cmd_exec_start` tiene como objetivo iniciar la rutina de un ejecutor distinto. Si dos comandos son recibidos en un mismo instante de tiempo, ambos ejecutores van a iniciar su rutina, pero como solamente hay un canal de datos de salida (`o_tx_data`), se agregó un circuito, compuesto por un árbitro y multiplexores, que gestiona el uso de este *buffer*.

Cada ejecutor debe notificar al árbitro que desea utilizar el *buffer* de salida forzando un estado lógico alto en el bit correspondiente de la señal `i_requesters`. Cuando esto sucede, el árbitro manipula los multiplexores para asignar el *buffer* de salida al ejecutor y lo notifica de este suceso con la señal `o_grants_unencoded`. Mientras el *buffer* esté tomado por un ejecutor el resto deberá esperar hasta que se libere. En el caso de que haya más de un ejecutor esperando usar el *buffer*, el árbitro decide cuál de ellos es el siguiente que lo tomará respetando un esquema de prioridades.

En la simulación de la Figura 3.15 se puede observar al ejecutor de comandos en funcionamiento. En este caso fue enviado el comando para leer el tamaño de la ventana de tiempo de medición, donde el valor de este registro es $0xA$.

La secuencia inicia al activarse el ejecutor correspondiente (bit 2 del puerto `i_cmd_exec_start`). El ejecutor expone en su puerto de salida de datos (`w_read_time_windows_us_tx_buff`) el valor del registro y utiliza la señal `w_read_time_windows_us_tx_active` para requerir al árbitro el control del *buffer* de salida global.

En el siguiente ciclo de reloj, el árbitro le entrega el *buffer* al ejecutor (ver señales `tx_buff_grants_unencoded` y `tx_buff_grants_encoded`) y manipula los multiplexores de salida para que los datos del ejecutor se presenten en el *buffer* de salida global (`o_tx_data`).

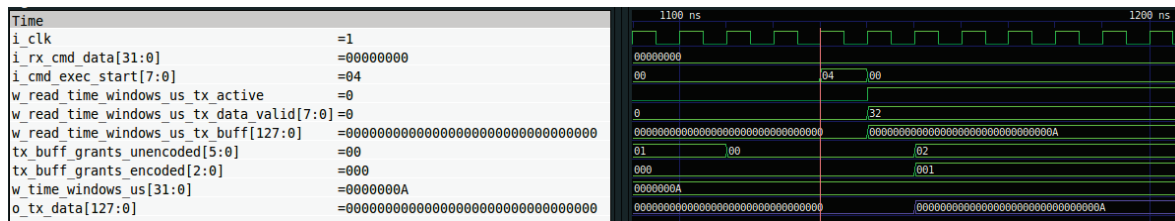


Figura 3.15: Simulación de la activación del ejecutor del comando de lectura de un registro de configuración.

3.2.2.1. Lectura de registros de configuración

El ejecutor `ce_read_register`, el cual puede observarse en la Figura 3.16, lee el valor de uno de sus puertos de entrada `i_reg_val` y lo envía a través de uno de sus puertos de salida `o_tx_data`. El diseño de este módulo es flexible, en el puerto `i_reg_val` puede ser colocado el valor de cualquier registro. Este módulo se instancia 4 veces para leer registros distintos que poseen:

- los canales activos actualmente (`ce_read_channels_enabled`).
- la cantidad de canales disponibles que existen para habilitar (`ce_read_channels_available`).
- la version de *bitstream* del *core* del TDC (`ce_read_crono_tdc_version`).
- el valor de la ventana de tiempo que se utilizará en la proxima medición (`ce_read_time_windows_us`).

En la Figura 3.17 se puede visualizar una simulación de este módulo en la cual el valor del registro es $0x200F00F1$. Un ciclo de reloj después de que la señal de habilitación llegue al puerto `i_start` el ejecutor presenta en su puerto de salida `o_tx_data` el valor del registro y la señal de notificación `o_tx_active` toma un valor lógico alto. Luego, se le concede acceso al *buffer* de salida global (ver señal `i_tx_allowed`) y ciclos de reloj después se le notifica que los datos fueron enviados (ver señal `i_tx_data_done`).

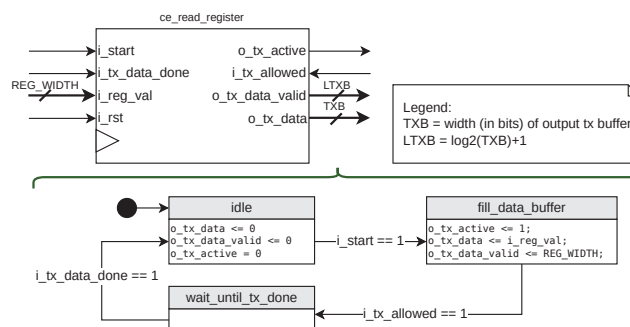


Figura 3.16: Máquina de estados del módulo *Command Executor Read Register*.

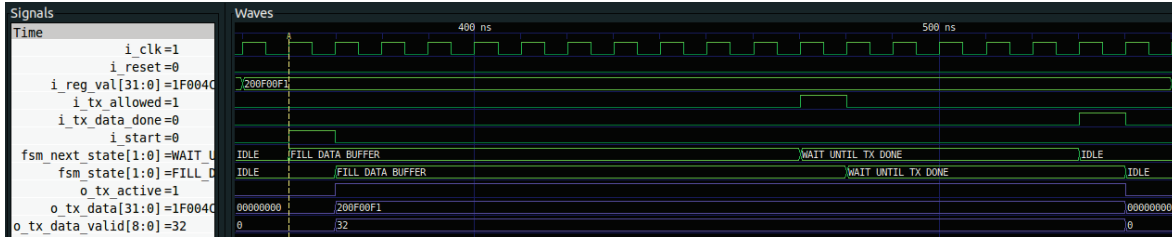


Figura 3.17: Simulación de la activación del ejecutor utilizado para los comandos de lectura de registros de configuración.

3.2.2.2. Escritura de registros de configuración

El ejecutor `ce_write_register`, cuyo diagrama puede observarse en la Figura 3.18, toma un valor de su puerto de entrada `i_rx_cmd_data` y lo escribe en el registro de salida `o_reg_val` cuando detecta un flanco ascendente en su puerto `i_start`. El registro de salida `o_reg_val` posee un valor por defecto, el cual puede ser configurable a través de un parámetro.

Este módulo fue instanciado múltiples veces para implementar ejecutores de varios comandos utilizados para escribir registros de configuración. Entre estos registros se encuentran los que poseen:

- los canales activos. Modificando este registro se pueden activar o desactivar canales.
- el valor de la ventana de tiempo que se utilizará en la proxima medición. Escribiendo en este registro se puede modificar el intervalo de tiempo que durará una medición del TDC.

En la Figura 3.19 se puede visualizar una simulación de este módulo en la cual el valor del registro `o_reg_val` se va actualizando con el valor del puerto de entrada `i_rx_cmd_data` cada vez que se activa la señal de inicio `i_start`.

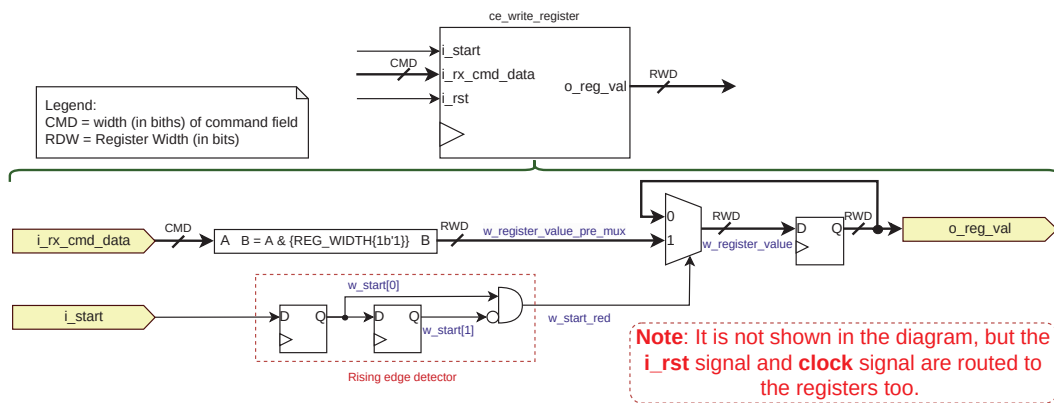


Figura 3.18: Diagrama del módulo *TDC Command Executor Write Register*.

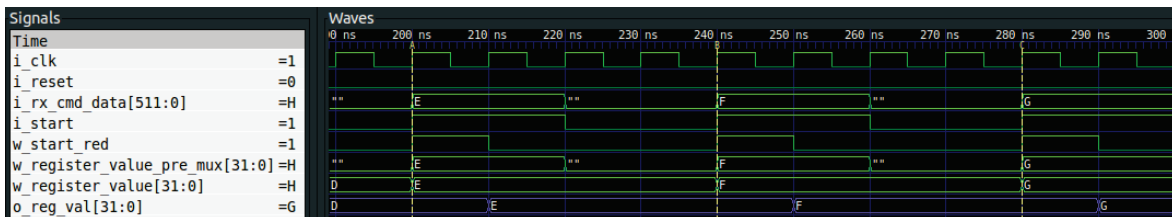


Figura 3.19: Simulación de la activación del ejecutor utilizado para los comandos de escritura de registros de configuración.

3.2.2.3. Medición de los canales de entrada

El ejecutor `ce_measure` es quien se encarga de lanzar una medición con los canales actualmente habilitados durante la ventana de tiempo configurada. Como se puede observar en la Figura 3.20, el módulo está compuesto por una FSM, un contador de ciclos de reloj, un temporizador de microsegundos y una tabla de conversión a ASCII.

La máquina de estados que se encuentra en la Figura 3.21 coordina todas las acciones del ejecutor y el temporizador es utilizado para determinar cuando se debe finalizar la medición. Por otro lado, la tabla de conversión a formato ASCII es utilizada para convertir a una cadena de caracteres la máscara de canales con la cual se midió, ya que al final de la medición se la envía al usuario en un mensaje.

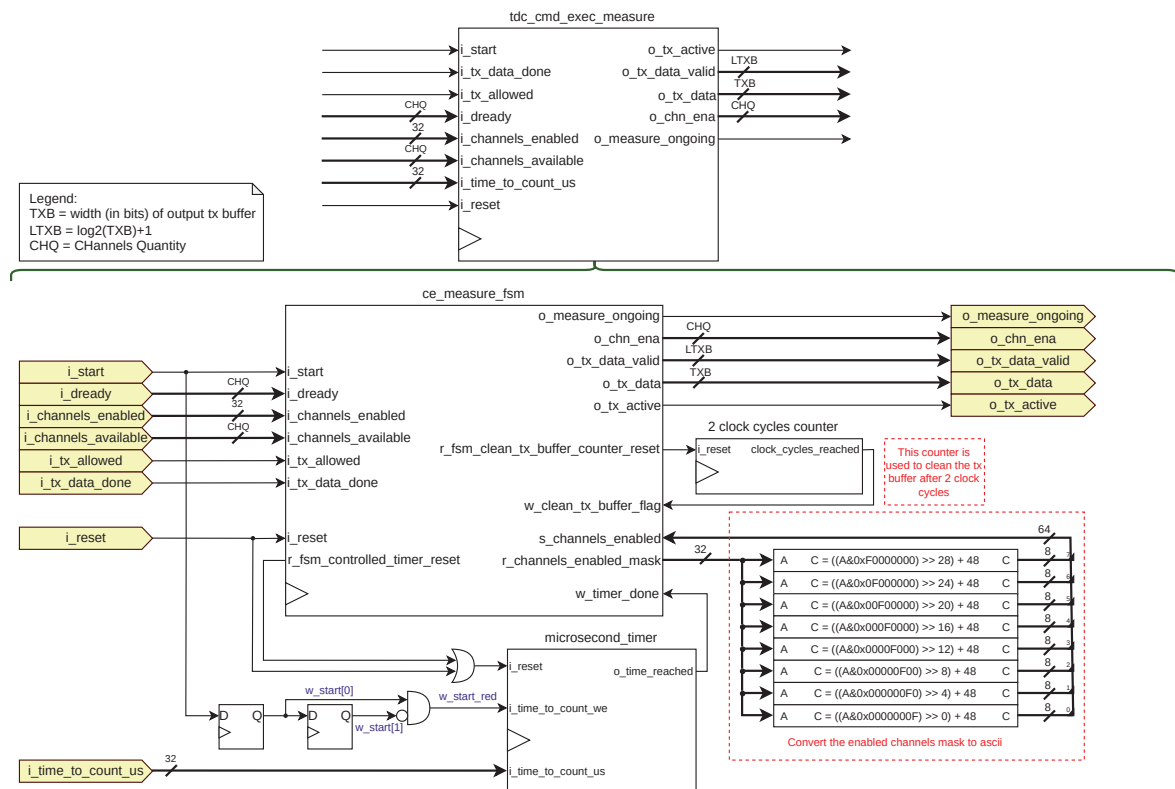


Figura 3.20: Diagrama del módulo *TDC Command Executor Measure*.

En las Figuras 3.22 y 3.23 se encuentra una simulación donde se puede apreciar una rutina de este ejecutor. En dicha simulación, se encuentran disponibles los canales 2, 1 y 0 (`i_channels_available = 0b111`). Sin embargo, se utilizaron solamente los canales 2 y 0 (`i_channels_enabled = 0b101`), y la ventana de tiempo es $10 \mu s$ (`i_time_to_count_us = 10`).

En la Figura 3.22 se puede observar que la FSM inicia en estado *idle* hasta que detecta que la señal `i_start` toma un estado lógico alto. Al avanzar al estado *set.time.windows*, se construye la máscara de bits con los canales habilitados (`r_channels_enabled_mask`) y su versión en formato ASCII (`s_channels_enabled`). Luego, cuando avanza al estado *enable.channels* indica a los canales que empiecen a medir a través del puerto `o_chn_ena` y reinicia el temporizador. La FSM permanece en el estado *wait.until.time.over* hasta que el temporizador finalice de contar.

En la Figura 3.23 se puede observar que cuando el temporizador finaliza (`w_timer_done`), la FSM evoluciona al estado *disable.channels*, en el cual indica a los canales que finalicen sus mediciones forzando un cero en el puerto `o_chn_ena`. Luego espera hasta que el puerto `i_dready` tome el mismo valor que la máscara de canales habilitados, lo cual indica que los datos de los canales ya se encuentran almacenados en sus memorias. Al suceder esto, avanza al estado *store.data.in.tx.buff*, en el cual

almacena en el *buffer* de salida `o_tx_data` el mensaje “ena00000005”. La FSM reposa en este estado durante 3 ciclos de reloj y después continúa al estado *clean_tx_buffer*, en el cual permanece hasta que el *buffer* de salida es transmitido, lo cual reconoce cuando el puerto de entrada `i_tx_data_done` toma un valor lógico alto. Al detectar esto, la máquina de estados vuelve al estado *idle*.

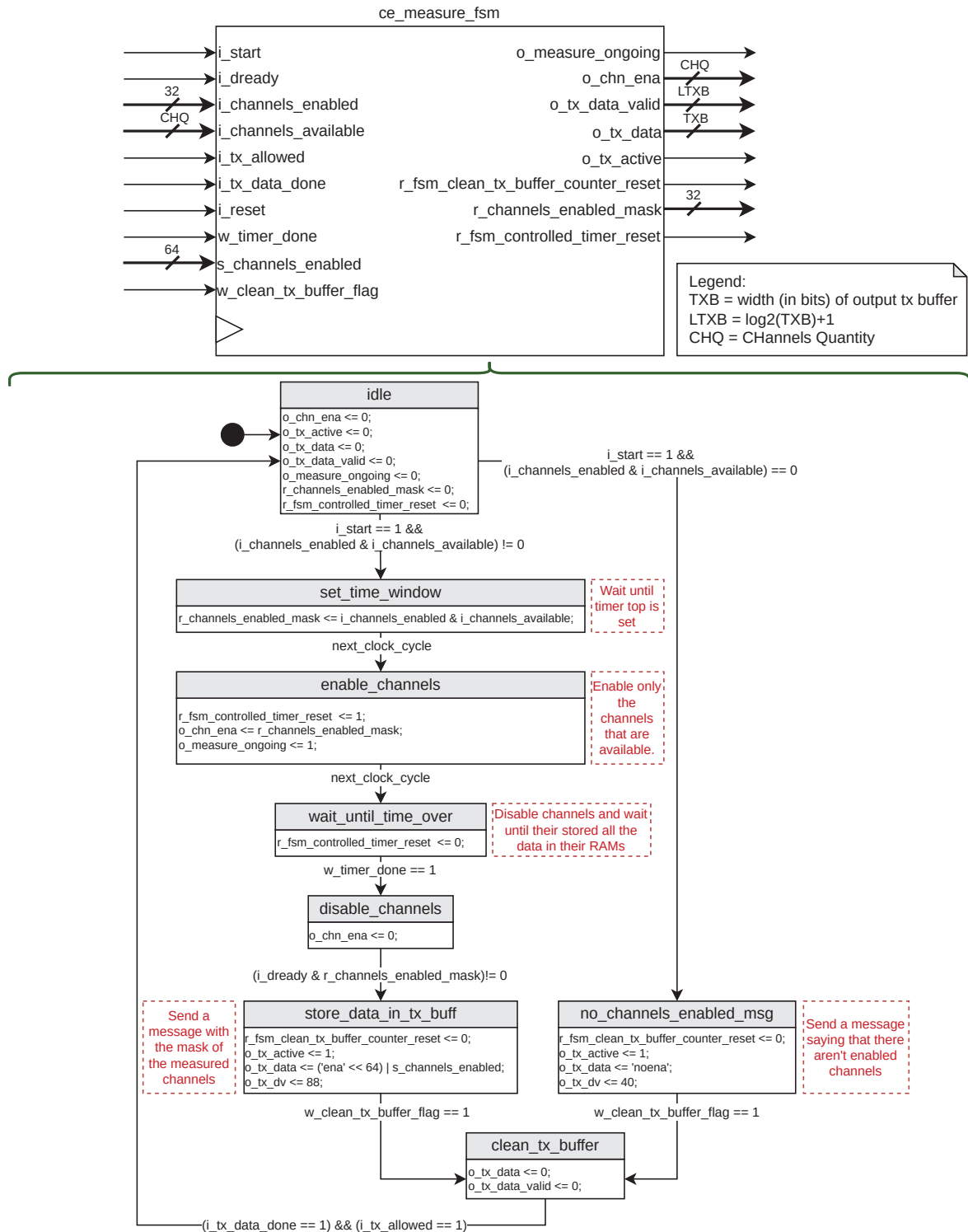


Figura 3.21: Máquina de estados del módulo *TDC Command Executor Measure*.

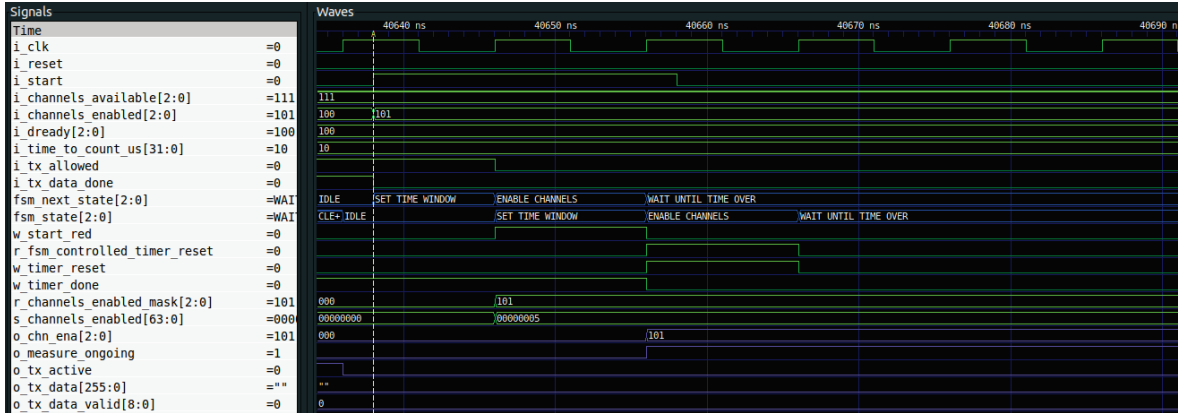


Figura 3.22: Inicio de una rutina del ejecutor *ce_measure*. En el marcador A se puede observar el inicio de la rutina, donde la señal de inicio toma un estado lógico alto.



Figura 3.23: Fin de una rutina del ejecutor *ce_measure*. En el marcador B se observa el momento en el cual el temporizador finaliza y ciclos más tarde se envía el mensaje “ena00000005” a través del *buffer* de salida global.

3.2.2.4. Lectura de los canales de entrada

El ejecutor `ce_read_channels_data` es el encargado de leer los datos almacenados en las memorias de los canales de entrada y enviarlos a través del *buffer* de salida global. Como se puede observar en la Figura 3.24, este ejecutor está compuesto por 3 etapas y una FSM.

La pieza central del ejecutor es la FSM, la cual se puede observar en la Figura 3.25. La rutina de lectura de los datos de los canales comienza cuando la FSM detecta un flanco ascendente en el puerto de entrada `i_start`. Luego, la misma interactúa con el árbitro del *buffer* global de salida para tomar el control del mismo utilizando los puertos `o_tx_active` y `i_tx_allowed`. El resto de las operaciones son realizadas en las etapas y la FSM solamente se cerciora de que se ejecuten en el orden correcto.

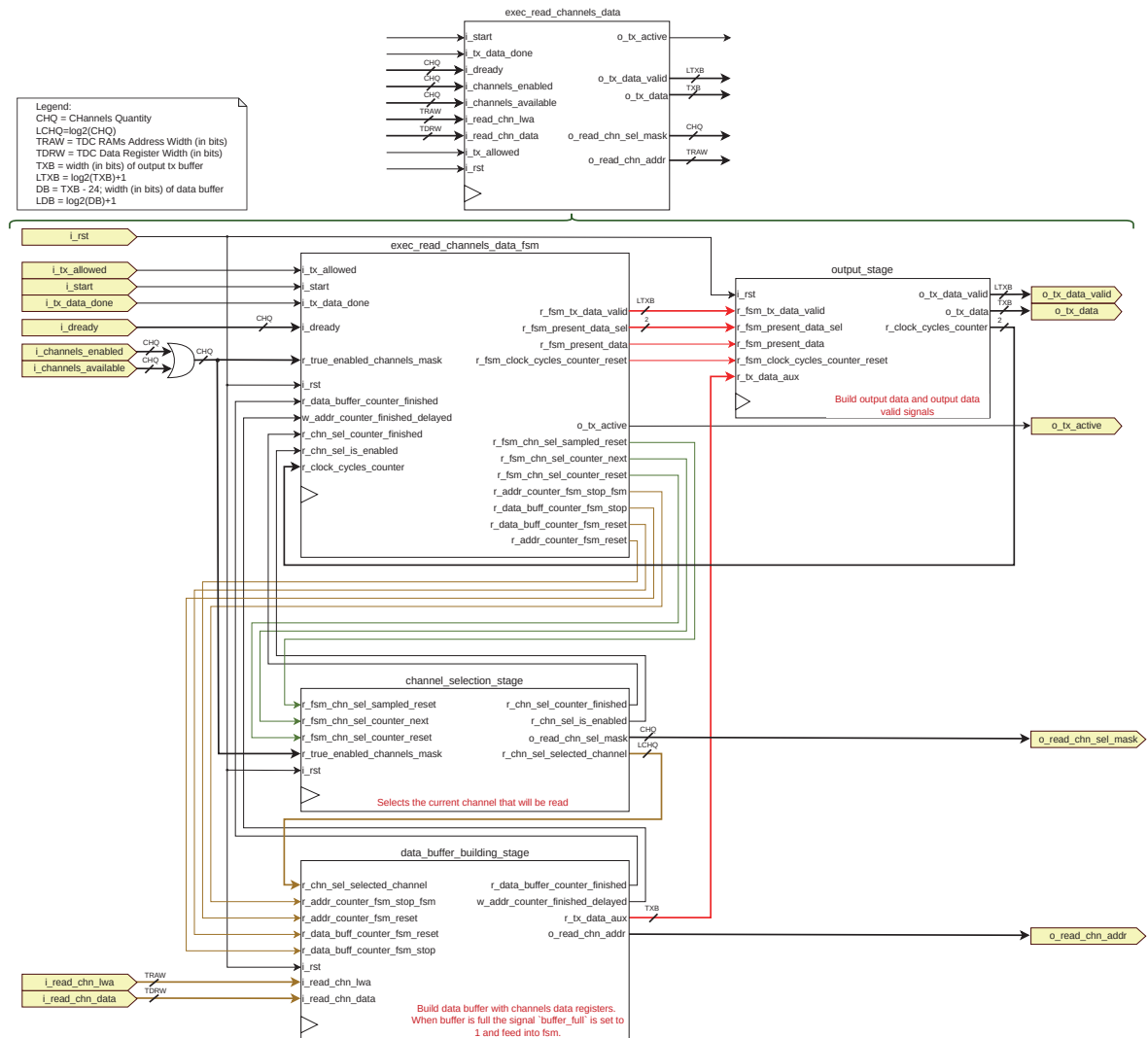


Figura 3.24: Diagrama de alto nivel del módulo encargado de leer los datos de los canales de entrada del TDC.

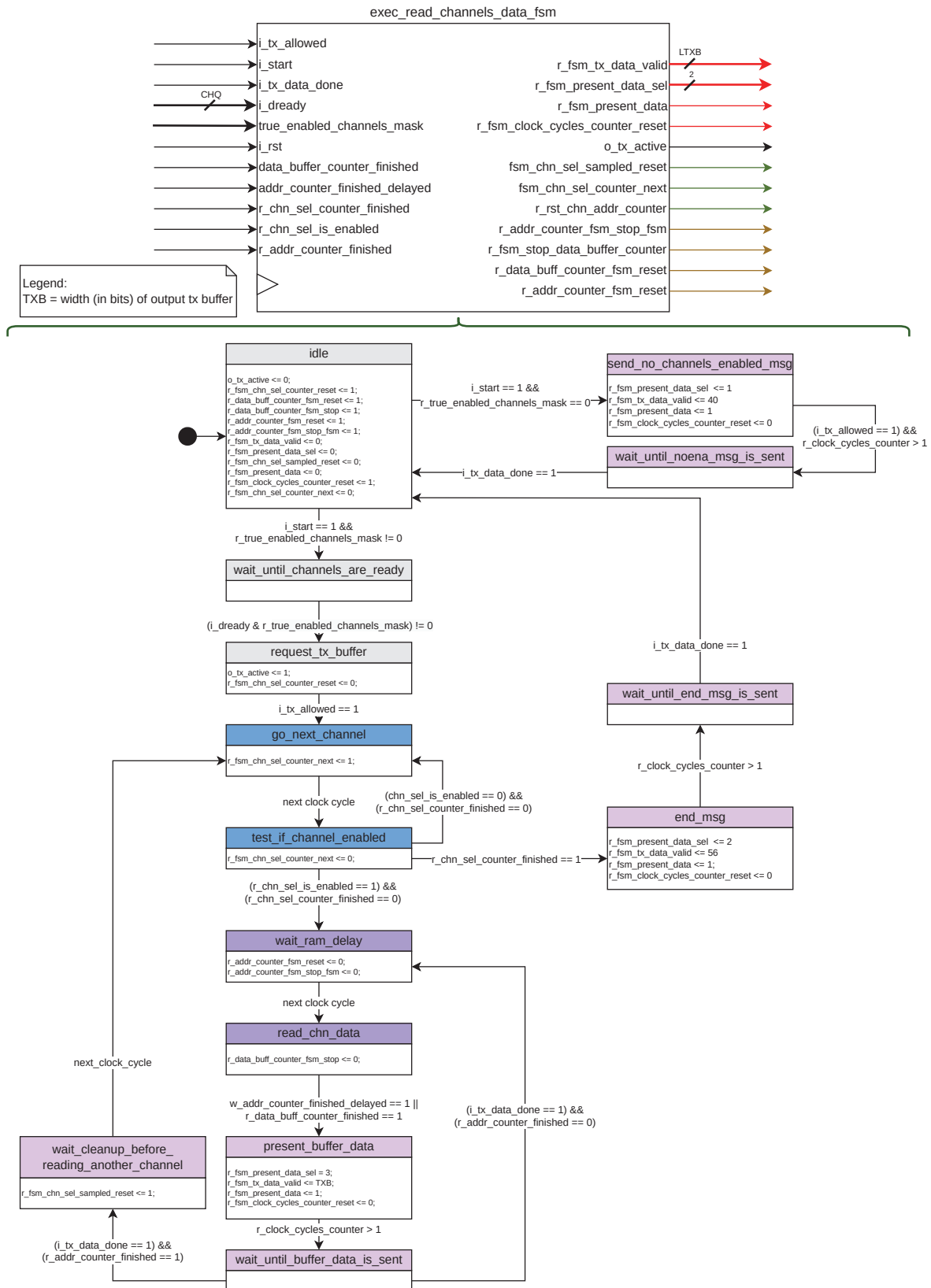
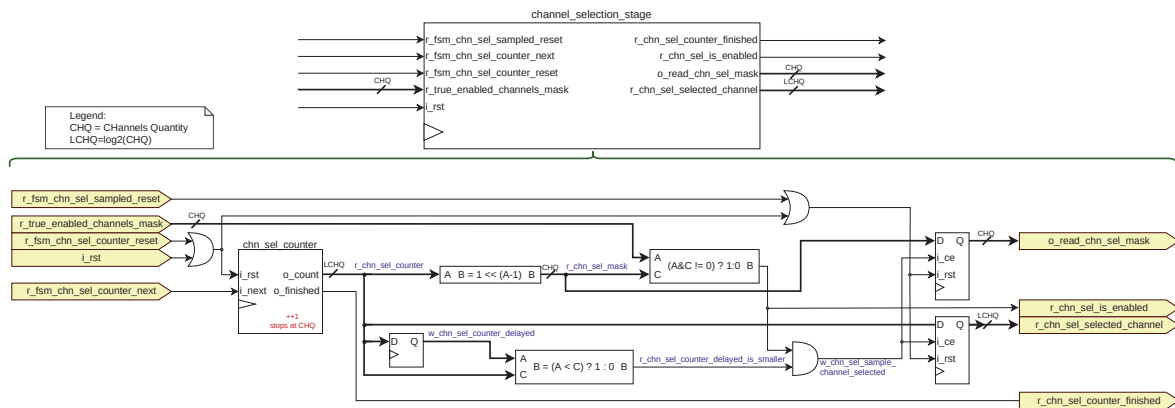


Figura 3.25: Máquina de estados del módulo encargado de leer los datos de los canales de entrada del TDC.

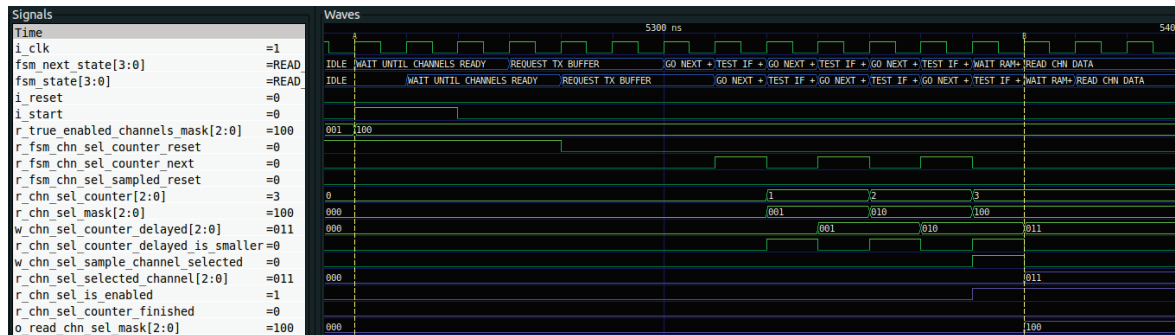
Como el ejecutor posee solamente un bus de datos para leer las memorias de los canales de

entrada se debe leer de a un canal por vez. La **etapa de selección de canales** (Figura 3.26a) es quien se encarga de seleccionar el canal del cual se leerán datos. Cuando un canal es seleccionado, externamente se conecta el bus de la memoria RAM de dicho canal con los puertos `i_read_chn_lwa`, `i_read_chn_data` y `o_read_chn_addr`.

Esta etapa posee un contador accionado por la máquina de estados, a través de la señal `r_fsm_chn_sel_counter_next`, el cual permite iterar sobre los canales de entrada. Con la salida de este contador (`r_chn_sel_counter`) se forma la máscara `r_chn_sel_mask`, la cual posee un 1 en el bit $N - 1$, donde N es el canal seleccionado actualmente (por ejemplo, el canal 3 corresponde a la máscara `0b100`). Si este canal está habilitado, se notifica a la máquina de estados a través del registro `r_chn_sel_is_enabled` y la máscara se transfiere al puerto de salida `o_read_chn_sel_mask`, lo que efectúa la selección del canal. Al finalizar la iteración sobre los posibles canales habilitados, la etapa activa la señal `r_chn_sel_counter_finished`, notificando a la máquina de estados de este evento. Este proceso se puede ver en la simulación de la Figura 3.26b.



(a) Etapa de selección del canal, dentro del ejecutor encargado de leer los datos de los canales de entrada.



(b) Simulación del ejecutor `ce_read_channels_data` donde se ve la rutina ejecutada por la etapa de selección de canales de entrada.

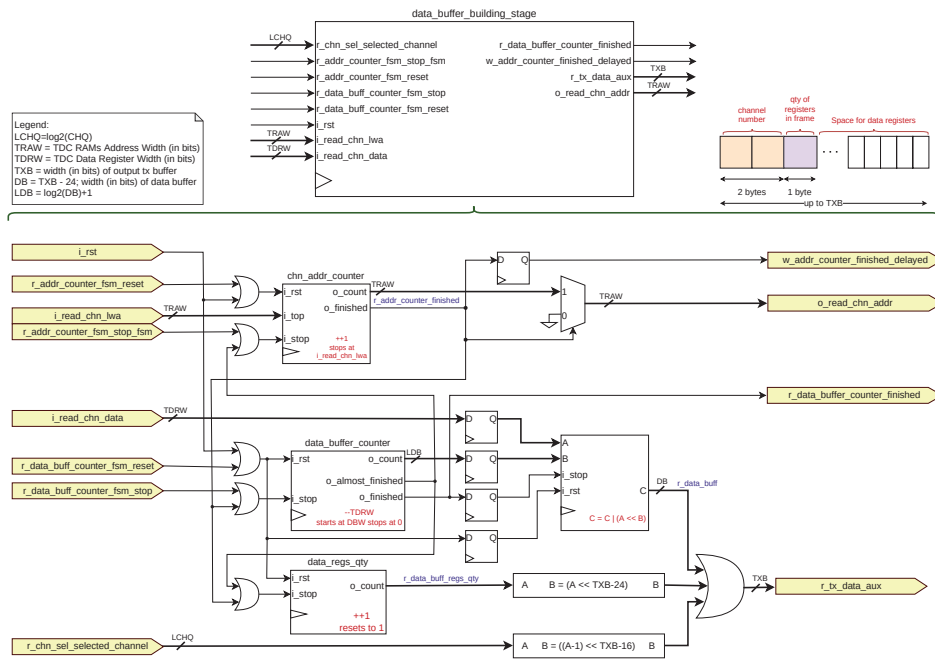
Figura 3.26: Etapa de selección de canales de entrada del ejecutor `ce_read_channels_data`.

La **etapa de construcción de buffers de datos** (Figura 3.27a) es activada por la máquina de estados luego de que un canal habilitado sea seleccionado. Esta etapa posee un contador el cual es utilizado para recorrer la RAM del canal hasta llegar a la última dirección de memoria escrita (la cual está determinada por el puerto de entrada `i_read_chn_lwa`). Los registros de datos extraídos de la RAM del canal se van posicionando de manera adyacente en un *buffer* hasta que el mismo se llena o se llega a la última dirección de memoria. Esto se puede visualizar en la simulación de la Figura 3.27b donde los valores `0x9`, `0x1`, `0xA` y `0x64` del puerto de entrada `i_read_chn_data` se van almacenando en el registro `r_data_buff`.

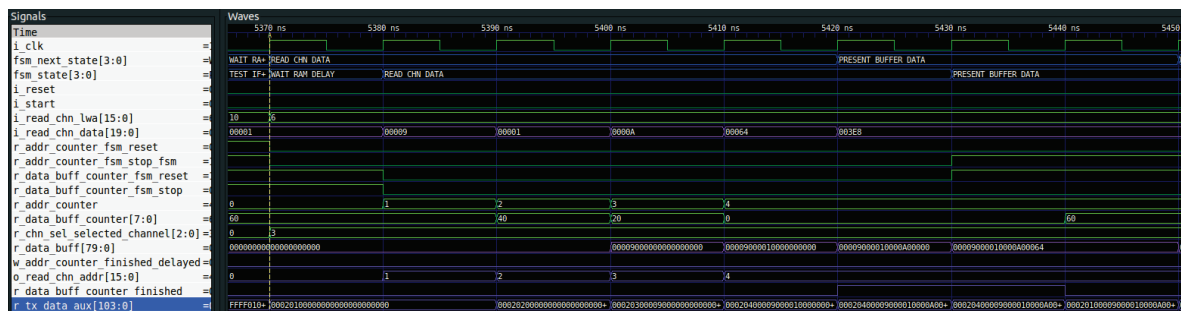
Cuando se llena el *buffer* o se llega a la última posición de memoria escrita, se crea un nuevo *buffer* `r_tx_data_aux` el cual posee en los dos primeros bytes el número del canal seleccionado, en el tercer byte la cantidad de registros de datos almacenados en el *buffer* y en el resto de los bytes

se encuentran los registros de datos. En la simulación de la Figura 3.27b, el valor de los primeros 3 bytes de este *buffer* es $0x000204$, lo que indica que los datos corresponden al canal $0x2$ y que hay $0x4$ registros en el *buffer*.

Al mismo tiempo que se crea el *buffer* `r_tx_data_aux`, se avisa a la máquina de estados que hay un nuevo *buffer* para ser enviado con la señal `r_data_buff_counter_finished`. Luego, la máquina de estados detiene el contador utilizado para recorrer la RAM del canal (ver señal `r_addr_counter_fsm_stop_fsm`) y reinicia el contador utilizado para llenar el *buffer* de datos (ver señal `r_data_buff_counter_fsm_reset`), estas señales se mantienen en este estado hasta que el *buffer* es transferido. Cuando el *buffer* es enviado, si aún quedan direcciones de memoria por leer en la RAM, se repite el proceso, creando un nuevo *buffer* y enviándolo.



(a) Etapa de construcción de *buffers* de datos, dentro del ejecutor encargado de leer los datos de los canales de entrada.



(b) Simulación del ejecutor `ce_read_channels.data` donde se ve la rutina ejecutada por la etapa de construcción de *buffers* de datos.

Figura 3.27: Etapa de construcción de *buffers* de datos del ejecutor `ce_read_channels.data`.

La **etapa de salida** (Figura 3.28) es la encargada de presentar en el *buffer* de salida global (`o_tx_data`) el *buffer* con los registros de datos de los canales, de enviar el mensaje de fin de transmisión (“enaread”) y de enviar el mensaje que indica que no hay canales habilitados (“noena”).

La máquina de estados es quien define qué mensaje será enviado utilizando el registro `r_fsm_present_data_sel`. Si el valor de este registro es $0x3$, entonces se almacena en el *buffer* de salida una trama de registros de datos, lo cual se puede observar en la simulación de la Figura 3.29a. Por otro

lado, como se puede ver en la Figura 3.29b, cuando ya no hay más registros de datos por transmitir, este registro toma el valor $0x2$ y en el *buffer* de salida se deposita el mensaje “enaread”, que indica que ya se leyeron todos los canales habilitados.

El valor del puerto de salida *o_tx_data* se mantiene hasta que es sobrescrito nuevamente por la máquina de estados. Sin embargo, como puede haber una lógica externa muestreando el valor del puerto *o_tx_data_valid* para detectar cuando debe enviar datos, el valor de este puerto solamente se mantiene durante 4 ciclos de reloj y luego se limpia llenándolo con ceros.

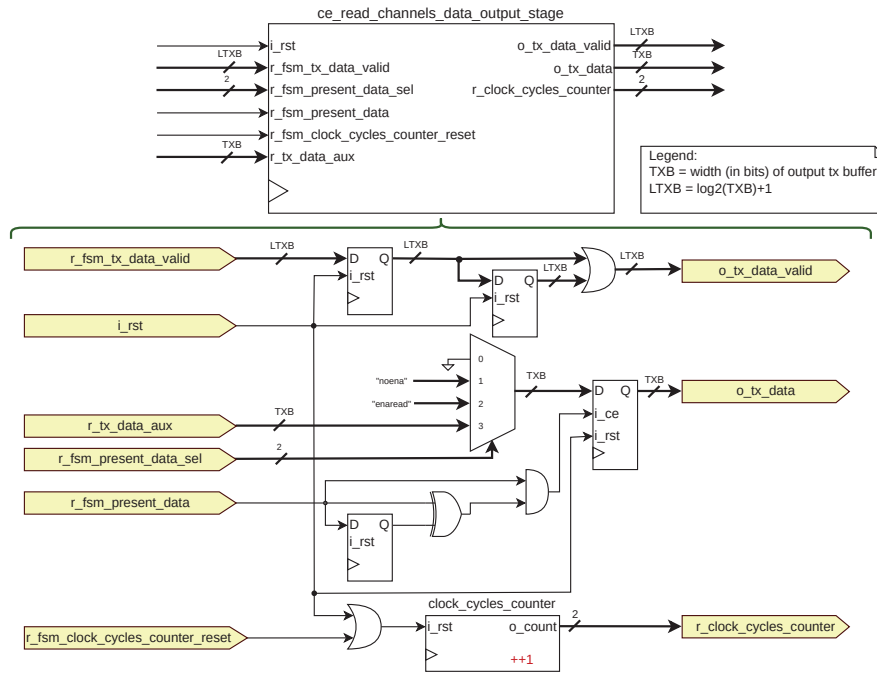
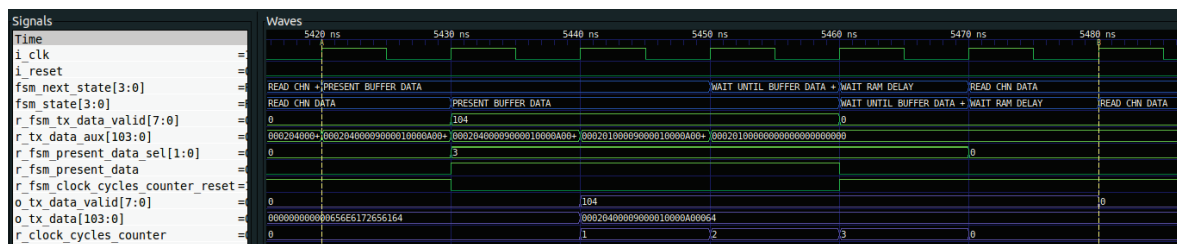
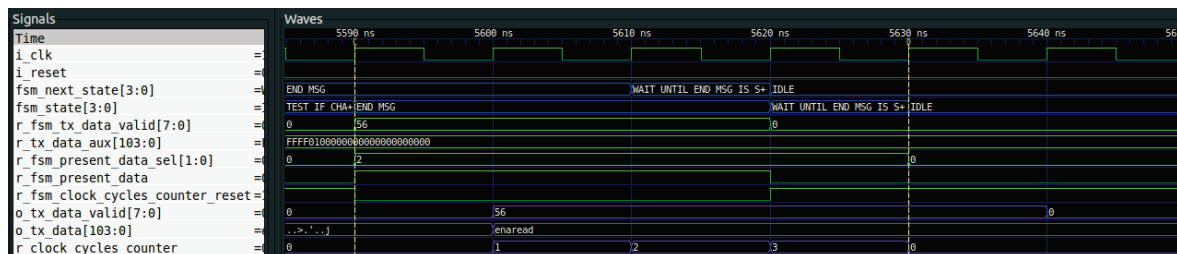


Figura 3.28: Etapa de salida, dentro del ejecutor encargado de leer los datos de los canales de entrada.



(a) Simulación del ejecutor *ce_read_channels_data* donde se ve la rutina ejecutada por la etapa de salida al transmitir un *buffer* con registros de datos.



(b) Simulación del ejecutor *ce_read_channels_data* donde se ve la rutina ejecutada por la etapa de salida al transmitir el mensaje de finalización de transmisión de registros de datos: “enaread”.

Figura 3.29: Simulación de la etapa de salida del ejecutor *ce_read_channels_data*.

3.3. Procesador de tramas HDLC-lite

En esta sección se detalla el protocolo de entramado que utiliza el TDC para comunicarse con el usuario.

La implementación de este tipo de protocolo en una comunicación serie es fundamental para mejorar la fiabilidad del sistema y garantizar una transmisión confiable de datos. Un protocolo de entramado permite separar los datos transmitidos en paquetes discretos (tramas), donde ambos extremos conocen el formato de la trama y pueden reconocerlas. Además, estos protocolos incluyen mecanismos de detección y corrección de errores, como el CRC, que permiten detectar errores en la transmisión de datos causados por ruido eléctrico en la línea u otros factores.

Para esta aplicación específica, se implementó un protocolo de entramado denominado HDLC-lite, el cual se ha consolidado como una opción muy popular en las interfaces de comunicación que se establecen entre procesadores de propósito general y co-procesadores de red. Cabe destacar que este protocolo en particular es altamente recomendado para la transmisión de las tramas pertenecientes al protocolo Spinel sobre la interfaz UART, tal y como se indica en la fuente bibliográfica [37].

En la Figura 3.30 se puede visualizar el formato de la trama HDLC-lite. Dentro del campo *payload* de esta trama se almacenan los mensajes intercambiados entre el usuario y el TDC.

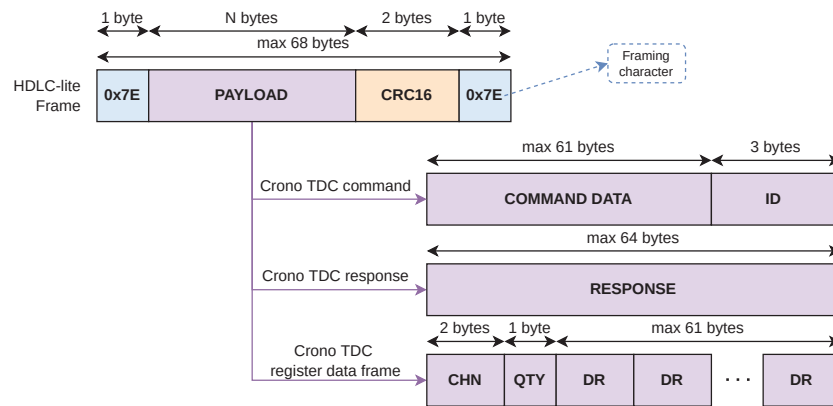


Figura 3.30: Entramado HDLC-lite con los distintos mensajes encontrados en la comunicación entre el TDC y el usuario.

En el protocolo HDLC-lite, las tramas de datos se identifican mediante el uso del carácter de entramado $0x7E$, de manera que al detectar este byte, el receptor puede reconocer la recepción de una trama de este protocolo.

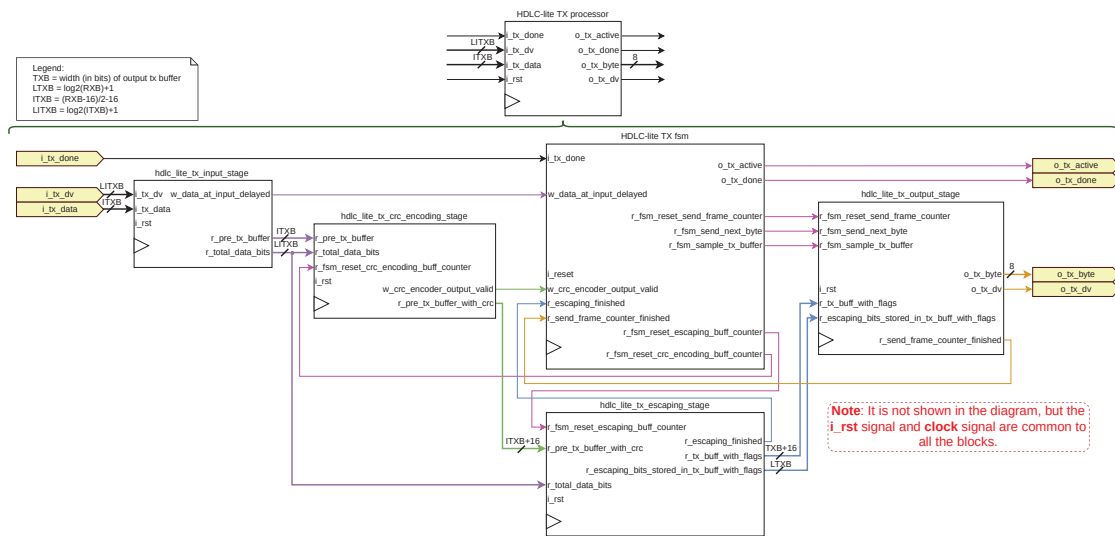
Sin embargo, en el caso de que el carácter de entramado $0x7E$ se encuentre dentro de los datos de la trama, el transmisor utiliza un proceso de escape (*escaping*) para garantizar la integridad de la información transmitida. Este proceso consiste en anteponer el carácter de escape $0x7D$ al byte en cuestión, y almacenar el resultado de la operación XOR entre el carácter escapado y el valor $0x20$ en el byte siguiente. El mismo procedimiento se lleva a cabo si uno de los bytes coincide con el carácter de escape ($0x7D$). El transmisor calcula el CRC de los datos previamente a realizar el escapado, de manera de contemplar el caso en el cual el CRC posea alguno de estos caracteres.

Por su parte, el receptor debe llevar a cabo un proceso de decodificación para revertir el proceso de escape y recuperar los datos transmitidos. En primer lugar, el receptor verifica que no haya ningún carácter de escape ($0x7D$) en la trama recibida. En caso de que se detecte un carácter de escape, se remueve y se realiza la operación XOR con el valor $0x20$ sobre el byte siguiente para recuperar el dato original. Después de la decodificación, el receptor calcula el valor del CRC de los datos recibidos y lo compara con el valor del CRC incluido en la trama por el transmisor. Si estos valores no coinciden, se considera que ha habido una alteración involuntaria en la información transmitida, lo que puede indicar la presencia de ruido eléctrico en la línea.

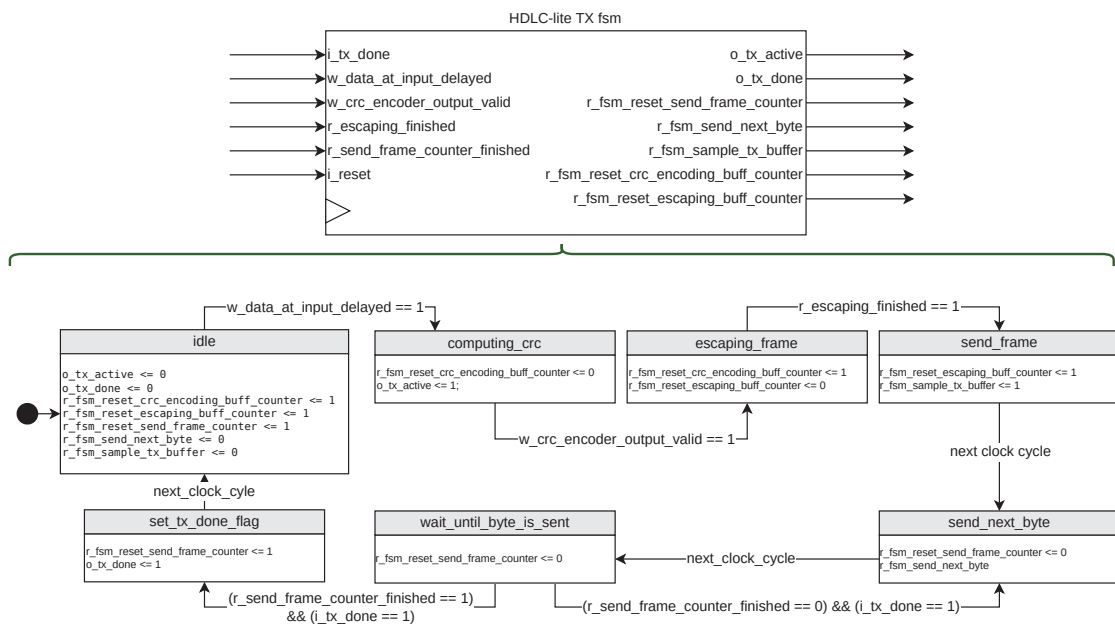
Para esta aplicación, se diseñó un transmisor de tramas (ver Sección 3.3.1) y un receptor de tramas (ver Sección 3.3.2) los cuales trabajan independientemente el uno del otro. El CRC utilizado en esta aplicación es el CRC16-CCITT, cuyo polinomio es $x^{16} + x^{12} + x^5 + 1$ ($0x1021$). Para el cálculo del CRC de cada trama se diseñó una calculadora de CRC basada en un *Linear Feedback Shift Register* (LFSR), a la cual se le deben ingresar datos en forma serie (bit a bit) y luego de 16 ciclos de reloj calcula el CRC16-CCITT de los mismos, pero el diseño en detalle de dicho bloque escapa a los alcances de este informe.

3.3.1. Transmisor HDLC-lite

En la Figura 3.31a se puede observar el módulo transmisor de tramas diseñado. Este módulo fue separado en varias etapas las cuales son coordinadas por una FSM central que puede visualizarse en detalle en la Figura 3.31b.



(a) Módulo transmisor de tramas HDLC-lite diseñado, con su máquina de estados y distintas etapas.



(b) Máquina de estados del módulo transmisor de tramas HDLC-lite.

Figura 3.31: Composición del transmisor de tramas HDLC-lite.

En la Figura 3.32 se presenta la etapa de entrada, la cual se encarga de detectar la presencia de un nuevo *buffer* de datos que debe ser transmitido. Esto se logra mediante el muestreo del puerto de entrada `i_tx_dv`. Cuando este puerto adquiere un valor M , distinto de cero, indica que hay M bits válidos en el puerto de entrada `i_tx_data`. Al suceder dicho evento, se muestrean los valores de los puertos `i_tx_dv` y `i_tx_data` y se almacenan en los registros `r_total_data_bits` y `r_pre_tx_buffer`, correspondientemente.

En la Figura 3.33 se ilustra un fragmento de simulación que muestra la operación de este bloque de *hardware*. En dicha simulación, el *buffer* a enviar posee el dato `0x123456789A`, compuesto por 37 bits.

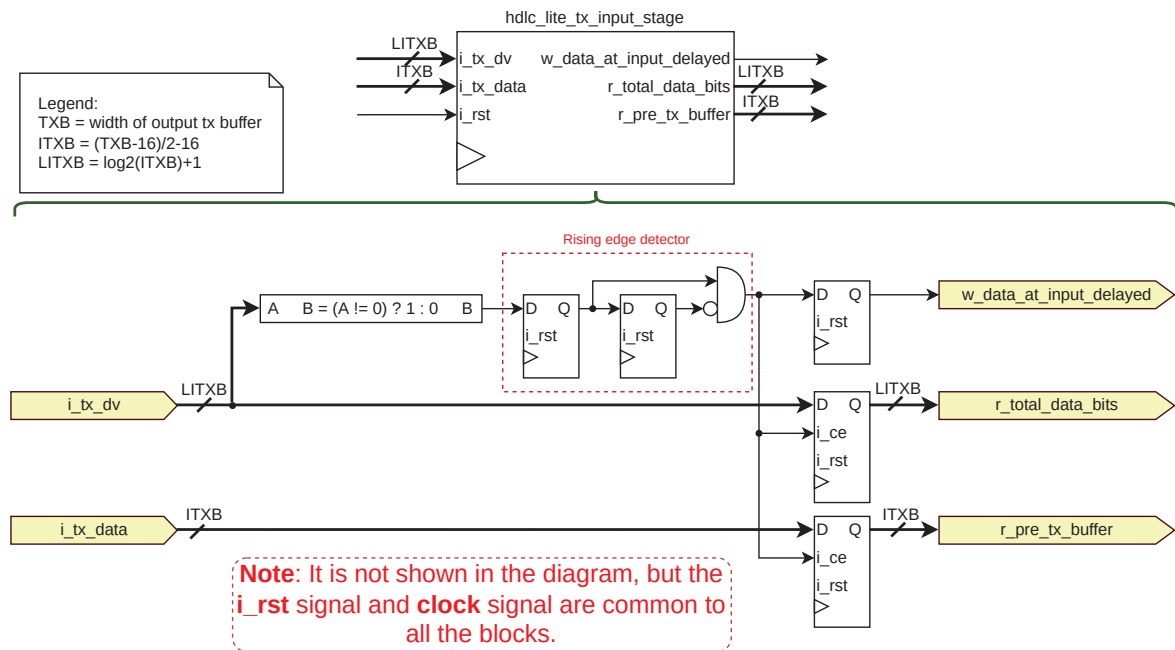


Figura 3.32: Etapa de entrada del módulo transmisor de tramas HDLC-lite.

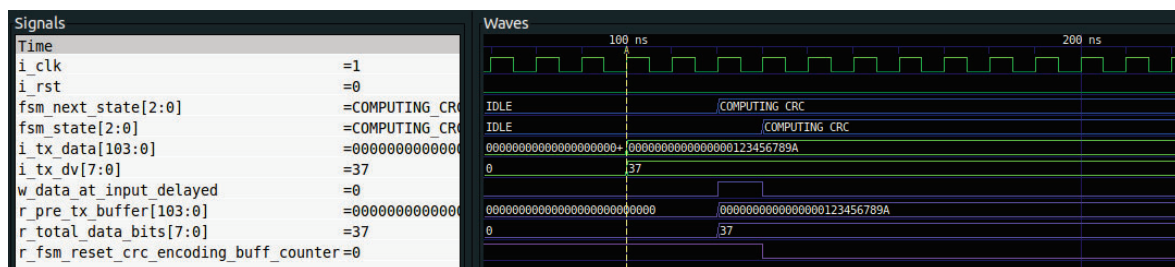


Figura 3.33: Fragmento de una simulación en el cual se visualiza la etapa de entrada del módulo transmisor de tramas HDLC-lite en funcionamiento.

En la Figura 3.34 se puede observar la etapa de cálculo del CRC de la trama. Esta etapa utiliza el contador decreciente `r_crc_encoding_buffer_counter` y un *barrel shifter* para ingresar todos los bits del *buffer* de datos en el módulo que calcula su CRC. Cuando el contador llega a 0 significa que todos los bits han sido introducidos en la calculadora de CRC, por lo cual, el contador le indica a esta última que ya no hay más datos válidos en su entrada a través de la señal `r_crc_encoding_buff_counter_finished`. La finalización del cálculo del CRC se indica con la señal `w_crc_encoder_output_valid`, y al llegar dicho momento los datos y el CRC se almacenan en el registro `r_pre_tx_buffer_with_crc`.

En la simulación de la Figura 3.35 se puede ver el cálculo del CRC-CCITT de la trama `0x123456789A`, el cual es `0xE9AC`. Cuando se ingresa el último bit, la calculadora se demora 16

ciclos de reloj en terminar de calcular el CRC y luego este valor aparece agregado en los bits menos significativos del *buffer* `r_pre_tx_buffer_with_crc` junto a los datos.

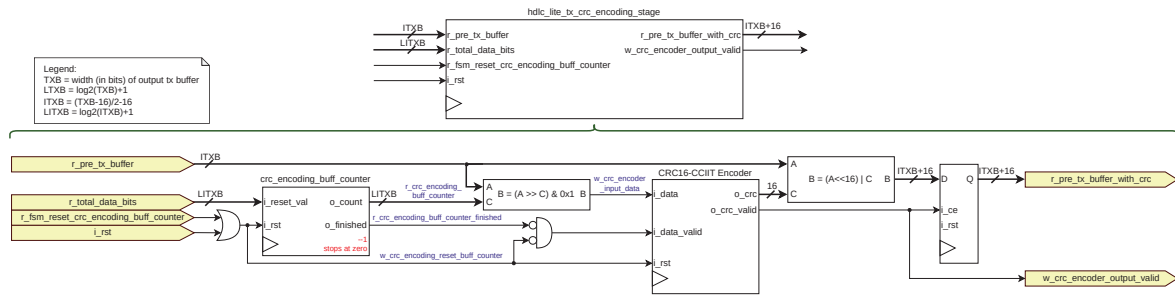


Figura 3.34: Etapa de cálculo del CRC del módulo transmisor de tramas HDLC-lite.

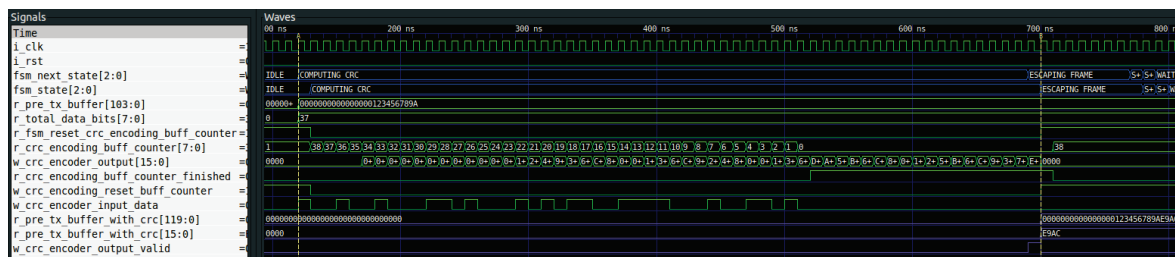


Figura 3.35: Fragmento de una simulación en el cual se visualiza la etapa de cálculo de CRC del módulo transmisor de tramas HDLC-lite en funcionamiento.

La etapa de codificación (Figura 3.36) es activada por la FSM cuando finaliza el cálculo del CRC. En esta etapa se reinicia el contador `r_escaping_buff_counter`, el cual, junto a un *barrel shifter*, es utilizado para recorrer los bytes de la trama. A medida que este contador avanza, un conjunto de comparadores verifica si el byte seleccionado (`w_escaping_byte_to_analyze`) debe ser escapado.

Si el caracter seleccionado no es el caracter de entramado (`0x7E`) o de escape (`0x7D`), se almacena el byte en el *buffer* `r_tx_buff_without_flags`, como se puede observar en la simulación de la Figura 3.37, en la cual ningún caracter debe ser escapado. Sin embargo, si el byte seleccionado es alguno de estos dos caracteres, se almacena en el *buffer* el byte de escape `0x7D` seguido del byte seleccionado después de realizar la operación XOR con el valor `0x20`, tal como se muestra entre los marcadores C y D de la simulación de la Figura 3.38. De esta manera se crea el *buffer* `r_tx_buff_with_flags`, el cual contiene la trama HDLC-lite completa, incluyendo los caracteres de entramado `0x7E`.

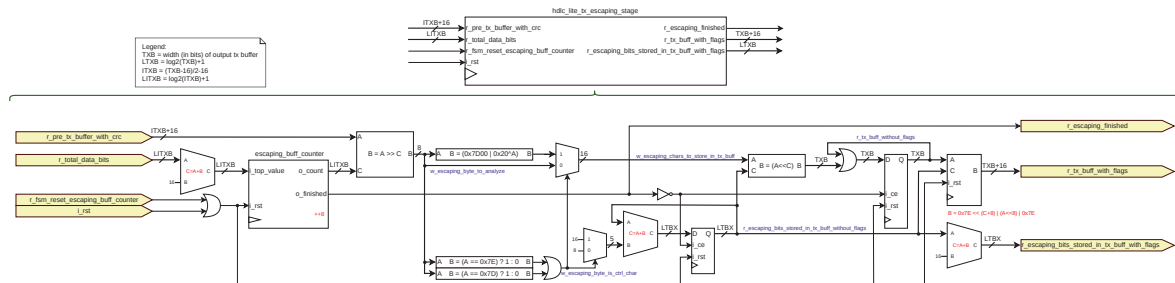


Figura 3.36: Etapa de escapado del módulo transmisor de tramas HDLC-lite.

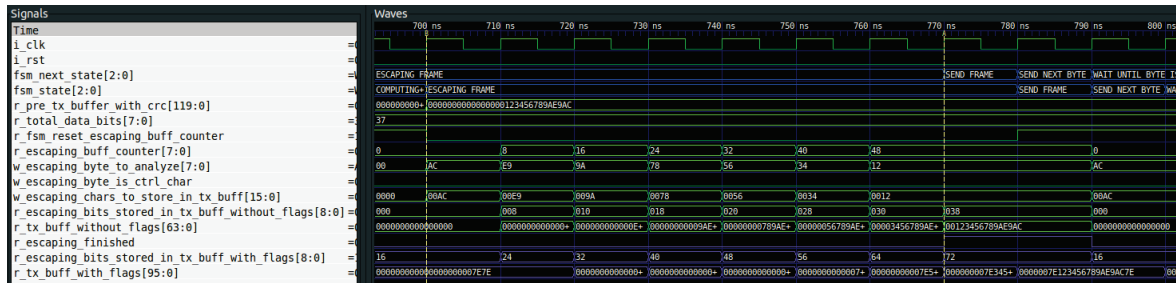


Figura 3.37: Simulación donde se puede ver la etapa de escapado del módulo transmisor de tramas HDLC-lite procesando una trama que no posee caracteres de control.

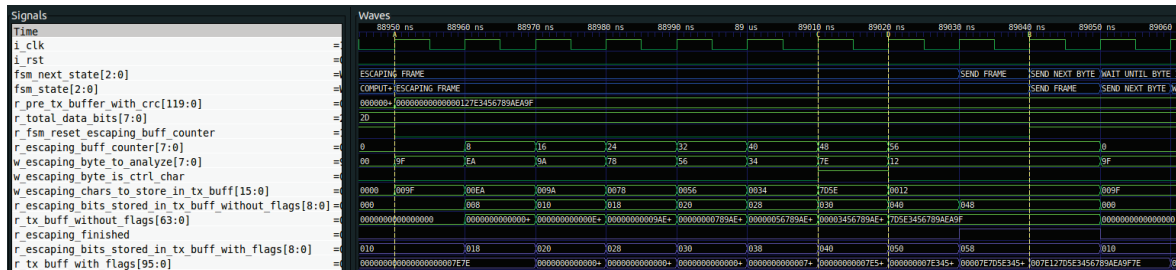


Figura 3.38: Simulación donde se puede visualizar la etapa de escapado del módulo transmisor de tramas HDLC-lite codificando el carácter $0x7E$ entre los marcadores C y D.

Una vez que la rutina de escapado finaliza, la señal `r_escaping_finished` se activa, notificando a la FSM de dicho evento. En ese momento es cuando la etapa de salida, la cual se representa en la Figura 3.39, es habilitada por la FSM. Durante la habilitación de la etapa de salida, la FSM reinicia el contador `r_send_frame_counter`, el cual es utilizado para enviar los bytes de la trama uno a uno. También, en dicho momento se muestrean los valores del *buffer* `r.tx_buff_with_flags`, que contiene la trama, y del registro `r_escaping_bits_stored_in_tx_buff_with_flags`, el cual posee la cantidad de bits totales de la trama.

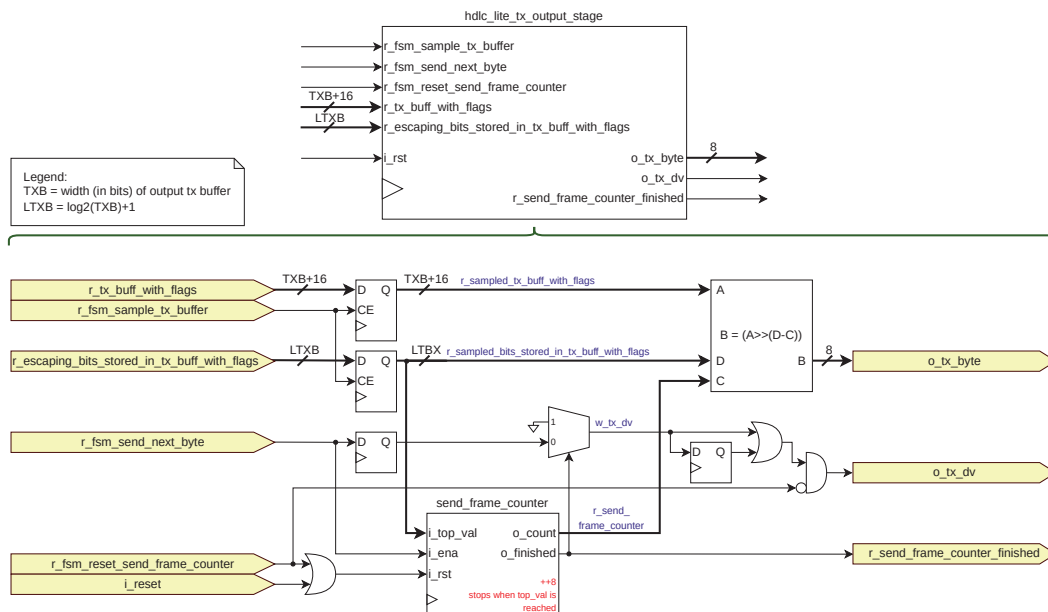


Figura 3.39: Etapa de salida del módulo transmisor de tramas HDLC-lite.

Cada vez que se envía un byte, la máquina de estados activa la señal `r_fsm_send_next_byte`. Al ocurrir esto, la etapa de salida presenta el byte a enviar en el puerto `o_tx_byte` y activa el puerto `o_tx_dv` durante dos ciclos de reloj para indicar que hay un byte válido en la salida. El puerto de entrada `i_tx_done` informa a la FSM que el byte ha sido transferido. Si todavía hay bytes pendientes de envío, la señal `r_fsm_send_next_byte` se activa nuevamente, lo que dispara el envío del siguiente byte. Cuando no quedan más bytes por enviar, el contador envía la señal `r_send_frame_counter_finished` a la FSM, la cual finaliza el proceso de envío de la trama. En la simulación de la Figura 3.40 se puede observar este proceso, en el cual la etapa de salida envía los bytes de la trama `0x7E123456789AE9AC7E`.

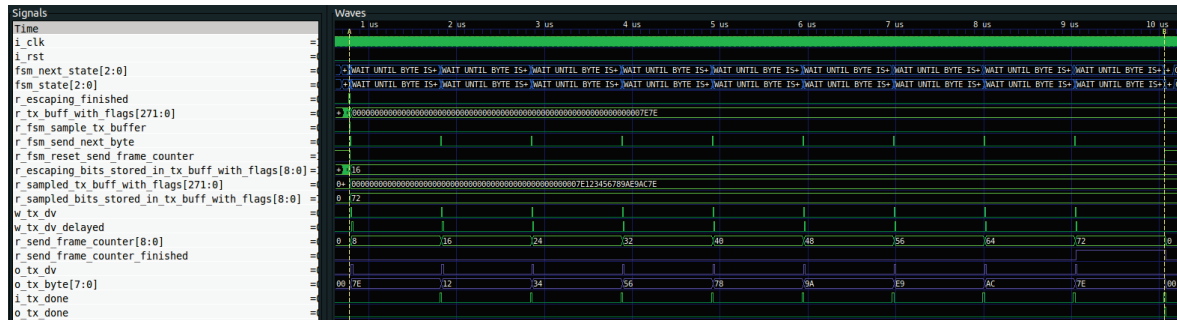


Figura 3.40: Fragmento de una simulación en el cual se visualiza la etapa de salida del módulo transmisor de tramas HDLC-lite enviando los bytes de la trama `0x7E123456789AE9AC7E`.

3.3.2. Receptor HDLC-lite

El módulo encargado de recibir los bytes enviados por el usuario y detectar si los mismos conforman una trama se puede observar en la Figura 3.41. De manera similar a como fueron diseñados otros módulos, este bloque está compuesto por una máquina de estados principal (Figura 3.42) que controla varias etapas. Cada una de estas etapas realiza una tarea distinta para lograr el objetivo del módulo y en esta sección se explorarán cada una de ellas.

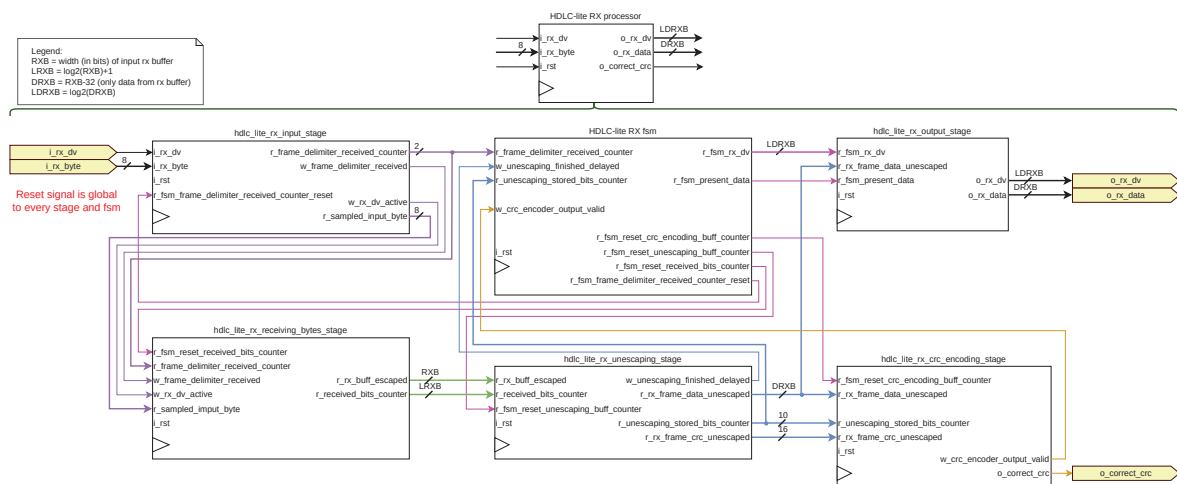


Figura 3.41: Módulo receptor de tramas HDLC-lite diseñado, con su máquina de estados y distintas etapas.

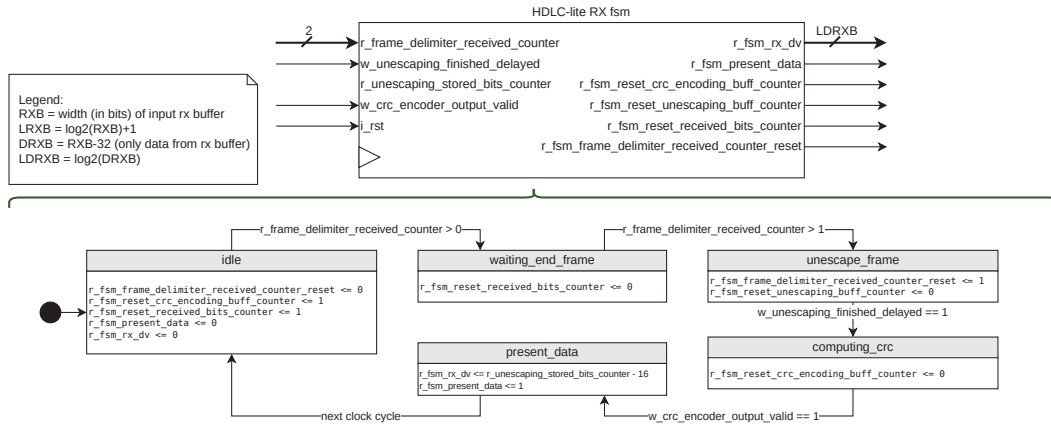


Figura 3.42: Máquina de estados del módulo receptor de tramas HDLC-lite.

La Figura 3.43 muestra la etapa de entrada del módulo. Esta etapa es responsable de detectar el carácter delimitador de trama $0x7E$, determinando cuando comienza y termina una trama. Cada vez que este carácter es detectado el módulo incrementa un contador (`r_frame_delimiter_received_counter`). Este contador es utilizado por la FSM para determinar cuando está comenzando y finalizando una trama. Este mecanismo se puede observar en la simulación de la Figura 3.44, donde el módulo está recibiendo la trama $0x7E123456789AE9AC7E$ y el contador de caracteres de entramado se incrementa cada vez que se recibe uno de ellos. Además, esta etapa se encarga de muestrear el byte de entrada (`i_rx_byte`) cada vez que ocurre un flanco ascendente en el puerto de entrada `i_rx_dv`, el cual indica que hay datos válidos (ver señales `w_rx_dv_active` y `r_sampled_input_byte` en Figura 3.44).

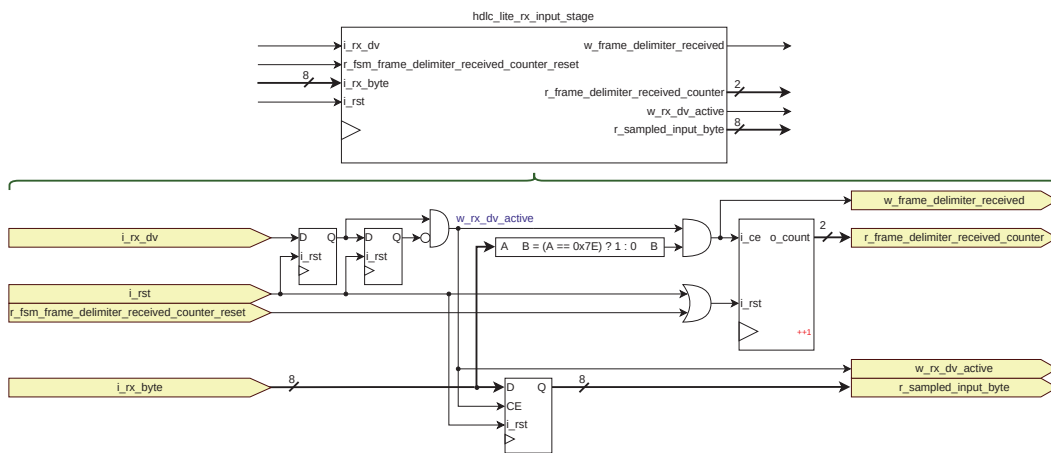


Figura 3.43: Etapa de entrada del módulo receptor de tramas HDLC-lite.

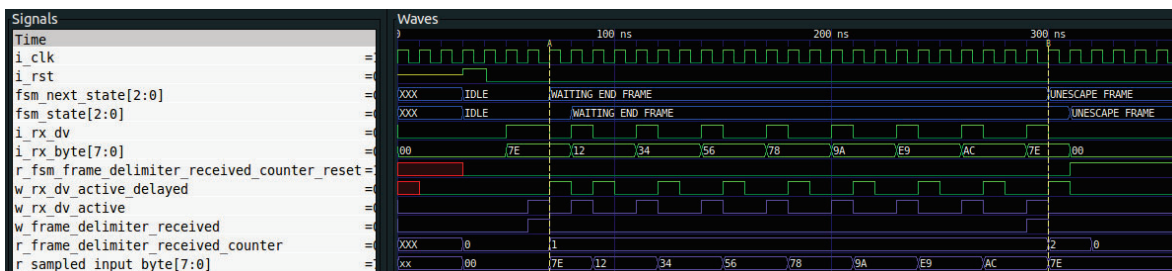


Figura 3.44: Fragmento de una simulación en el cual se visualiza la etapa de entrada del módulo receptor de tramas HDLC-lite en funcionamiento.

La etapa de recepción de bytes, la cual se encuentra en la Figura 3.45, trabaja en conjunto con la etapa de entrada, almacenando los bytes muestreados por esta última en un *buffer*. Cuando esta etapa recibe un byte incrementa en 8 el contador `r_received_bits_counter` y utiliza su valor para almacenar el byte en la ubicación correspondiente del *buffer* `r_rx_buff_escaped`. En la simulación de la Figura 3.46, se puede observar que el valor final que toma este contador es 56, que son los bits totales entre los datos y el CRC de la trama de prueba `0x7E123456789AE9AC7E`.

El *buffer* `r_received_bits_next_byte_to_store` posee el byte recibido desplazado M posiciones a la izquierda, donde M es la diferencia entre el tamaño del *buffer* y el valor del contador. En el ciclo de reloj posterior al muestreo del byte, el mismo se almacena en el *buffer* `r_received_bits_buff_escaped_pre_shift` junto a los bytes recibidos previamente. Este último *buffer* es desplazado hacia la derecha de manera que el último byte de la trama esté ubicado en los 8 bits menos significativos del *buffer* `r_rx_buff_escaped`. Este proceso se puede observar con detalle en la simulación de la Figura 3.46.

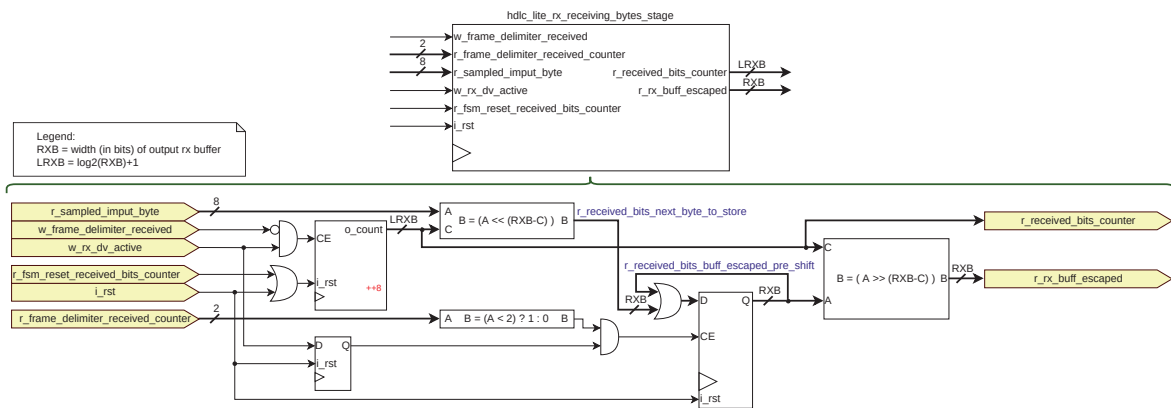


Figura 3.45: Etapa de recepción de bytes del módulo receptor de tramas HDLC-lite.

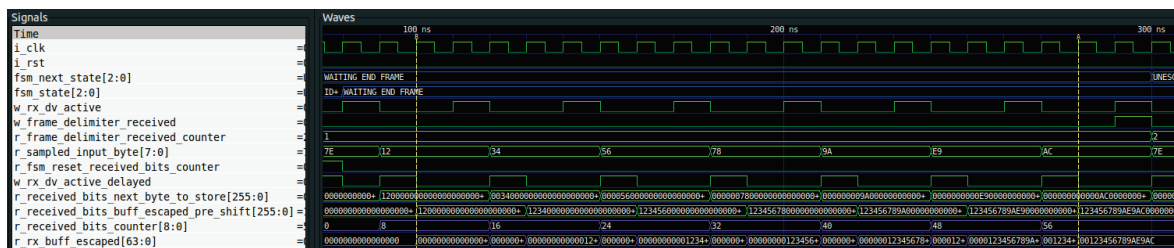


Figura 3.46: Fragmento de una simulación en el cual se visualiza la etapa de recepción de bytes del módulo receptor de tramas HDLC-lite en funcionamiento.

En la Figura 3.47 se puede observar la etapa de decodificación, la cual es la responsable de revertir el proceso de escape realizado por el transmisor. Esta etapa utiliza el contador `r_unescaping_buff_counter` y un *barrel shifter* para verificar si algún byte de la trama es el caracter de escape (`0x7D`). En el caso en el cual detecte un caracter de escape, realiza la operación XOR con el caracter escapado para recuperar la información original.

En la simulación de la Figura 3.48, se puede apreciar que la trama de prueba no presenta caracteres de escape, lo que permite el almacenamiento directo de todos sus bytes en el *buffer* `w_rx_frame_unescaped_shifted`. Por otro lado, en la simulación de la Figura 3.49, se encuentra el caracter de escape en la trama. En este caso, el caracter de escape no se almacena en el *buffer* y al siguiente byte se lo almacena luego de realizar la operación XOR.

Luego de decodificar la trama, esta etapa separa los datos y el CRC, almacenándolos en los *buffers* `r_rx_frame_data_unescaped` y `r_rx_frame_crc_unescaped`, correspondientemente.

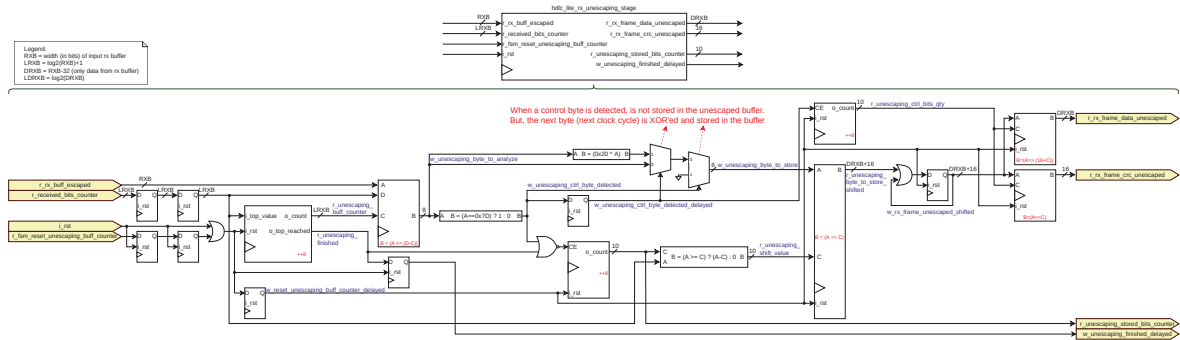


Figura 3.47: Etapa de decodificación del módulo receptor de tramas HDLC-lite.

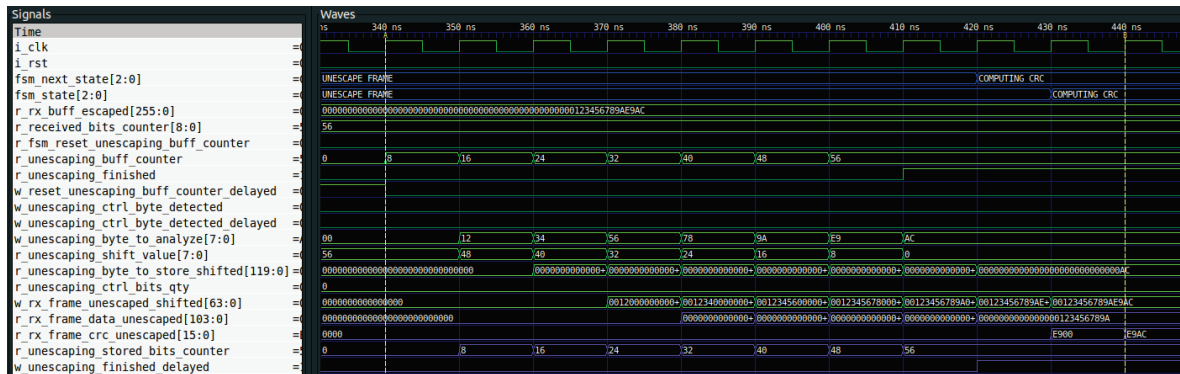


Figura 3.48: Fragmento de una simulación en el cual se visualiza la etapa de decodificación del módulo receptor de tramas HDLC-lite en funcionamiento. En este caso en particular no hay ningún carácter de control en la trama.

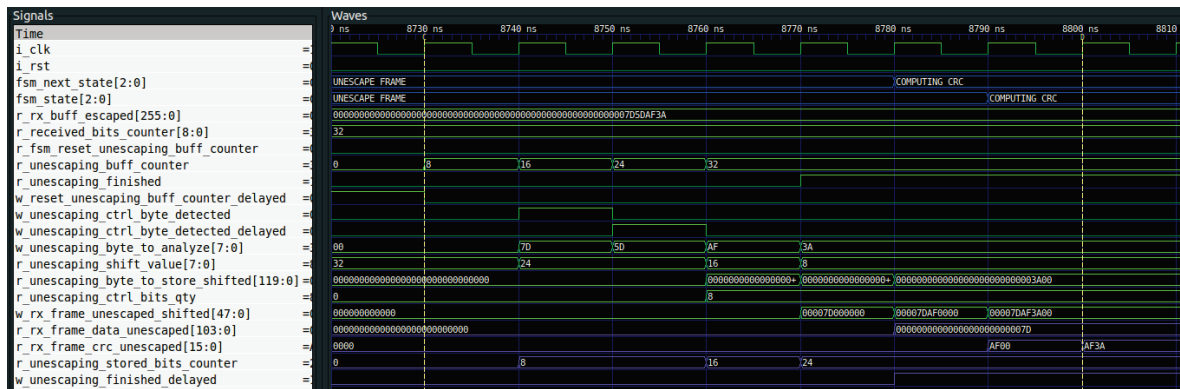


Figura 3.49: Fragmento de una simulación en el cual se visualiza la etapa de decodificación del módulo receptor de tramas HDLC-lite en funcionamiento. En este caso se encuentra el carácter de control `0x7D` dentro de la trama.

La siguiente etapa en realizar su trabajo es la etapa de cálculo de CRC, la cual se puede observar en la Figura 3.50. Utilizando el contador `r_crc_encoding_buff_counter` y un *barrel shifter*, esta etapa ingresa bit a bit la trama decodificada en una calculadora de CRC. Este contador es decremental y posee un valor inicial igual a la cantidad de bits de datos de la trama. En el caso de la simulación de la Figura 3.51, el campo de datos de la trama es `0x123456789A`, contabilizando un total de 40 bits. De esta manera, los bits se ingresan a la calculadora de CRC empezando por el bit más significativo.

Esta calculadora señala la finalización del cálculo a través de la señal `w_crc_encoder_output_valid`. Al finalizar el cálculo del CRC, se lo compara con el CRC incluido en la trama. Si ambos son iguales,

mantiene en estado alto el puerto de salida `o_correct_crc` durante 3 ciclos de reloj, como se puede observar en la simulación de la Figura 3.52.

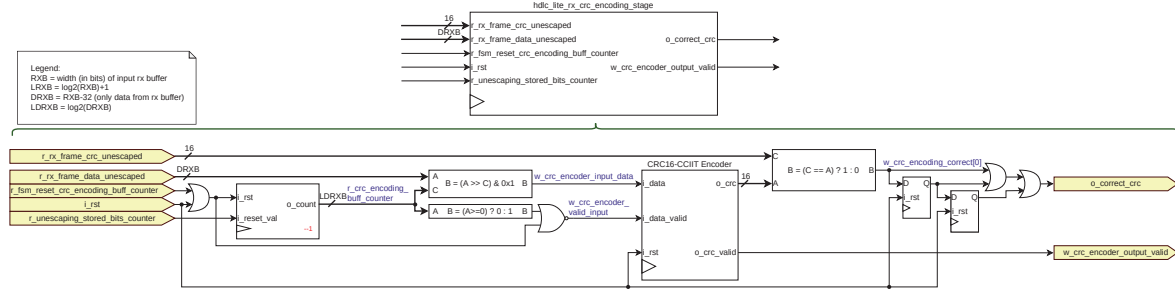


Figura 3.50: Etapa de cálculo de CRC del módulo receptor de tramas HDLC-lite.

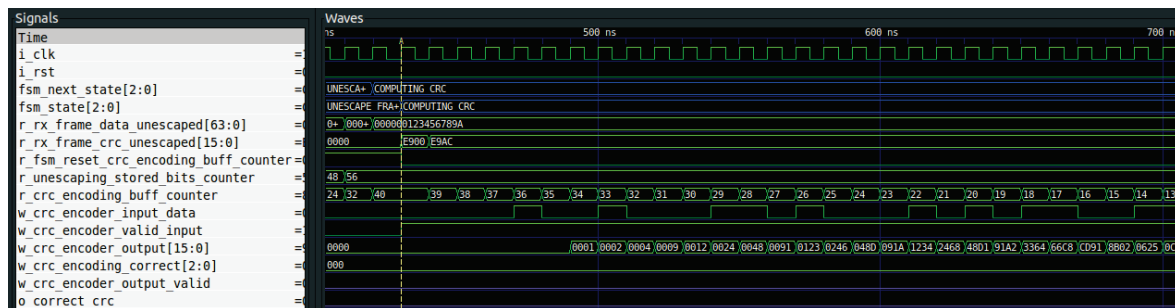


Figura 3.51: Fragmento de una simulación en el cual se visualiza la etapa de cálculo de CRC del módulo receptor de tramas HDLC-lite iniciando su rutina.

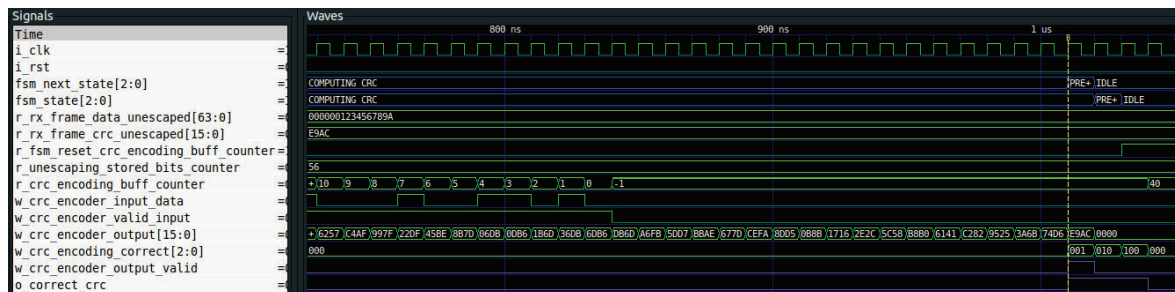


Figura 3.52: Fragmento de una simulación en el cual se visualiza la etapa de cálculo de CRC del módulo receptor de tramas HDLC-lite finalizando su rutina. En esta simulación, el CRC incluido en la trama es igual al CRC calculado.

Por último, la etapa de salida (la cual se puede observar en la Figura 3.53) expone en el *buffer* de salida `o_rx_data` los datos de la trama. También, exhibe en el puerto de salida `o_rx_dv` la cantidad de bits de información en dicho *buffer* durante 3 ciclos de reloj. Este sencilla rutina se puede observar en la simulación de la Figura 3.54.

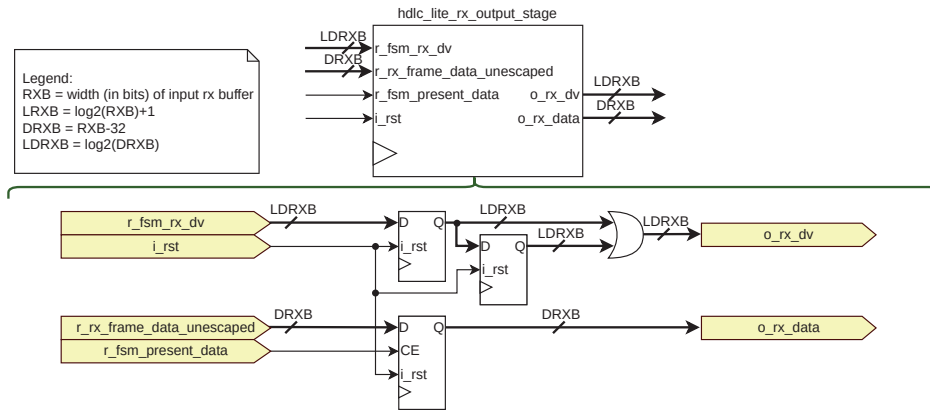


Figura 3.53: Etapa de salida del módulo receptor de tramas HDLC-lite.

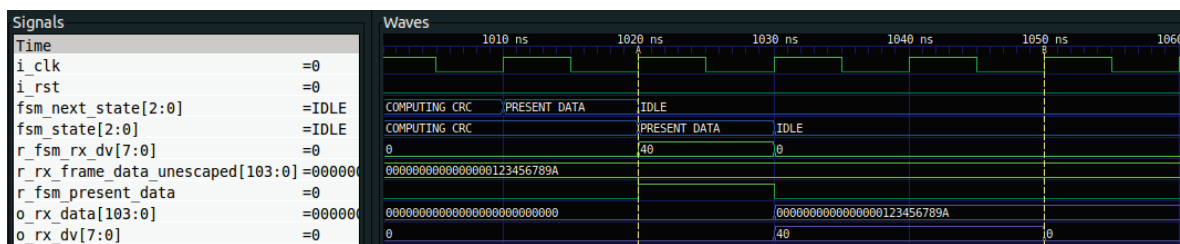


Figura 3.54: Fragmento de una simulación en el cual se visualiza la etapa de salida del módulo receptor de tramas HDLC-lite en funcionamiento.

3.4. Resumen

En este capítulo se explicó la estructura de cada uno de los canales de entrada del TDC (ver Sección 3.1). Inicialmente se presentó el bloque de *hardware* ISERDESE, que se encuentra en la FPGA seleccionada, y se explicó cómo permite tomar 4 muestras por ciclo de reloj de la señal de entrada. También, se describió el diseño de un decodificador de pulsos que permite almacenar en memoria solamente los datos necesarios para la reconstrucción de la señal de entrada y agregar información adicional a cada registro de datos mediante un *tag*. Cabe destacar que se detalló la memoria implementada en cada canal de entrada, exponiendo métricas de la utilización de los recursos de memoria propios de la FPGA por cada canal de TDC implementado.

En la Sección 3.2 se describió en detalle la lógica de procesamiento y ejecución de comandos del TDC, la cual se basa en la interacción entre un procesador y un ejecutor de comandos. Cada comando posee un ejecutor asociado, el cual realiza la rutina en respuesta al mismo. En la sección mencionada se listaron los ejecutores encontrados y se detalló su principio de funcionamiento, remarcando las acciones llevadas a cabo por el TDC al recibir estos comandos.

Al final del capítulo (ver Sección 3.3) se describió el protocolo de entramado HDLC-Lite, el cual es utilizado entre el TDC y el usuario para definir un formato para los bytes intercambiados a través del puerto serie. Dentro de esta descripción, se detalla el formato de las tramas, los caracteres de control y el CRC utilizado para detección de errores. Para implementar este protocolo dentro del TDC, se diseñó un procesador de tramas HDLC-lite, el cual consta de dos bloques: un receptor y un transmisor. El transmisor posee la responsabilidad de generar una trama a partir de un *buffer* de datos y de enviarla byte a byte. Por otro lado, el receptor se encarga de recibir una serie de bytes, detectar cuando estos conforman una trama y, si este es el caso, construir la trama para que pueda ser procesada a posteriori.

En el siguiente capítulo se detalla el diseño de un PCB el cual contiene la FPGA escogida y una gran cantidad de canales de entrada LVDS, el cual está orientado a ser utilizado en Cubesats.

4. Diseño de PCB

En este capítulo se detalla el diseño del PCB el cual contiene la FPGA deseada y 19 canales LVDS expuestos a través de un conector FMC.

Inicialmente, se encuentra un análisis del ancho de banda que debe poseer el PCB para medir las señales de alta velocidad sin dañar su integridad y luego se listan y describen los requerimientos que fueron considerados para diseñarlo. Después de listar los requerimientos, se presenta un diagrama en bloques de alto nivel del PCB y, con ayuda del esquemático, se describen brevemente cada uno de los bloques. En el resto de las secciones se describen en detalle las decisiones de diseño realizadas, otorgando fundamentos teóricos y prácticos para cada una de ellas. Por último, se expone el diseño final, acompañado de una vista detallada del mismo.

4.1. Ancho de banda del sistema

El objetivo de un TDC es medir intervalos de tiempo pequeños entre dos sucesos temporales, en el presente diseño, el objetivo es medir señales del orden de los nanosegundos cuyos tiempos de subida y bajada son de $0,6 \text{ ns}$. Este mínimo tiempo de subida se deriva del comparador LVDS LMH7220 [38] utilizado en un trabajo previo del laboratorio [13]. Es importante entender el ancho de banda que debe poseer el *hardware* para lograr este cometido, dado que para los pequeños tiempos involucrados, entran en juego consideraciones de diseño más rigurosas.

4.1.1. Ancho de banda de las señales a medir

Para determinar el ancho de banda de una señal digital, se puede empezar examinando el espectro de una señal cuadrada perfecta, la cual posee, por definición, un tiempo de subida nulo. Si la frecuencia de repetición de dicha señal es 1 GHz, entonces las frecuencias en el espectro de dicha señal serán múltiplos impares de 1 GHz [39].

El espectro de una señal representa todas las componentes senoidales que componen a la misma. Si uno quisiera reconstruir una señal en el dominio del tiempo a la perfección debería transformar cada componente armónico al dominio del tiempo y sumarlos entre sí. Pero, también es posible sumar una cierta cantidad de armónicos (no todos) y aún así obtener una representación lo suficientemente buena de la señal cuadrada. Aquí es donde entra en juego el término ancho de banda, el cual se definirá como la frecuencia del armónico de mayor grado necesario para obtener una representación lo suficientemente buena de la señal cuadrada.

Como puede observarse en la Figura 4.1, cuanto mayor sea el grado del armónico, menor será el tiempo de subida del mismo en el dominio del tiempo. Es por ello que el ancho de banda de una señal está íntimamente relacionado con el tiempo de subida de la misma. Si se eliminan componentes de alta frecuencia de la señal entonces el tiempo de subida se incrementará.

Para el caso particular de una señal cuadrada representada con un limitado número de armónicos de alta frecuencia se puede estimar su ancho de banda a partir de su mínimo tiempo de subida utilizando la Ecuación 4.1 (la cual fue extraída de [39]). Por lo tanto, siendo $0,6 \text{ ns}$ el tiempo de subida mínimo, el ancho de banda de las señales a medir resulta ser: $BW = \frac{0,35}{0,6 \text{ ns}} = 583 \text{ MHz}$. Este valor es de especial interés para el diseño de las pistas LVDS y la elección de los conectores de los canales del TDC.

$$BW = \frac{0,35}{RT} \quad (4.1)$$

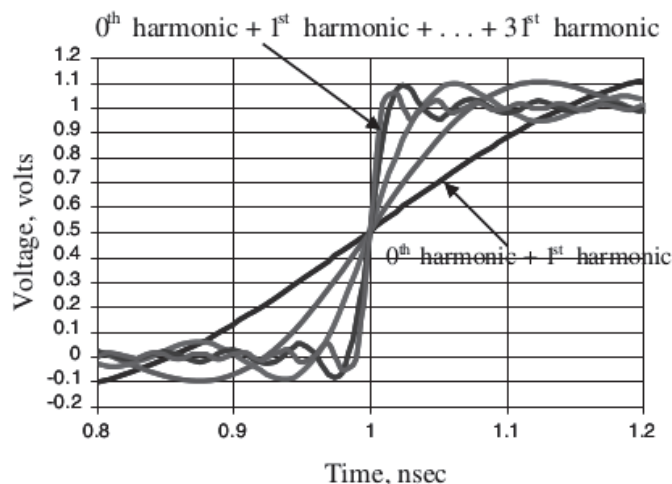


Figura 4.1: Reconstrucción de una señal cuadrada ideal de 1 GHz sumando hasta el armónico 31. Extraído de [39].

4.1.2. Ancho de banda mínimo del hardware

El *hardware* diseñado debe ser capaz de transmitir señales cuyo ancho de banda pueda llegar a ser hasta 583 MHz. El ancho de banda de una interconexión (un PCB por ejemplo) se define como la frecuencia a la cual una señal de entrada ve reducida su amplitud a un 70,7% (-3 dB) al atravesarla.

Sin embargo, es de mayor interés pensar en el tiempo de subida que poseerá una señal al atravesar la interconexión. Las interconexiones poseen un tiempo de subida mínimo que pueden transmitir, el cual se denomina **tiempo de subida intrínseco**. Si una señal con un tiempo de subida menor al tiempo de subida intrínseco de un sistema viaja a través de él, la integridad de la señal a la salida se verá perjudicada. Esto es porque la interconexión no podrá transmitir fielmente la señal.

En [39] se describe la Ecuación 4.2 con la cual se puede calcular el tiempo de subida que poseerá una señal al atravesar una interconexión, conociendo su tiempo de subida y el tiempo de subida intrínseco del sistema. Se puede visualizar que si el tiempo de subida intrínseco del sistema es mucho menor que el tiempo de subida de la señal, el tiempo de subida de la señal a la salida (RT_{salida}) será similar al que poseía a la entrada ($RT_{entrada}$).

$$RT_{salida} = \sqrt{RT_{entrada}^2 + RT_{intrinseco}^2} \quad (4.2)$$

Se puede utilizar la Ecuación 4.1 para calcular el ancho de banda del sistema para que posea un tiempo de subida intrínseco determinado. Pero la bibliografía [39] recomienda que la interconexión posea al menos el doble del ancho de banda de la señal de interés. Por lo tanto, la interconexión comprendida por los pines de la FPGA hasta el conector que expone las señales LVDS debe poseer un ancho de banda de 1,166 GHz.

4.2. Requerimientos

A partir del análisis previo, de conversaciones con los interesados (integrantes del laboratorio) y lo aprendido durante la prueba de concepto se delinearon los requerimientos para el diseño del PCB. Estos requerimientos se encuentran en la Tabla 4.1.

Tabla 4.1

Requerimientos de la placa de circuito impreso desarrollada.

ID	Requerimiento	Razón
1	La placa debe contar con una FPGA Artix Series 7 XC7A35T-2CSG324I.	Es la FPGA seleccionada por el laboratorio. Fue comprada previamente al inicio del proyecto.
2	La placa debe poseer al menos 8 entradas LVDS.	Una fortaleza de la arquitectura de TDC utilizada es su escalabilidad. Por lo tanto, se quiere contar con varios canales de entrada.
3	La placa debe poseer un puerto serie de la FPGA.	Dado que se debe comandar al TDC desde una computadora, se debe exponer una interfaz serie en la placa.
4	La placa debe poseer una memoria <i>flash</i> no volátil en la cual se pueda almacenar la configuración de la FPGA.	La FPGA utilizada posee solamente memoria volátil, por lo cual, luego de desenergizarla, su configuración se pierde. Es por esto que la placa debe contar con una memoria no volátil en la cual se pueda guardar dicha configuración.
5	La placa debe exponer la interfaz <i>Joint Test Action Group</i> (JTAG) de la FPGA.	A través de esta interfaz se configurará la FPGA a través de un programador externo.
6	La placa debe alimentarse con tensiones entre 4,5 V y 13 V.	Es deseable que durante el desarrollo la placa pueda alimentarse directamente de una computadora. Pero, como este PCB sienta las bases de un desarrollo para Cubesats, se desea que pueda alimentarse con 12 V.
7	Las entradas LVDS de la placa deberán contar con conectores cuyo ancho de banda sea mayor a 1,166 GHz.	Debido a que el ancho de banda de las señales a medir ronda los 583 MHz, los conectores deben estar preparados para no perjudicar la integridad de la señal.
8	La placa deberá contar con al menos dos leds y un botón para ser utilizado durante el desarrollo del HDL de la FPGA.	Es de utilidad contar con elementos de depuración al momento del desarrollo del HDL. Los leds se pueden utilizar para señalización y el botón puede ser utilizado para introducir en estado de <i>reset</i> al diseño configurado en la FPGA.
9	Se deberán generar e introducir dos señales de reloj distintas a la FPGA.	Una señal de reloj será utilizada para el sincronismo del TDC y la otra para el sincronismo de un generador de pulsos. Con esto se puede emular la detección de pulsos en un sistema asíncrono con un mismo PCB.
10	El PCB deberá respetar el estándar CSKB.	Dado que el PCB sienta las bases de un TDC para ser utilizado en Cubesats, debe respetar el factor de forma y las interfaces propuestas por este estándar.

4.3. Diagrama en bloques

El PCB se diseñó a partir de los requerimientos listados en la Tabla 4.1 y en la Figura 4.2 se puede visualizar un diagrama en bloques de alto nivel del mismo.

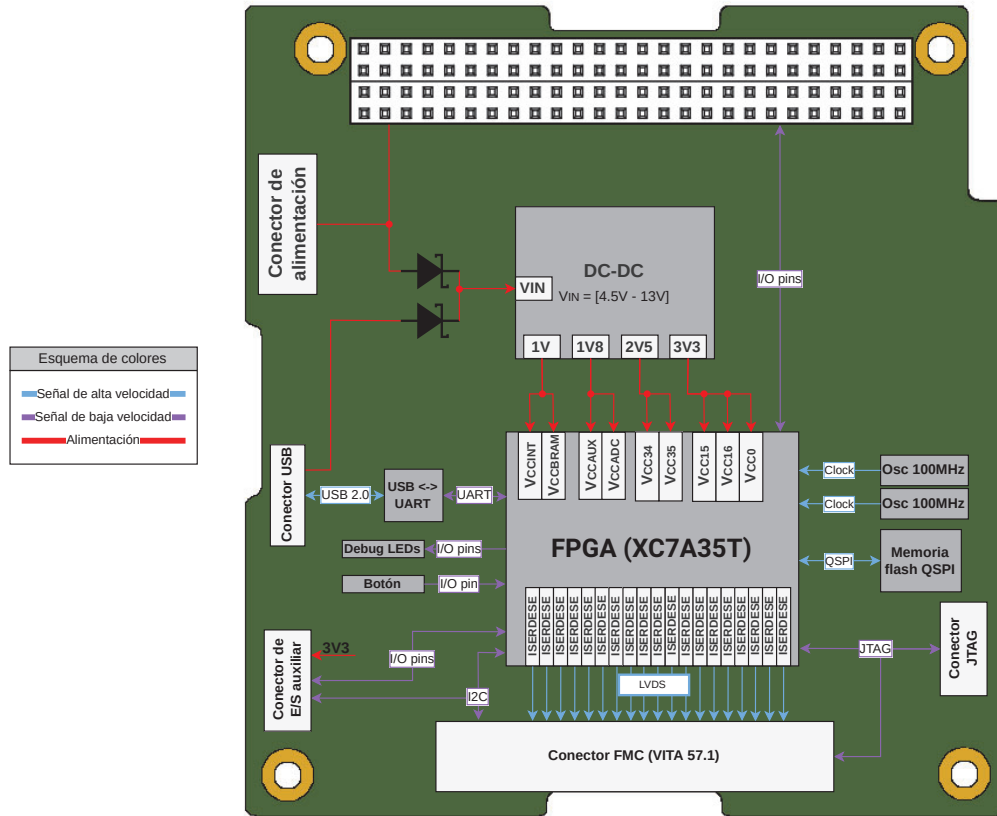


Figura 4.2: Diagrama en bloques del PCB.

En la hoja *crono_tdc.Sch.Doc* del esquemático (ver Anexo A.1) se pueden observar las interconexiones de alto nivel del *hardware*. Cada bloque encontrado en esta hoja corresponde a otra hoja del esquemático.

En el bloque **Power Input** se encuentra el conector dedicado de alimentación del PCB donde se pueden introducir tensiones entre 4,5 V y 13 V. También, se encuentra un conjunto de diodos en paralelo que permiten alimentar al PCB por el conector dedicado y al mismo tiempo utilizar el puerto USB para establecer una comunicación serie con la FPGA.

Las fuentes de alimentación del PCB, compuestas por cuatro reguladores *buck* que generan las tensiones necesarias para energizar a la FPGA, se encuentran en el bloque **Power Supply**.

Una interfaz USB C y un puente USB-UART se encuentra en el bloque **USB**. Esta interfaz UART se conecta directamente a la FPGA y los 5 V provistos por el conector USB pueden utilizarse como entrada para las fuentes *buck*.

La interfaz JTAG de la FPGA se encuentra expuesta en un conector dedicado, esto se puede observar con más detalle en el bloque **JTAG**. También, se colocó un arreglo de *jumpers* para poder configurar en modo *daisy-chain* a la FPGA del PCB y a alguna lógica programable que se encuentre conectada en el puerto FMC.

La FPGA con todos sus componentes de soporte se encuentran en el bloque **FPGA**. Dentro de este bloque se encuentran los capacitores del *Power Distribution System* (PDS) de la FPGA, las conexiones realizadas en sus bancos, la generación de las señales de reloj y la memoria *flash* para almacenar el *bitstream*.

Un conector PC/140 y las conexiones del CSKB se encuentran en el bloque **Cubesat Kit Bus**. El PCB puede ser alimentado a través de este conector (tensiones entre 4,5 V y 13 V) y, además, expone varias interfaces de Entrada/Salida (o *Input/Output* en inglés) (IO) en él.

El PCB expone varias interfaces de IO de la FPGA en un conector dedicado. Además, posee leds y botones conectados a la FPGA. Todo esto se puede encontrar en el bloque *AUX Inputs and Outputs*.

También, el PCB puede ser utilizado como una *carrier* para módulos FMC, dado que respeta (en su mayor medida) el estándar VITA 57.1. El conector FMC y las interconexiones de este estándar se encuentran en el bloque *FMC VITA 57.1*.

Por último, en el bloque *Mechanical* se encuentran los agujeros de montaje de la placa (los que respetan el estándar CSKB) y los fiduciales.

4.4. Alimentación

Las tensiones de alimentación recomendadas para una FPGA varían según su *speed grade* ¹. La FPGA utilizada (XC7A35T-2CSG324I) posee un *speed grade* -2 y las condiciones de operación recomendadas por el fabricante de la FPGA para sus alimentaciones se encuentran en la Tabla 4.2.

Tabla 4.2

Alimentaciones de la FPGA XC7A35T-2CSG324I. En blanco, las tensiones que son necesarias y, en gris, las que no lo son. Datos extraídos de la hoja de datos de la FPGA [33].

Alimentación	Descripción	Mínimo [V]	Típico [V]	Máximo [V]
V_{CCINT}	Núcleo de la FPGA	0.95	1.00	1.05
V_{CCBRAM}	Block RAM	0.95	1.00	1.05
V_{CCAUX}	Auxiliar	1.71	1.80	1.89
V_{CCADC}	Interfaz analógica configurable (XADC)	1.71	1.80	1.89
V_{CCO3V3}	Bancos de IO conectados a 3,3 V	3.135	3.3	3.465
V_{CCO2V5}	Bancos de IO conectados a 2,5 V para utilizar la interfaz LVDS_25	2.375	2.5	2.625
V_{CCBATT}	Batería	1.00	N/A	1.89
V_{REFP}	Tensión de referencia externa del XADC	1.20	1.25	1.30

Dado que en la aplicación no hay necesidad de realizar mediciones analógicas, se utilizará la tensión de referencia interna del XADC. Con esto se evita utilizar una referencia de tensión externa a la FPGA, las cuales son más precisas pero introducen un componente más en el PCB. Considerando esto no es necesario generar la tensión V_{REFP} .

Estas FPGAs poseen un mecanismo de encriptado de *bitstream* en el cual se deben almacenar llaves *Advanced Encryption Standard* (AES) en una memoria RAM interna de la FPGA la cual debe estar alimentada por una batería. En este diseño en particular, no es necesario utilizar dicho mecanismo, por lo cual no se incluye una batería (no es necesario generar la tensión V_{CCBATT}).

Como se puede visualizar en la Tabla 4.2, la FPGA necesita varios rieles de tensión para funcionar (1 V, 1,8 V, 2,5 V y 3,3 V), el diseño de la fuente de alimentación para suministrar estas tensiones se puede encontrar en el anexo B.

¹El *speed grade* determina el grado de validaciones que realizó el fabricante para garantizar que dicha FPGA puede operar a determinadas frecuencias. Las FPGAs Series 7 están disponibles en varios *speed grades*: -3, -2, -1, -1L y -2L; donde el grado -3 soporta mayores velocidades y el -1 soporta las menores.

4.5. Configuración de la FPGA

Las FPGAs son configuradas grabando una secuencia de bits llamada *bitstream* en su memoria interna. Este *bitstream* define las funciones lógicas y las conexiones entre los circuitos de la FPGA. Xilinx provee una guía de configuración [40] de sus dispositivos *Series 7* donde se describen los mecanismos por los cuales estas FPGAs pueden ser configuradas.

Los *bitstream* poseen un tamaño fijo para cada FPGA y, en particular, para la FPGA utilizada tiene un tamaño de 17536096 *bits* (aproximadamente 2,09 *MiB*). La memoria interna de esta FPGA es volátil, por lo cual, debe ser configurada siempre luego de ser energizada.

La FPGA posee diversos métodos para cargar este *bitstream* en su memoria interna, pero en el presente trabajo solamente dos de ellos generan interés:

- JTAG/*boundary-scan configuration mode*.
- *Serial Peripheral Interface (SPI) flash configuration mode (x1, x2, x4)*.

Estos métodos pueden ser elegidos a través de tres pines de la FPGA (M[2:0]). En la Tabla 4.3 se lista el valor que deben poseer estos pines para seleccionar cada modo de configuración. En la placa, se fijó el pin M0 a un estado lógico alto, el pin M1 a un estado lógico bajo y se colocó un *dip switch* que permite seleccionar el valor del pin M2. Por lo tanto, cambiando el estado de este *dip switch* se puede elegir un modo de configuración u otro.

Tabla 4.3

Selección del modo de configuración de la FPGA a partir del estado de los pines M[2:0].

Modo de configuración	M[2:0]
JTAG	101
<i>Master SPI</i>	001

Además de ejecutarse en cada reinicio, la secuencia de configuración de la FPGA puede forzarse aplicando un estado lógico bajo al pin PROGRAM_B. Cuando esto sucede, la configuración actual de la FPGA es eliminada y, dependiendo del valor de los pines M[2:0], la FPGA intentará leer un *bitstream* de la memoria *flash* (modo *Master SPI*) o esperará a ser programada a través de la interfaz JTAG (modo JTAG). En el PCB se colocó un botón para forzar el estado del pin PROGRAM_B, dejando un estado lógico alto por defecto.

El pin DONE de la FPGA es *open-drain* y es forzado a un estado lógico alto cuando se finaliza la secuencia de configuración. Se colocó un led en dicho pin el cual debe encenderse cuando termine la secuencia. El objetivo del mismo es indicar la finalización del proceso de configuración.

El pin INIT_B es forzado a un estado lógico bajo por la FPGA cuando:

- Está en un estado de reinicio de su configuración.
- Está borrando el contenido de la memoria donde almacena el *bitstream*.
- Detecta un error en la configuración (en la transferencia del *bitstream*).

Luego del proceso de inicialización, la FPGA espera hasta detectar un estado lógico alto en este pin para continuar con el resto de la secuencia de configuración definida por el estado de los pines M[2:0]. Es por eso que en algunos diseños se utiliza este pin para retrasar la secuencia de configuración de la FPGA. En este caso se fijó a un estado lógico alto con un *pull up*, dado que no es necesario retrasarla.

La mayoría de los pines de configuración se encuentran en el banco 0 y algunos específicos se encuentran en los bancos 14 y 15. En el caso de alimentar a los bancos 14 y 15 con una tensión de alimentación de 2,5 V o 3,3 V el pin de selección de voltaje de bancos de configuración (CFGBVS) debe estar atado a un estado lógico alto, en caso contrario, debe estar conectado a GND. En este caso, se conectó CFGBVS a 3,3 V.

Cabe destacar que cuando se utiliza el banco 14 o el banco 15 durante la configuración de la FPGA, las alimentaciones de estos bancos debe ser la misma que la del banco 0. En este caso, como

la interfaz SPI que se utiliza para leer el *bitstream* de la memoria *flash* está ubicada en el banco 14, se alimentó dicho banco con 3,3 V.

Los pines VREF_P y VREF_N se utilizan para entregar una referencia de tensión al Conversor Analógico Digital (o *Analog to Digital Converter* en inglés) (ADC) de la FPGA. El XADC de la FPGA también posee una referencia de tensión dentro del mismo chip, la cual se selecciona conectando los pines VREF_P y VREF_N a tierra. La referencia de tensión interna del chip no es tan precisa como la que se podría entregar externamente con algún integrado dedicado a ello. Sin embargo, como esta aplicación no está enfocada a utilizar el ADC de la FPGA, se utilizará la referencia interna por simplicidad.

En la hoja *bank0_config.SchDoc* (ver Anexo A.1) se visualiza el esquemático final de la configuración del banco 0 y los dispositivos relacionados a la configuración de la FPGA.

4.5.1. Modo de configuración JTAG

La familia de FPGAs *Series 7* de Xilinx soporta el estándar IEEE 1149.1, definiendo un *Test Access Port* (TAP) y una arquitectura de *Boundary Scan*. Esta arquitectura permite cargar una configuración directamente a la FPGA y es comúnmente llamada JTAG.

Las herramientas de Xilinx permiten configurar estas FPGAs interactuando directamente con el controlador TAP de la interfaz. Para esto, se necesita un programador de Xilinx que conecte esta interfaz JTAG a una PC.

El cable JTAG HS3 [41] que provee Xilinx posee un programador. Este cable está terminado con un conector Molex *Milli-Grid* hembra, por lo cual, se usó el conector Molex *Milli-Grid* macho apareado (*Part Number*: 878321420) para exponer la interfaz JTAG en el PCB, respetando el *pinout* del cable de Xilinx (ver Figura 4.3). Este conector es el mismo que utiliza la placa de desarrollo ZCU102 para el mismo fin [42].

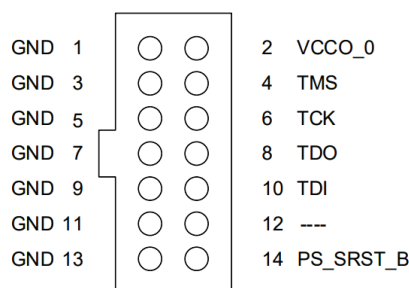


Figura 4.3: *Pinout* del cable JTAG HS3. Extraído de [41].

También, dado que la interfaz JTAG está conectada a un conector FMC, se dejó la posibilidad de configurar con unos resistores de 0 Ω el JTAG en modo *daisy-chain*. El conector JTAG y los resistores de 0 Ω se pueden encontrar en la hoja *jtag.SchDoc* del esquemático (ver Anexo A.1).

4.5.2. Modo de configuración Master SPI

En este método, el *bitstream* se encuentra almacenado dentro de una memoria no volátil externa, la FPGA lo lee a través de una interfaz SPI y lo graba en su memoria interna.

El software de Xilinx permite programar la memoria *flash* SPI de manera indirecta a través de la FPGA utilizando la interfaz JTAG. Esto sucede en un proceso de dos etapas, primero la FPGA es configurada con un circuito que puede interactuar con la memoria *flash* y luego el *bitstream* es transferido a la memoria *flash* a través del circuito implementado en la FPGA. Todo este proceso lo realiza la herramienta de *software* Vivado de Xilinx, por lo cual, el usuario se abstrae del mismo. Luego de que el *bitstream* sea almacenado en la memoria *flash*, la próxima vez que la secuencia de

configuración sea iniciada, la FPGA leerá dicho *bitstream* de la memoria *flash* si los pines M[2:0] de la misma tienen el valor 001 (ver Tabla 4.3).

En la Figura 4.4 se puede visualizar el esquema de conexión especificado por el fabricante para poder descargar un *bitstream* a una memoria SPI conectada a la FPGA.

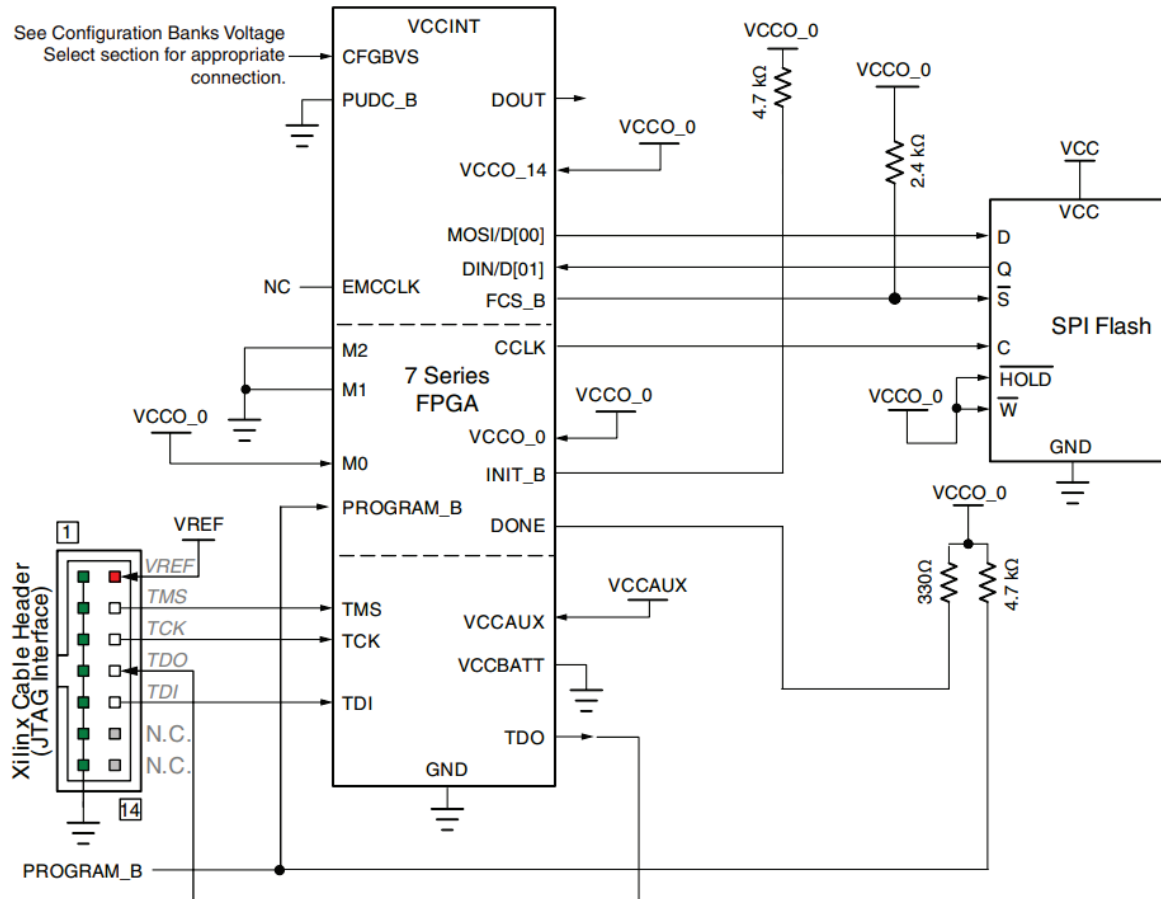


Figura 4.4: Esquema circuital necesario para descargar un *bitstream* a una memoria *flash* conectada por SPI a una FPGA *Series 7*. Extraído de la guía de configuración [40].

En la guía de usuario de configuración [40] se recomienda utilizar una memoria *flash* de al menos 32 Mb para almacenar dicho *bitstream*. Por otro lado, en la guía de programación y depuración [43] se encuentran las memorias *flash* soportadas por la herramienta Vivado. A partir de la información encontrada en dichas guías, se escogió la memoria S25FL064LABMFI013 [44] de la familia S25F-L.

En la placa de desarrollo Arty A7 [45] se utiliza la memoria S25FL032P, la cual es de una familia predecesora de la familia S25F-L, y en la placa Basys 3 [46] se utiliza una memoria S25FL128S, la cual posee un conjunto de comandos compatible con la familia S25FL-P. Cabe destacar que la Basys 3 utiliza una FPGA ($bitstream_{XC7A100T} = 30,606 \text{ Mb}$) cuyo *bitstream* es mayor al de la FPGA utilizada ($bitstream_{XC7A35T} = 17,536 \text{ Mb}$), es por ello que la memoria utilizada en dicha placa posee un tamaño mayor.

Las señales involucradas en la lectura y escritura de la memoria *flash* se describen en la Tabla 4.4, dichas señales son las mismas que se utilizan en la placa Arty A7 [45].

Tabla 4.4

Pines de la FPGA involucrados en la lectura y escritura de la memoria flash.

Nombre de Pin	Banco	Descripción
CCLK_0	0	Señal de reloj generada por la FPGA dado que se cumple el rol de maestro en la comunicación SPI con la memoria.
IO_L1P_T0_D00_MOSI_14	14	Señal de datos D00 de la comunicación <i>Quad Serial Peripheral Interface</i> (QSPI) con la memoria.
IO_L1N_T0_D01_DIN_14	14	Señal de datos D01 de la comunicación QSPI con la memoria.
IO_L2P_T0_D02_14	14	Señal de datos D02 de la comunicación QSPI con la memoria.
IO_L2N_T0_D03_14	14	Señal de datos D03 de la comunicación QSPI con la memoria.
IO_L6P_T0_FCS_B_14	14	Señal de selección de esclavo (<i>chip select</i>) en la comunicación QSPI con la memoria.

4.6. Puente UART-USB

Se incluyó un puente USB-UART para poder comunicarse con la FPGA directamente desde una computadora sin necesidad de incorporar un adaptador intermedio.

Se escogió el integrado FT231XS-U2 [47] para implementar este puente. Dicho chip permite conectar una interfaz UART con una interfaz USB, alimentándose a través de la misma interfaz USB. El chip posee un oscilador interno, por lo tanto no requiere conectar ningún cristal externo. Además soporta formatos de tramas y velocidades en su interfaz UART (las mismas son configuradas a través del *driver* utilizado en la PC).

Este chip incluye reguladores de tensión de 3,3 V y 1,8 V internos, por lo que se lo puede alimentar desde los 5 V entregados por una computadora a través de la interfaz USB. La salida del regulador interno de 3,3 V está expuesta en el pin 3V3OUT y las entradas de alimentación del integrado son las siguientes:

- VCC: es la entrada de alimentación de los reguladores internos de 3,3 V y de 1,8 V. Con estos reguladores el chip alimenta a sus circuitos internos.
- VCCIO: es la entrada de alimentación de las interfaces de IO del integrado. Se puede alimentar directamente con el regulador interno de 3,3 V a través del pin 3V3OUT.

En la nota de aplicación [48] se indica que se deben colocar capacitores en las líneas de datos de la interfaz USB para ajustar el tiempo de subida y bajada de las señales que viajan a través de ellas. Por otro lado, se indica que se deben colocar resistores de 27 Ω en serie en las mismas, dado que el chip posee una impedancia de entrada pequeña en los pines de la interfaz USB (aproximadamente 36 Ω). Como la impedancia de un cable USB es 90 Ω , a altas velocidades pueden ocurrir reflexiones en la línea de transmisión que pueden perjudicar la comunicación. Con estos resistores, la impedancia que ve el cable es $27 \Omega + 27 \Omega + 36 \Omega = 90 \Omega$ y, por lo tanto, el problema de las reflexiones se vería mitigado.

El integrado también posee los pines CBUS1, CBUS2, CBUS3 y CBUS4, los cuales se pueden configurar para implementar diversas funcionalidades. Entre estas funcionalidades, estos pines pueden cambiar su estado lógico cuando se recibe o se envían datos por la interfaz USB. Estas configuraciones se pueden realizar con una herramienta de *software*, pero, por defecto el pin CBUS1 está configurado para pulsar cuando se reciben datos y el pin CBUS2 cuando se envían datos. Se colocaron leds en estos pines para señalar cuando hay tráfico en la interfaz USB.

Si bien la especificación USB C es extensa, para esta aplicación en particular solo es necesario centrarse en algunas señales de la misma:

- D_P y D_N: son los pares diferenciales de datos de la interfaz USB 2.0. Los conectores USB C poseen dos pares diferenciales para proveer reversibilidad. En el PCB, el negativo de un par debe ir conectado con el negativo del otro par. Análogamente, se debe realizar lo mismo con el positivo de cada par.
- CC1 y CC2 (*Channel Configuration pins*): estas señales sirven para identificar quien es el extremo que provee la alimentación en la interfaz. El estándar requiere que estas señales posean un *pull down* de $5,1\text{ k}\Omega$ en el extremo que es alimentado (el *sink*). Luego, el extremo que provee la alimentación (el *source*), tendrá un *pull up* en cada una de estas señales. Con este valor de *pull up* el *host* determina cuánta corriente puede suministrar.
- VBUS: es la alimentación, la cual debe ser nominalmente 5 V . En este estándar, hasta que el *source* no detecta que haya un dispositivo conectado a través de las señales CC1 y CC2, no provee ninguna alimentación.

En las interfaces USB puede ocurrir alguna Descarga Electroestática (o *Electrostatic Discharge* en inglés) (ESD) durante las conexiones y desconexiones. Estas ESD pueden dañar circuitos que estén en el PCB (como por ejemplo, la FPGA). Por lo tanto, se colocó un diodo Supresor de Voltaje Transitorio (o *Transient Voltage Suppressor* en inglés) (TVS) en cada línea de la interfaz USB. En condiciones nominales, los diodos TVS no conducen y no afectan al circuito de ninguna manera. Pero, cuando se presenta un evento de alta tensión, el cual puede ser producido por una ESD, el diodo conduce corriente fijando una caída de tensión (llamada voltaje de *clamping*) entre sus terminales. De esta manera, el TVS protege los circuitos del PCB de cualquier evento de ESD que se pueda provocar en el puerto USB.

El diseño de esta interfaz se basó en el adaptador USB a puerto serie *FT231x Breakout Board* [49] y en el módulo LC231x [50]. En la hoja *usb.SchDoc* del esquemático (ver Anexo A.1) se puede visualizar el circuito implementado.

4.7. Canales del TDC

Por requerimiento, la interfaz de los canales del TDC debe ser LVDS. Según el estándar TIA/EIA-644 se especifica una velocidad teórica máxima de 1.923 Gbps para estas interfaces (ver nota de aplicación [51]). Pero, en la práctica, la tasa máxima de transmisión de datos está limitada por la calidad del medio de enlace entre el emisor y el receptor.

Para introducir este tipo de interfaces, se toma como ejemplo el comparador LVDS de alta velocidad LMH7220 [38], el cual fue utilizado en un trabajo previo del grupo de investigación [13]. Este comparador entrega una corriente de $3,25\text{ mA}$ la cual produce una caída de tensión de 325 mV sobre una resistencia de terminación de $100\ \Omega$. A nivel conceptual, la interfaz se puede visualizar en la Figura 4.5.

El estándar TIA/EIA-644 especifica el uso de líneas de transmisión cuya impedancia característica está entre $90\ \Omega$ y $132\ \Omega$, por lo tanto, la resistencia terminadora ubicada en el receptor cumple otro rol además de generar una caída de tensión determinada. Como el medio de enlace posee una impedancia característica de $100\ \Omega$ si la impedancia del receptor no es de $100\ \Omega$ entonces algunas componentes de alta frecuencia de la señal que viajan por el medio pueden reflejarse y perjudicar la integridad de la señal.

Dado que la interfaz LVDS es un esquema diferencial, cada canal del TDC involucrará dos conductores (uno transportando la señal positiva y el otro la negativa). Si se utilizan conectores coaxiales de alta frecuencia (SMA, MCX, LEMO, etc) se deben colocar dos por cada canal porque son cables *single ended*. Con esto en mente, se debe recordar que el objetivo de esta arquitectura de TDC es escalar en cantidad de canales y colocar dos conectores coaxiales por cada uno de ellos resulta impráctico (en términos de espacio en el PCB y en costos). Es por ello que es preferible un conector de alta velocidad el cual integre varios pares diferenciales.

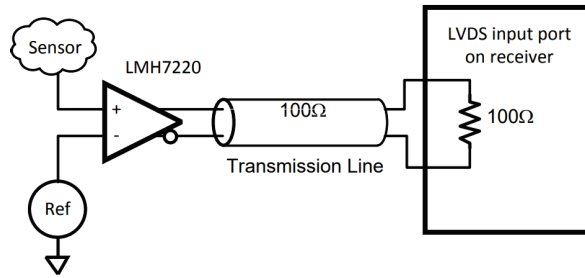


Figura 4.5: Comparador LMH7220 como *driver* de una interfaz LVDS. Extraído de la hoja de datos [38].

4.7.1. Estándar VITA 57.1

Se estudió la especificación VITA 57.1 [52], la cual define los lineamientos para la serie de productos FMC. En esta especificación se describe una interfaz electromecánica entre placas con una o más FPGAs (*carriers boards*) y placas que poseen interfaces de IO de interés (*mezzanine modules*).

En esta aplicación, el PCB diseñado es un *carrier* que implementa un TDC, donde las señales de entrada ingresan a través del conector FMC. El módulo *mezzanine* es quien contendrá el o los sensores con su o sus correspondientes AFE.

El estándar define dos tipos de conectores, un conector *High Pin Count* (HPC) con 400 contactos y otro conector *Low Pin Count* (LPC) con 160 contactos. En el conector LPC se pueden conectar hasta 34 pares diferenciales y en el conector HPC hasta 80. Dado que 34 pares diferenciales es mayor al número dictado por el requerimiento, se optó por el conector LPC por simplicidad y porque es más económico. La asignación de pines de este conector, definida por el estándar, se puede observar en la Figura 4.6.

	K	J	H	G	F	E	D	C	B	A
1	NC	NC	VREF_A_M2C	GND	NC	NC	PG_C2M	GND	NC	NC
2	NC	NC	PRSN_T_M2C_L	CLK0_C2M_P	NC	NC	GND	DP0_C2M_P	NC	NC
3	NC	NC	GND	CLK0_C2M_N	NC	NC	GND	DP0_C2M_N	NC	NC
4	NC	NC	CLK0_M2C_P	GND	NC	NC	GBTCLK0_M2C_P	GND	NC	NC
5	NC	NC	CLK0_M2C_N	GND	NC	NC	GBTCLK0_M2C_N	GND	NC	NC
6	NC	NC	GND	LA00_P_CC	NC	NC	GND	DP0_M2C_P	NC	NC
7	NC	NC	LA02_P	LA00_N_CC	NC	NC	GND	DP0_M2C_N	NC	NC
8	NC	NC	LA02_N	GND	NC	NC	LA01_P_CC	GND	NC	NC
9	NC	NC	GND	LA03_P	NC	NC	LA01_N_CC	GND	NC	NC
10	NC	NC	LA04_P	LA03_N	NC	NC	GND	LA06_P	NC	NC
11	NC	NC	LA04_N	GND	NC	NC	LA05_P	LA06_N	NC	NC
12	NC	NC	GND	LA08_P	NC	NC	LA05_N	GND	NC	NC
13	NC	NC	LA07_P	LA08_N	NC	NC	GND	GND	NC	NC
14	NC	NC	LA07_N	GND	NC	NC	LA09_P	LA10_P	NC	NC
15	NC	NC	GND	LA12_P	NC	NC	LA09_N	LA10_N	NC	NC
16	NC	NC	LA11_P	LA12_N	NC	NC	GND	GND	NC	NC
17	NC	NC	LA11_N	GND	NC	NC	LA13_P	GND	NC	NC
18	NC	NC	GND	LA16_P	NC	NC	LA13_N	LA14_P	NC	NC
19	NC	NC	LA15_P	LA16_N	NC	NC	GND	LA14_N	NC	NC
20	NC	NC	LA15_N	GND	NC	NC	LA17_P_CC	GND	NC	NC
21	NC	NC	GND	LA20_P	NC	NC	LA17_N_CC	GND	NC	NC
22	NC	NC	LA19_P	LA20_N	NC	NC	GND	LA18_P_CC	NC	NC
23	NC	NC	LA19_N	GND	NC	NC	LA23_P	LA18_N_CC	NC	NC
24	NC	NC	GND	LA22_P	NC	NC	LA23_N	GND	NC	NC
25	NC	NC	LA21_P	LA22_N	NC	NC	GND	GND	NC	NC
26	NC	NC	LA21_N	GND	NC	NC	LA26_P	LA27_P	NC	NC
27	NC	NC	GND	LA25_P	NC	NC	LA26_N	LA27_N	NC	NC
28	NC	NC	LA24_P	LA25_N	NC	NC	GND	GND	NC	NC
29	NC	NC	LA24_N	GND	NC	NC	TCK	GND	NC	NC
30	NC	NC	GND	LA29_P	NC	NC	TDI	SCL	NC	NC
31	NC	NC	LA28_P	LA29_N	NC	NC	TDO	SDA	NC	NC
32	NC	NC	LA28_N	GND	NC	NC	3P3VAUX	GND	NC	NC
33	NC	NC	GND	LA31_P	NC	NC	GND	GND	NC	NC
34	NC	NC	LA30_P	LA31_N	NC	NC	TRST_L	GA0	NC	NC
35	NC	NC	LA30_N	GND	NC	NC	GA1	12P0V	NC	NC
36	NC	NC	GND	LA33_P	NC	NC	3P3V	GND	NC	NC
37	NC	NC	LA32_P	LA33_N	NC	NC	GND	12P0V	NC	NC
38	NC	NC	LA32_N	GND	NC	NC	3P3V	GND	NC	NC
39	NC	NC	GND	VADJ	NC	NC	GND	3P3V	NC	NC
40	NC	NC	VADJ	GND	NC	NC	3P3V	GND	NC	NC

Figura 4.6: Asignación de pines del conector LPC definido en la especificación VITA 57.1. Extraído de la especificación VITA 57.1 [52].

La especificación requiere que las señales sean asignadas a los pines en orden ascendente, es por ello que los canales del TDC se asignaron respetando la numeración de los pares diferenciales del

estándar. Es decir, el canal 0 del TDC se conectó al par diferencial LA00, el canal 1 al par diferencial LA01, el canal 2 al par diferencial LA02 y así sucesivamente. También se conectaron dos canales a los pines DP0.C2M y DP0.M2C para aprovechar el uso de módulos comerciales los cuales exponen estas señales en conectores SMA o MCX.

Dado que ya se está incluyendo un conector para módulos FMC, se realizaron conexiones adicionales para cumplir con el estándar. De esta manera, el PCB puede ser utilizado para conectar otros módulos FMC comerciales. El detalle de estas conexiones se puede encontrar en el Anexo D.

4.7.2. Conector FMC

Se escogió el conector de alta velocidad ASP-134603-01 de Samtec. Si bien el fabricante no provee resultados de ensayos de alta velocidad de este componente, como estos conectores están basados en la familia de conectores SEAF, los resultados de ensayos sobre estos últimos [53] se pueden extrapolar al conector ASP-134603-01 (información brindada por el fabricante). En estos reportes de ensayos hay tres características que son de especial interés: la **impedancia diferencial**, la **pérdida de inserción** y la **pérdida de retorno**.

La **impedancia diferencial** debe ser la adecuada para la interfaz utilizada. En el caso del presente diseño, la impedancia diferencial del conector debe ser $100 \Omega \pm 5\%$ dado que se utilizan interfaces LVDS. En la Figura 4.7 se visualiza la impedancia diferencial en función del tiempo de subida. Para las señales cuyo tiempo de subida es $0,5 \text{ ns}$, la impedancia se encuentra entre 100Ω y $104,1 \Omega$ [53]. Por lo tanto, como las señales a medir poseen un tiempo de subida ligeramente mayor ($0,6 \text{ ns}$), la impedancia diferencial de este conector es la correcta.

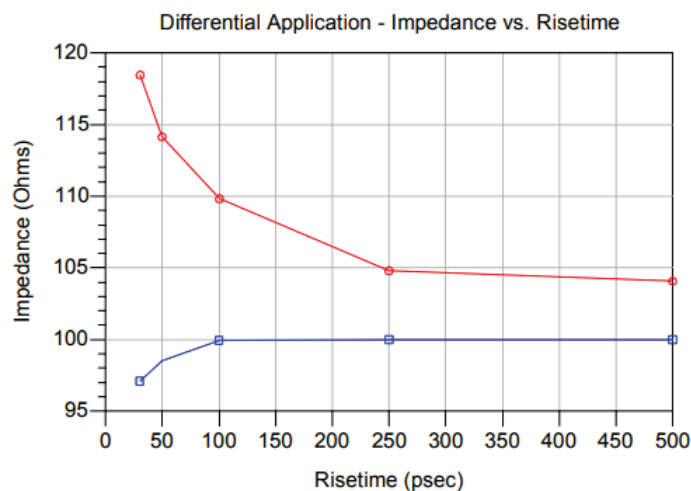


Figura 4.7: Impedancia diferencial de los conectores SEAM/SEAF de Samtec. En rojo su valor máximo y en azul su valor mínimo. Extraído del reporte de ensayos de alta velocidad [53].

La **pérdida de inserción** describe la cantidad de energía que una señal pierde al atravesar un conector. La misma se define según la Ecuación 4.3.

$$IL_{dB} = 10 \log\left(\frac{P_{entrada}}{P_{salida}}\right) \quad (4.3)$$

La **pérdida de retorno** describe la cantidad de energía de la señal que se refleja al atravesar el conector y se define según la Ecuación 4.4.

$$RL_{dB} = 10 \log\left(\frac{P_{entrada}}{P_{reflejada}}\right) \quad (4.4)$$

En el reporte de los ensayos de alta velocidad [53], el fabricante expone los parámetros S_{21} y S_{11} del conector, los cuales están relacionados con los valores mencionados previamente. La definición de los parámetros S de una determinada red se puede visualizar en la Figura 4.8.

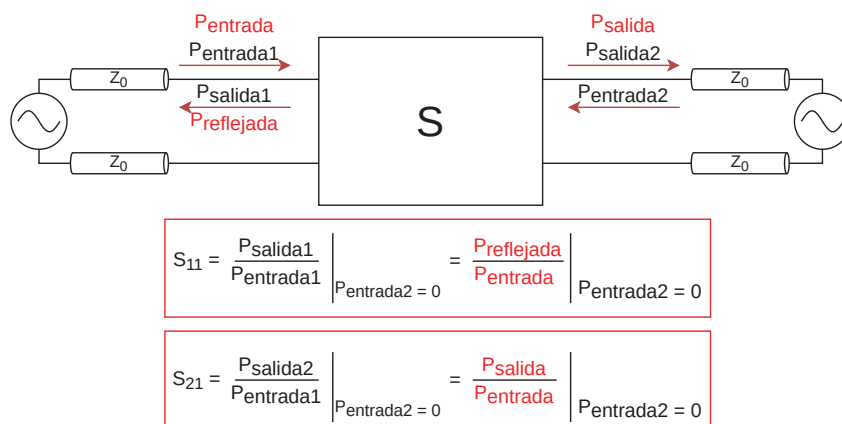


Figura 4.8: Definición de los parámetros S_{11} y S_{21} de una interconexión.

El parámetro S_{21} está íntimamente relacionado con la pérdida de inserción y se puede calcular con la Ecuación 4.5. Este parámetro tendrá siempre un valor negativo porque la potencia de salida siempre será menor que la potencia de entrada (siempre va a existir pérdida al atravesar el sistema).

En la Figura 4.9a se observa que, dentro del ancho de banda de interés ($1,166 \text{ GHz}$), el peor valor de S_{21} es cercano a $-0,1 \text{ dB}$. Utilizando la Ecuación 4.5, se obtiene que $P_{salida} = 0,977 P_{entrada}$, con lo que se puede estimar que en el peor de los casos la potencia de salida se atenuará aproximadamente un 2,3% respecto a la potencia de entrada.

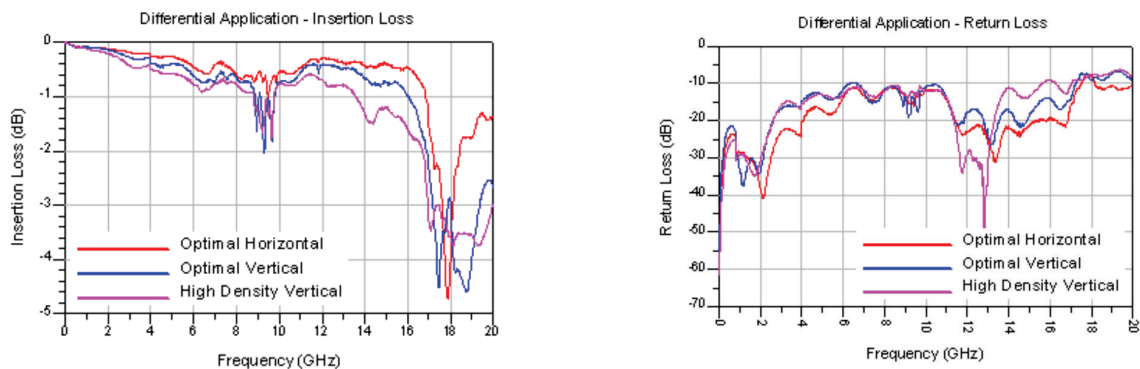
$$S_{21_{dB}} = 10 \log \left(\frac{P_{salida}}{P_{entrada}} \right) \quad (4.5)$$

Por otro lado, el parámetro S_{11} está íntimamente relacionado con la pérdida de retorno y se calcula con la Ecuación 4.6. Este parámetro también será un valor negativo, dado que la potencia reflejada siempre será menor que la potencia de entrada.

En la Figura 4.9b se observa que el parámetro S_{11} posee un máximo de -22 dB dentro del ancho de banda del sistema ($1,166 \text{ GHz}$). Utilizando la Ecuación 4.6, obtenemos que $P_{reflejada} = 0,0063 P_{entrada}$. Por lo tanto, en el peor de los casos un 0,63% de la potencia de entrada se reflejará.

$$S_{11_{dB}} = 10 \log \left(\frac{P_{reflejada}}{P_{entrada}} \right) \quad (4.6)$$

Visto esto, las pérdidas de inserción y de retorno del conector son pequeñas dentro del ancho de banda del sistema, por lo que se lo encuentra adecuado para esta aplicación. En las hojas *fmc_lpc_d.c.SchDoc* y *fmc_lpc_h.g.SchDoc* del esquemático (ver Anexo A.1) se puede visualizar el conector escogido, los canales del TDC conectados al mismo y las señales correspondientes del estándar VITA 57.1.



(a) Parámetro S_{21} (pérdida de inserción) de los conectores SEAM/SEAF de Samtec.

(b) Parámetro S_{11} (pérdida de retorno) de los conectores SEAM/SEAF de Samtec.

Figura 4.9: Pérdida de retorno y de inserción de los conectores SEAM/SEAF de Samtec. Ambos gráficos fueron extraídos de [53].

4.7.3. Banco 34 y 35

Para implementar un canal del TDC se necesitan dos pares diferenciales de la FPGA y un bloque ISERDESE. En la hoja de datos de la FPGA [54] se especifica que cada pin de IO posee un bloque ISERDESE, por lo cual, esto no es una limitación. Por otro lado, en la guía de usuario [55] se menciona que la mayoría de los pines de IO pueden utilizarse para implementar pares diferenciales. En particular, si un pin se nombra IO_LXX entonces significa que es parte del par diferencial XX.

La única interfaz LVDS disponible en la FPGA se denomina *LVDS_25 I/O*. Esta interfaz requiere que el banco donde se implemente esté alimentado con 2,5 V si se desean utilizar dichas interfaces como salida o si se desea utilizar una resistencia terminadora interna de 100 Ω . Al hacer uso de esta resistencia terminadora interna se simplifica el diseño y el *ruteo* mismo de las señales, dado que sino, se debería agregar 1 resistor al PCB por cada canal de TDC. Es por ello que se alimentaron los bancos 34 y 35 con 2,5 V (ver hoja *fpga_power.SchDoc* en Anexo A.1).

En las hojas *bank34.SchDoc* y *bank35.SchDoc* del esquemático (ver Anexo A.1) se pueden visualizar los bancos 34 y 35 de la FPGA. También, en la hoja *tdc_channels_assignment.SchDoc* del esquemático se puede visualizar la asignación de señales de la FPGA al conector FMC.

4.8. Señales de reloj externas

La placa de desarrollo utilizada en la prueba de concepto [45] posee una señal de reloj base de 100 MHz. Dicha señal de reloj es generada por el oscilador externo ASEM1-100.000MHZ-LC-T [56]. Se escogió el oscilador SIT2001BI-S2-33S-100.000000 [57], el cual es de la misma tecnología que el utilizado en la Arty A7 (tecnología de Sistemas Microelectromecánicos (MEMS)).

La FPGA utilizada está dividida internamente en regiones de reloj, las cuales incluyen todos los elementos síncronos que están ubicados dentro de un mismo banco. Las señales de reloj deben ir conectadas a aquellos pines de la FPGA que están preparados para introducir una señal de reloj dentro del chip. Estos pines, denominados *Multi-Region Clock Capable I/O* (MRCC) y *Single-Region Clock Capable I/O* (SRCC), poseen un *ruteo* dedicado dentro de la FPGA para acceder rápidamente a los recursos de reloj globales y regionales de la misma. La diferencia entre ellos es que los pines MRCC tienen un mayor acceso a otras regiones de reloj en comparación a los pines SRCC. Se terminaron conectando las señales de reloj a los pines MRCC, dado que otorgan una mayor flexibilidad al alcanzar más regiones de la FPGA.

Se colocaron dos fuentes de reloj distintas en el PCB, dado que, con este arreglo se puede utilizar una señal de reloj para generar pulsos y la otra para sincronizar al TDC. Esto es de suma utilidad para continuar el desarrollo del HDL del TDC, dado que permitiría realizar detecciones de

señales asíncronas utilizando una sola placa. No habría necesidad de generar los pulsos con otra placa, como se hizo en la prueba de concepto.

4.9. Especificación CubeSat Kit PCB

En la industria espacial, y más específicamente en el segmento de Cubesats, la interconexión de las distintas placas electrónicas que los componen se basa en la especificación PC/104-Plus [58]. Dado que se está desarrollando un prototipo de TDC que puede ser utilizado en un Cubesat, se decidió tomar como base la especificación *CubeSat Kit PCB* [59], la cual está basada en la especificación PC/104-Plus. Esta especificación define un factor de forma y una interfaz eléctrica determinada por dos conectores de dos filas de 26 pines.

En la Figura 4.10 se visualiza este tipo de interconexión, donde las placas que componen al Cubesat se apilan una encima de la otra. Este tipo de arquitectura disminuye el cableado necesario entre los distintos módulos del satélite. También, permite utilizar componentes *Commercial Off-The-Shelf* (COTS) que respeten la especificación en la construcción del satélite.

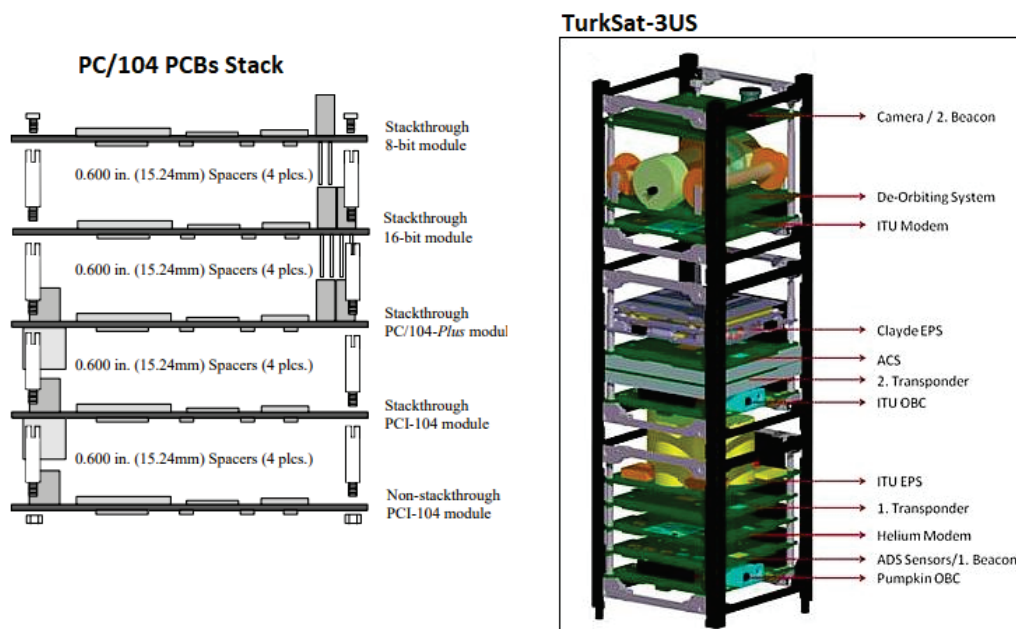


Figura 4.10: Izquierda: *stack* de placas electrónicas respetando las especificaciones PC/104 [58]. Derecha: Cubesat TurkSat-3USat [60] donde se puede visualizar la interconexión entre las placas.

En cuanto a la asignación de señales en los pines de los conectores del CSKB se tomó de referencia la *CubeSat Kit Motherboard (MB)* diseñada por la empresa Pumpkin [61]. De esta manera, se podría utilizar esta *On Board Computer (OBC)* para comandar al TDC.

La asignación final de pines en el conector del PCB se encuentra en la hoja *cubesat_kit_bus.SchDoc* del esquemático (ver Anexo A.1). Con este conexionado, se puede alimentar a la placa desde el conector CSKB y también se exponen varias señales de IO de la FPGA para otorgar mayor flexibilidad en futuros proyectos.

4.10. Stackup

Para minimizar costos, se diseñó el PCB utilizando el *stackup* estándar para placas de 8 capas provisto por PCBWay. El mismo se puede visualizar en las instrucciones de fabricación encontradas en el Anexo A.2.

El PCB posee 2 capas dedicadas para *rutear* señales (*L1-SIG TOP* y *L8-SIG BOT*) y 2 capas para *rutear*, principalmente, los polígonos de alimentación (*L3-PWR/SIG* y *L5-PWR/SIG*). Estas dos últimas capas de alimentación también se utilizaron para *rutear* algunas señales, dado que hay zonas densas en el PCB, como por ejemplo, la conexión de los pares diferenciales LVDS entre la FPGA y el conector FMC. El resto de las capas están dedicadas a GND para que cada una de las otras capas posea un plano de referencia cercano.

Se puede visualizar en las instrucciones de fabricación que el acabado superficial de la placa debe ser *Electroless Nickel Immersion Gold* (ENIG) y no el acabado *Hot Air Solder Leveling* (HASL), el cual es utilizado comúnmente. Una diferencia fundamental entre estos dos tipos de acabados es que el ENIG logra una gran planitud sobre el PCB. Por otro lado, el HASL provoca que la superficie del PCB no esté totalmente nivelada, lo que trae problemas al montar componentes con un *pitch* muy pequeño. En este caso, como se está montando un componente *Ball Grid Array* (BGA) con una gran cantidad de pines y un *pitch* de 0,8 mm, se decidió utilizar el acabado ENIG.

4.11. Resultado final del diseño

En la Figura 4.11 se puede ver una vista en detalle del PCB donde se encuentran las dimensiones y ciertas funcionalidades del mismo. En el mismo, se puede destacar que se respetó el factor de forma de la especificación *CubeSat Kit PCB* y se colocaron los agujeros de montaje indicados por el estándar VITA 57.1. Todos los archivos de manufactura necesarios para fabricar el diseño, junto con el proyecto de Altium, se pueden encontrar en el repositorio [24].

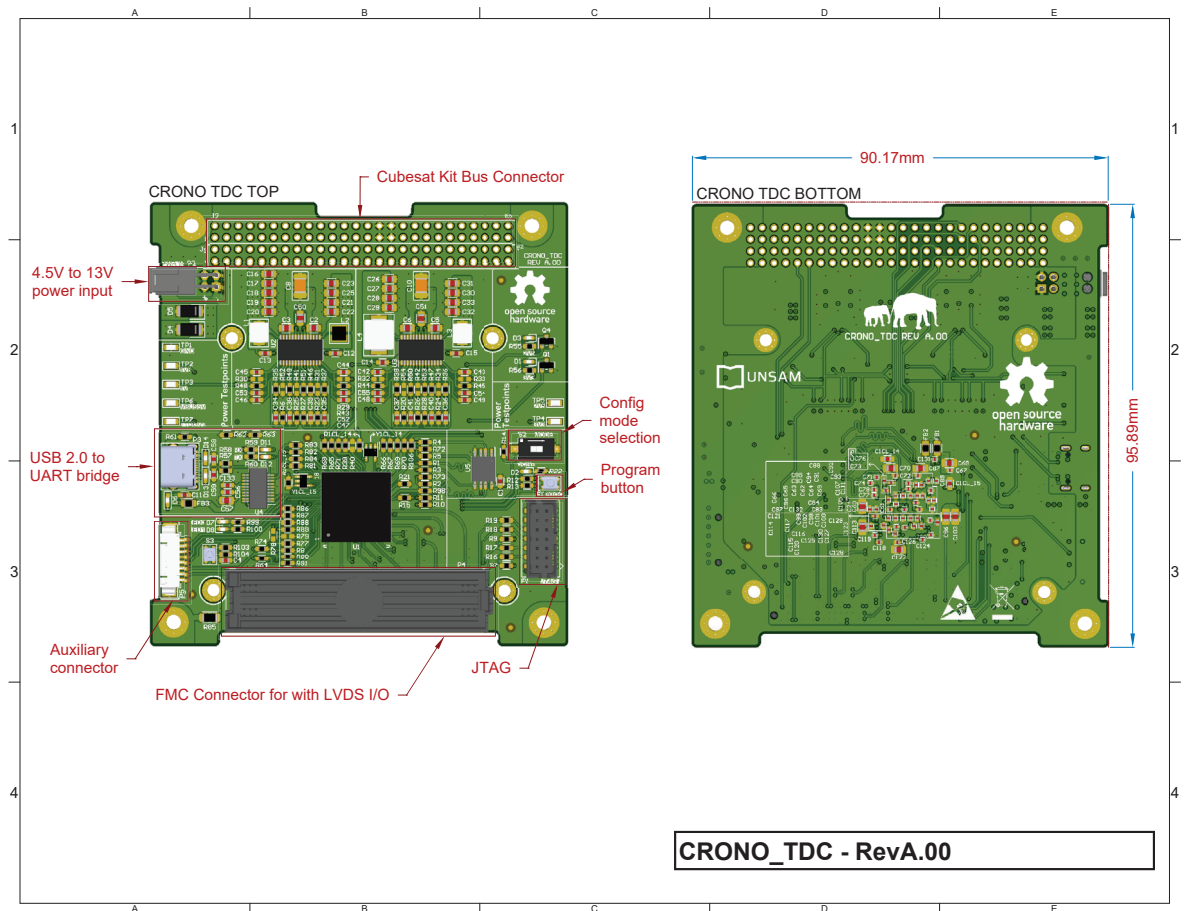


Figura 4.11: Resultado final del diseño del PCB del TDC. En rojo, se encuentran señalizadas ciertas funcionalidades del PCB.

4.12. Resumen

En este capítulo se describió en detalle el diseño de un PCB el cual contiene una FPGA XC7A35T-2CSG324I con 19 interfaces LVDS expuestas a través de un conector FMC. Este PCB respeta la especificación *CubeSat Kit PCB*, por lo cual podría ser fácilmente integrado en un Cubesat. También respeta (en mayor medida) la especificación VITA 57.1, por lo que podría servir como *carrier* para algún módulo FMC comercial que posea un sensor de interés.

En cada una de las interfaces LVDS que posee el PCB se puede implementar un canal de TDC, lográndose el objetivo de diseño de poseer un TDC escalable. Se realizó un análisis del ancho de banda que deben poseer estas interfaces para que las señales de interés del cliente (el laboratorio) puedan ser medidas sin encontrarse perjudicada la integridad de la señal. Toda la interconexión, desde los pines de la FPGA hasta los pines del conector FMC fue diseñada con un ancho de banda de $1,166\text{ GHz}$, por lo cual, se podrían medir señales cuyo tiempo de subida sea tan pequeño como $0,6\text{ ns}$.

Se detalló también el diseño de la fuente de alimentación para la FPGA, con todas las consideraciones pertinentes debido a su complejidad. También, se describieron los mecanismos por los cuales la FPGA puede ser configurada y las interfaces de IO que fueron expuestas.

Por último, se indicaron ciertos aspectos de las instrucciones de fabricación y el repositorio donde se encuentran los archivos de manufactura.

En el siguiente capítulo se introducirán los ensayos llevados a cabo en la prueba de concepto, donde se comentará el procedimiento de los mismos y los resultados obtenidos.

5. Prueba de concepto

En este capítulo se detalla la prueba de concepto ejecutada para validar el funcionamiento del TDC. El objetivo de la misma es ejercitar cada uno de los bloques del *core*: los canales de entrada, la lógica de procesamiento y ejecución de comandos y los encargados de implementar el protocolo de entramado.

Otro objetivo de la prueba es verificar el ancho de pulso mínimo que puede ser medido con el TDC. Aquí se encontró una limitación, dado que se disponía solamente de placas de desarrollo de FPGA interconectadas a través de interfaces cuyas impedancias no estaban adaptadas.

En las siguientes secciones se describe el procedimiento y el alcance de los ensayos realizados en esta prueba de concepto. Por otro lado, también se describe el desarrollo y verificación de un generador de pulsos arbitrarios de alta velocidad. Se utilizó este generador para excitar al TDC con una señal de entrada conocida, para luego contrastarla con los datos generados por el TDC. Por último, se expone la utilización de recursos de la FPGA que demanda el *core* diseñado.

5.1. Generador de pulsos arbitrarios

Para poder validar el funcionamiento del TDC se diseñó e implementó en una FPGA un generador de patrones arbitrarios. La finalidad del mismo es exponer al TDC a una señal de alta velocidad conocida, a través de una interfaz diferencial, para luego reconstruir la forma de onda con los datos que el mismo haya registrado.

El objetivo principal de este generador es emitir pulsos en el orden de los 10 *ns* a través de una interfaz diferencial. Teniendo en cuenta esto, el generador solamente consta de un contador incremental el cual indexa un registro que contiene el patrón escogido. Por lo tanto, en cada iteración del contador se expone un bit distinto del patrón escogido.

El contador es habilitado al detectarse un flanco ascendente en un puerto llamado `i_start` lo cual otorga flexibilidad porque permite disparar la generación del patrón a través de un *General Purpose Input/Output* (GPIO) controlado por algún microcontrolador.

El RTL de este bloque se puede observar en la Figura 5.1 y el resultado de una simulación en la cual se genera el patrón de 10 bits `0b1010001010` se puede encontrar en la Figura 5.2. En la simulación se puede apreciar que cuando el circuito detecta un flanco ascendente en la señal `i_start` el contador se reinicia. En cada cuenta, un nuevo bit del registro `r_pattern` se transfiere al puerto de salida `o_pattern`. Esto provoca que el intervalo de tiempo ocupado por cada bit del patrón sea igual al período de la señal de reloj con la cual está funcionando el circuito.

También, el puerto de salida `o_valid` tendrá un valor lógico alto mientras se estén emitiendo bits del patrón. El objetivo de este puerto es notificar a una lógica externa cuando hay una salida válida.

Cabe destacar que en las simulaciones funcionales se utilizó el patrón de 10 bits `0b1010001010`, pero se puede configurar cualquier otro a través de un parámetro.

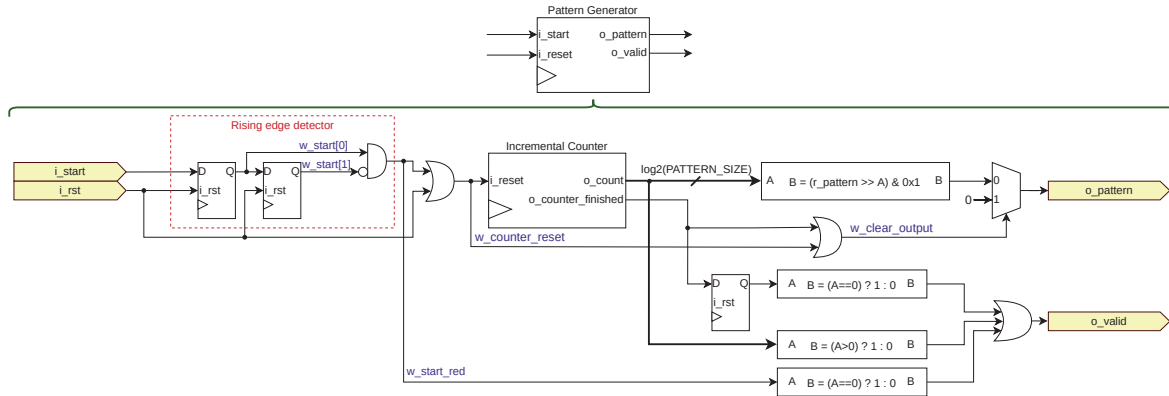


Figura 5.1: Diagrama del generador de patrones.

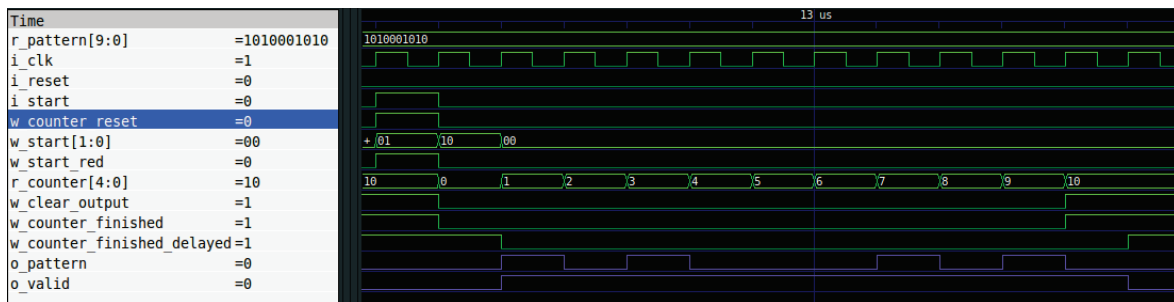


Figura 5.2: Simulación del generador de patrones utilizando el patrón de 10 bits 0b1010001010.

5.1.1. Prueba de laboratorio

Se verificó el comportamiento del generador de pulsos sincronizándolo con señales de reloj de diversas frecuencias: 10 MHz, 50 MHz, 100 MHz, 200 MHz, 250 MHz y 300 MHz. El patrón de prueba escogido para este ensayo fue una sucesión de 10 bits: 0b1010001010.

La placa de desarrollo utilizada para generar los pulsos fue la Arty Z7-10, la cual contiene un SoC XC7Z010-1CLG400C [62]. La interfaz eléctrica utilizada se denomina Señal Diferencial de Transición Minimizada (o *Transition Minimized Differential Signaling* en inglés) (TMDS), la cual es la misma que se utiliza en la norma *High-Definition Multimedia Interface* (HDMI). Un esquema de una interfaz TMDS se puede visualizar en la Figura 5.3, donde la norma [63] especifica $AV_{cc} = 3,3 V \pm 5 \%$ y $R_T = 50\Omega \pm 10 \%$.

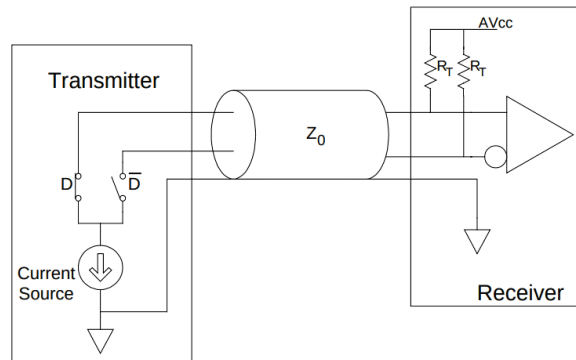


Figura 5.3: Diagrama de una interconexión TMDS. Extraído de [63].

Se mapeó la salida del generador de patrones en un puerto Pmod de la placa Arty Z7-10 y se

colocaron resistores terminadores de $51 \Omega \pm 1\%$. Se excitó el puerto `i_start` del generador con un GPIO de una *Raspberry Pi* y se midió con un osciloscopio entre el positivo V_p y el negativo V_n del par diferencial. Se utilizó la operación MATH del osciloscopio para realizar la diferencia de estas dos señales para obtener la señal TMDS en tiempo real. En la Figura 5.4 se puede observar un diagrama del *setup* montado para esta prueba.

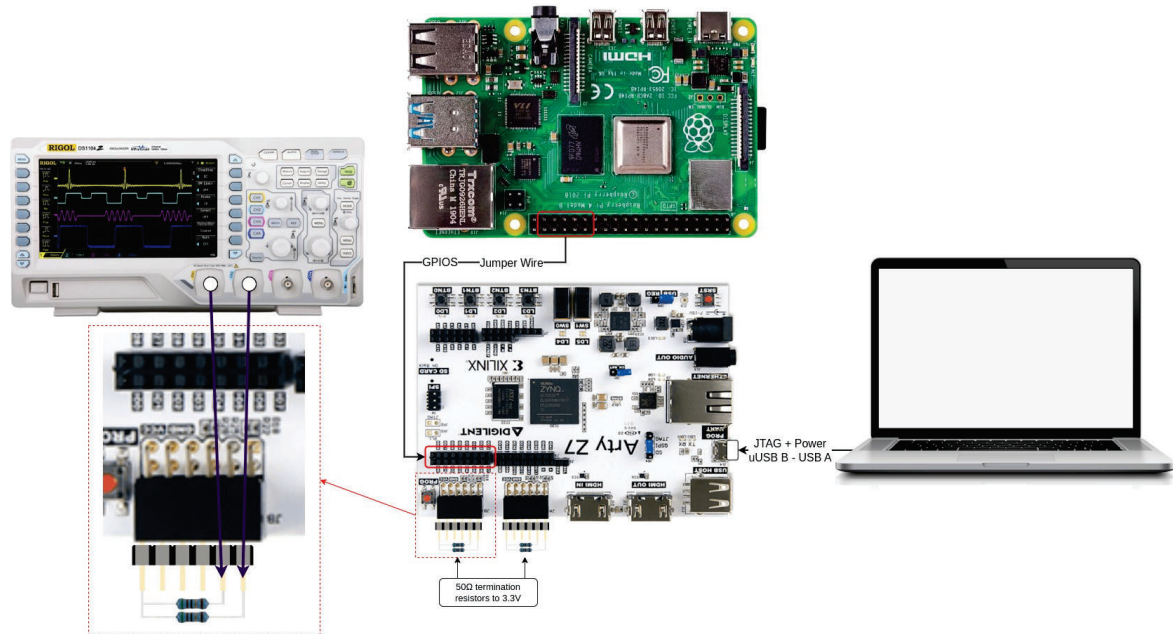


Figura 5.4: Diagrama del *setup* montado para medir la salida del generador de pulsos.

En las primeras pruebas se utilizó un osciloscopio RIGOL DS1104-Z Plus, el cual posee un ancho de banda de 100 MHz (ver Figuras 5.5, 5.6, 5.7 y 5.8). Posteriormente, se utilizó un osciloscopio Tektronix MSO54 el cual posee un ancho de banda de 2 GHz (ver Figuras 5.9, 5.10 y 5.11). Siguiendo la regla práctica definida por la Ecuación 5.1, se estimó que el mínimo tiempo de subida que podría medir el osciloscopio Tektronix MSO54 es 175 ps . Mientras que los tiempos de subida más pequeños que podría medir el Rigol DS1104 son $3,5 \text{ ns}$.

$$\text{risetime} = \frac{0,35}{\text{bandwidth}} \quad (5.1)$$

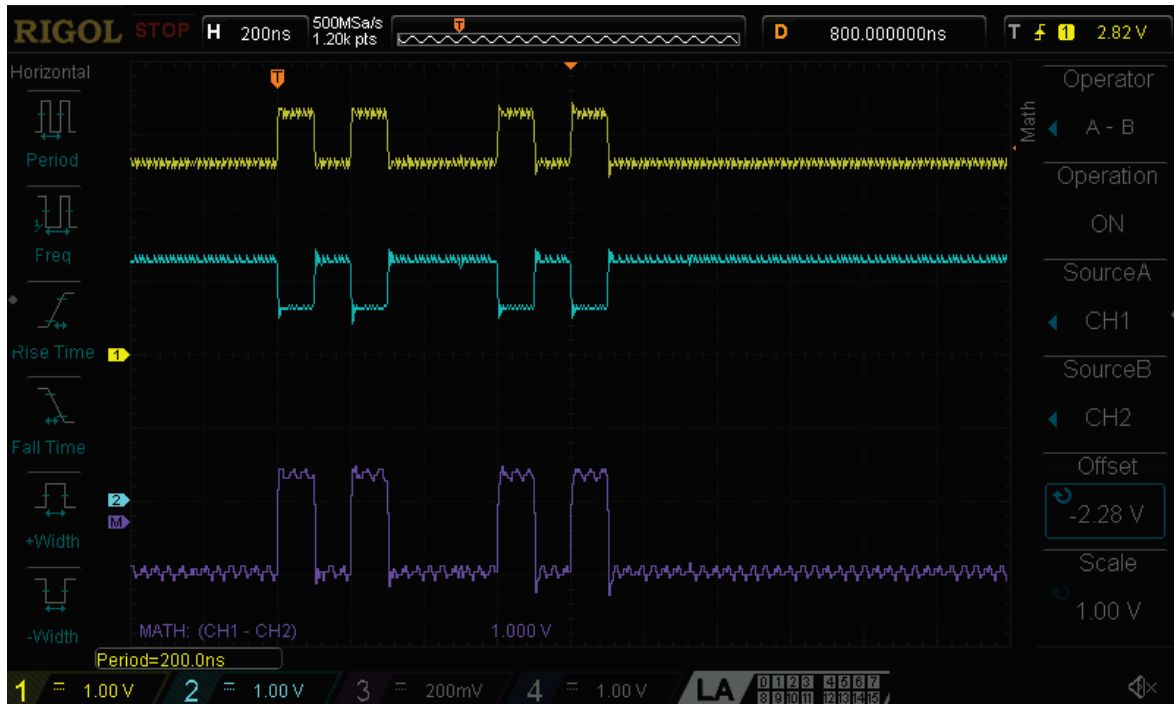


Figura 5.5: Medición del patrón de prueba con una frecuencia base de 10 MHz con el osciloscopio Rigol DS1104-Z Plus. Canal 1: Positivo de la señal TMDS, V_p . Canal 2: Negativo de la señal TMDS, V_n . Canal MATH: Señal TMDS, $V_p - V_n$.



Figura 5.6: Medición del patrón de prueba con una frecuencia base de 50 MHz con el osciloscopio Rigol DS1104-Z Plus. Canal 1: Positivo de la señal TMDS, V_p . Canal 2: Negativo de la señal TMDS, V_n . Canal MATH: Señal TMDS, $V_p - V_n$.



Figura 5.7: Medición del patrón de prueba con una frecuencia base de 100 MHz con el osciloscopio Rigol DS1104-Z Plus. Canal 1: Positivo de la señal TMDS, V_p . Canal 2: Negativo de la señal TMDS, V_n . Canal MATH: Señal TMDS, $V_p - V_n$.



Figura 5.8: Medición del patrón de prueba con una frecuencia base de 200 MHz con el osciloscopio Rigol DS1104-Z Plus. Canal 1: Positivo de la señal TMDS, V_p . Canal 2: Negativo de la señal TMDS, V_n . Canal MATH: Señal TMDS, $V_p - V_n$.

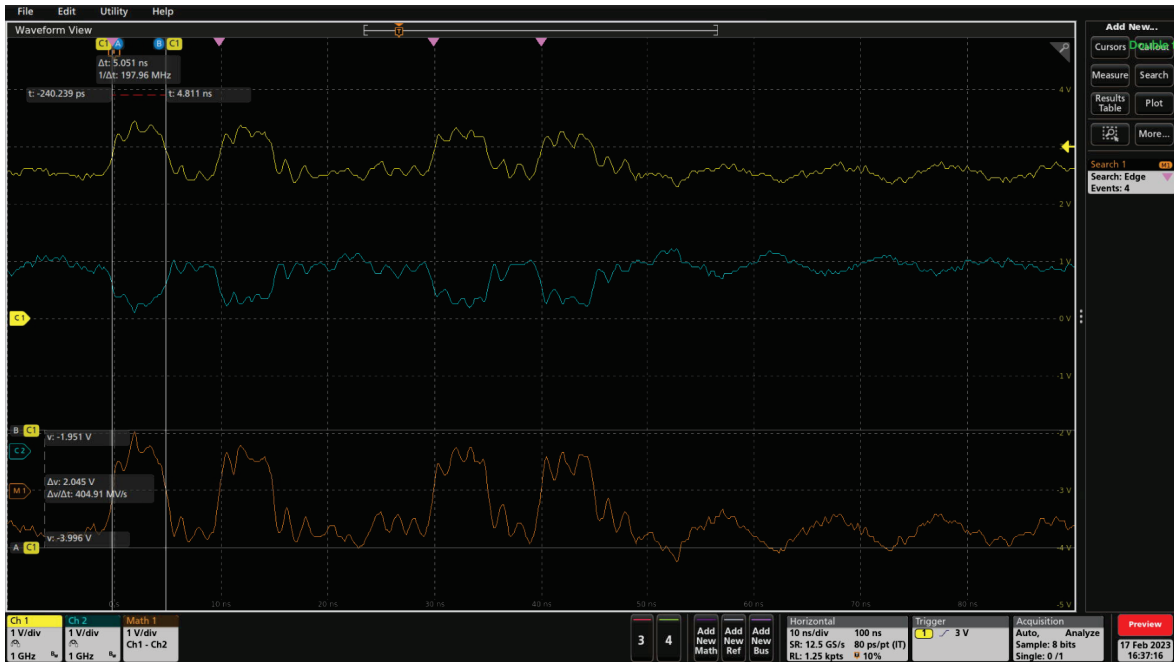


Figura 5.9: Medición del patrón de prueba con una frecuencia base de 200 MHz con el osciloscopio Tektronix MSO54. Canal 1: Positivo de la señal TMDs, V_p . Canal 2: Negativo de la señal TMDs, V_n . Canal MATH: Señal TMDs, $V_p - V_n$.

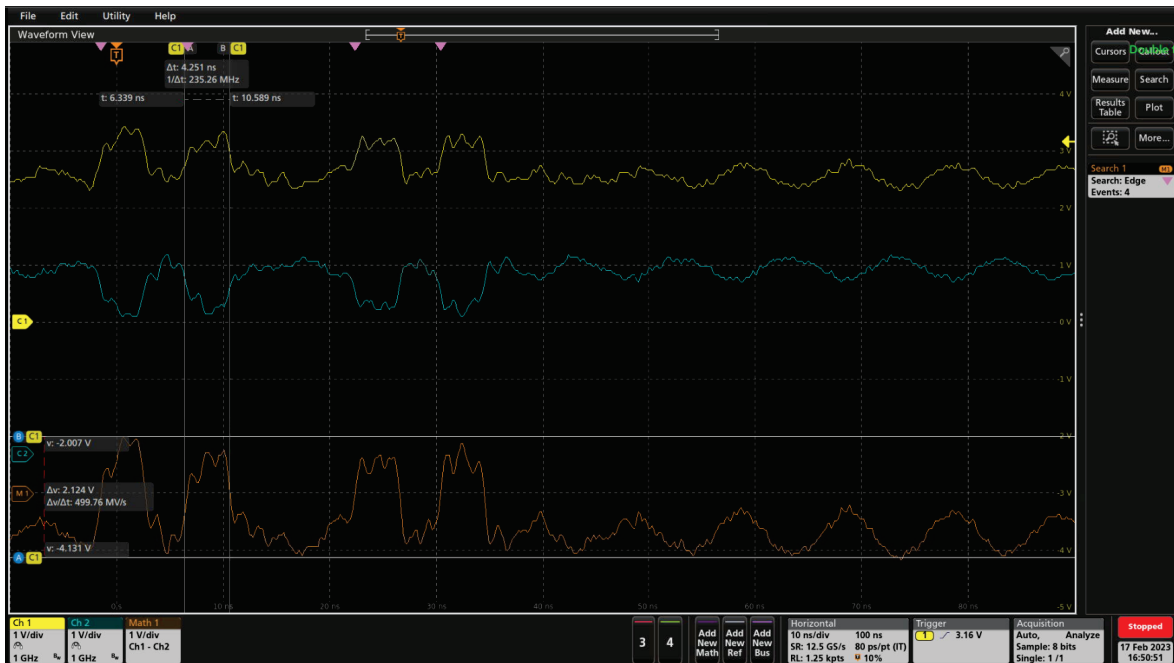


Figura 5.10: Medición del patrón de prueba con una frecuencia base de 250 MHz con el osciloscopio Tektronix MSO54. Canal 1: Positivo de la señal TMDs, V_p . Canal 2: Negativo de la señal TMDs, V_n . Canal MATH: Señal TMDs, $V_p - V_n$.

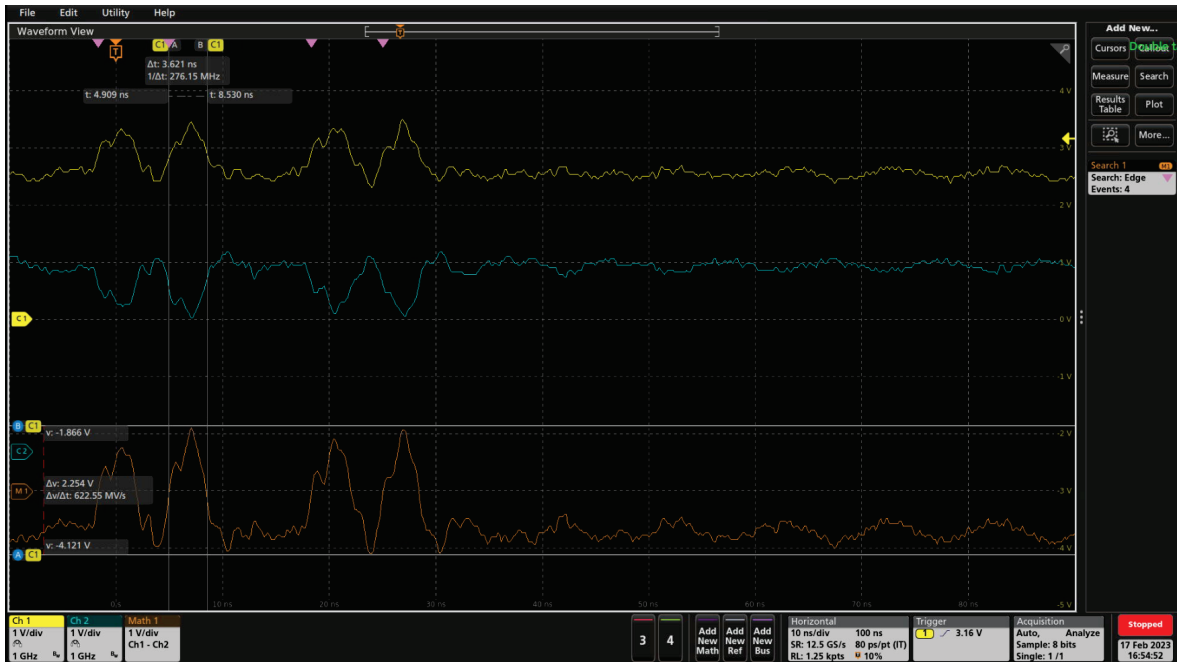


Figura 5.11: Medición del patrón de prueba con una frecuencia base de 300 MHz con el osciloscopio Tektronix MSO54. Canal 1: Positivo de la señal TMDs, V_p . Canal 2: Negativo de la señal TMDs, V_n . Canal MATH: Señal TMDs, $V_p - V_n$.

Se puede observar que cada bit del patrón ocupa un ciclo de reloj. Por ejemplo, en el caso del patrón de 100 MHz (ver Figura 5.7), cada bit posee una duración de 10 ns .

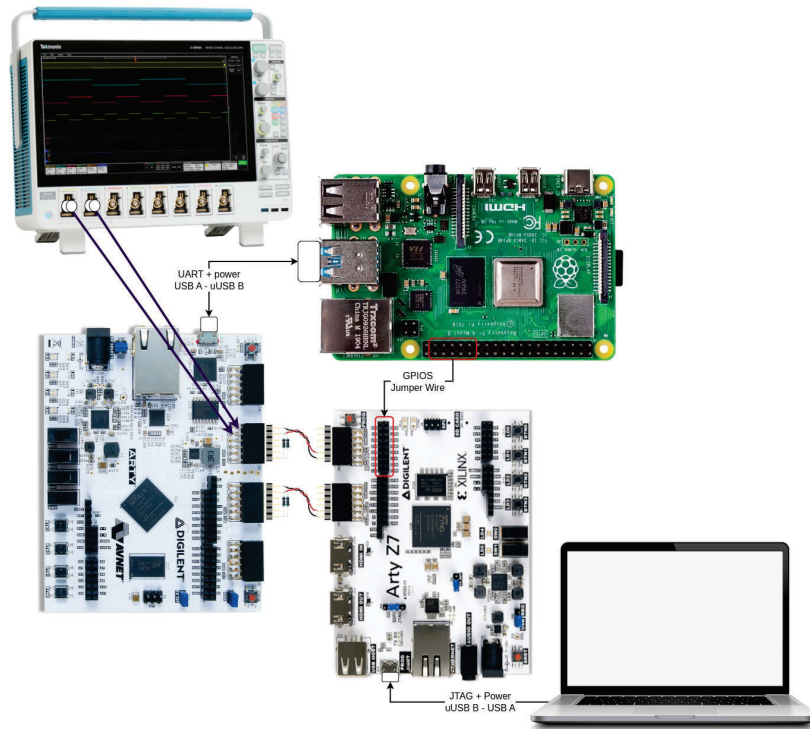
También, se puede observar que a medida que la frecuencia de la señal aumenta la misma comienza a deformarse. Esto se atribuye principalmente a la desadaptación de la impedancia generada por el conector y los resistores terminadores. Esta prueba permitió concluir que se pueden generar pulsos de 5 ns con este *hardware* si se lo utiliza a una frecuencia de reloj de 200 MHz sin encontrarse deformaciones apreciables en la señal.

5.2. Ensayos con la cadena completa

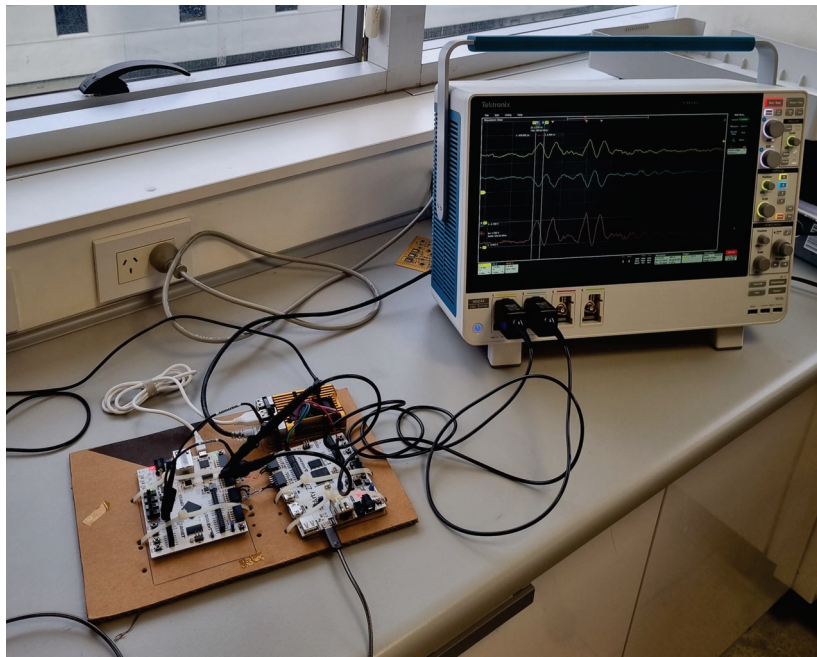
Una vez que fue caracterizado el funcionamiento del generador de pulsos, se procedió a ejecutar un ensayo para verificar el funcionamiento integral del TDC. El mismo consistió en comandar el TDC desde una computadora a través de un puerto serie para configurarlo y lanzar una medición. Dentro de la ventana de tiempo de medición, se utilizó el generador de pulsos para presentarle al TDC una señal de entrada conocida. Mientras tanto, se midió con el osciloscopio la entrada de un canal del TDC para conocer con exactitud la forma de onda de la señal. Luego, se extrajeron los datos del TDC a través del puerto serie y se graficaron los resultados para comparar con lo medido con el osciloscopio.

Se realizó esta prueba midiendo con dos canales del TDC al mismo tiempo, para validar la funcionalidad de medir en simultáneo más de una entrada. En ambos canales se introdujo el mismo patrón de pulsos de 10 bits: $0b1010001010$.

En la Figura 5.12a se puede observar un diagrama del *setup* necesario para ejecutar esta prueba y en la Figura 5.12b se puede ver el *setup* montado en el laboratorio.



(a) Diagrama del *setup* montado para ejecutar la prueba de la cadena completa de medición.



(b) Fotografía del *setup* montado en el laboratorio para ejecutar la prueba de la cadena completa de medición.

Figura 5.12: *Setup* experimental montado para ejecutar el ensayo de la cadena completa del TDC.

En esta prueba, se configuró el RTL del TDC para que posea 4 canales, dos de ellos ubicados en el puerto Pmod JB y otros dos en el Pmod JC de la placa de desarrollo Arty A7-35. Se habilitó solamente un canal de cada puerto, en particular, los canales 0 y 2. La frecuencia de reloj base a la cual estaba funcionando el TDC era 100 MHz y la ventana de tiempo total de medición configurada fue de 41 us .

Se hicieron pruebas con pulsos de 20 ns (ver Figura 5.13), 10 ns (ver Figura 5.14), 5 ns (ver

Figura 5.15), 4 ns (ver Figura 5.16) y 3,3 ns (ver Figura 5.17). En los gráficos generados con la biblioteca de Python [23], se encuentran en cruces rojas los puntos adquiridos por el ISERDESE y en violeta la forma de onda reconstruida utilizando dichos puntos junto con la información agregada por el TDC en cada una de las mediciones.

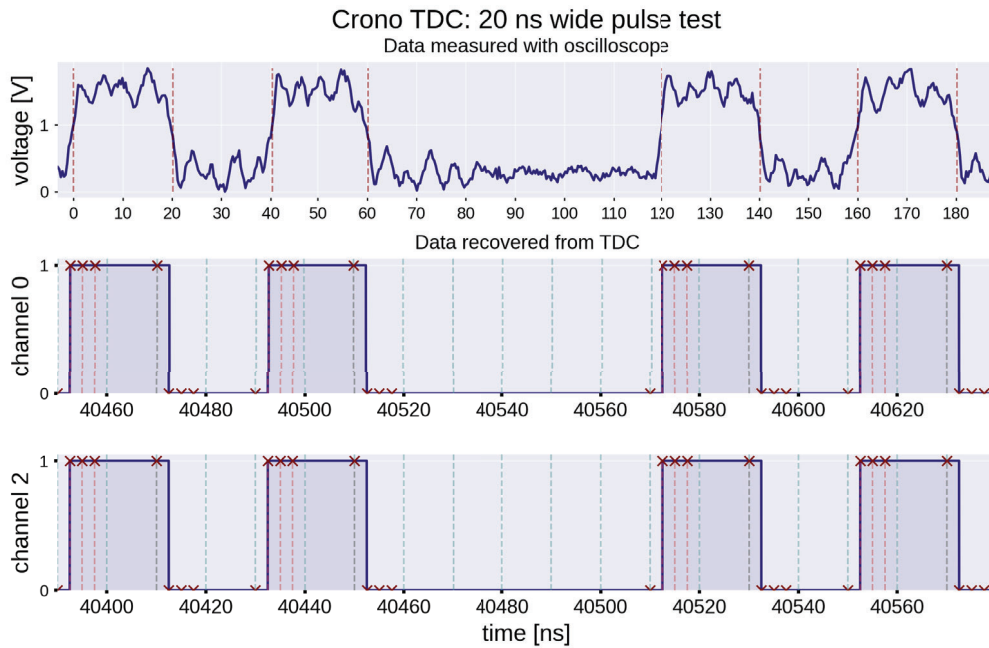


Figura 5.13: Medición de la señal en la entrada de uno de los canales y las formas de onda reconstruidas a partir de los datos del TDC en el ensayo con pulsos de 20 ns.

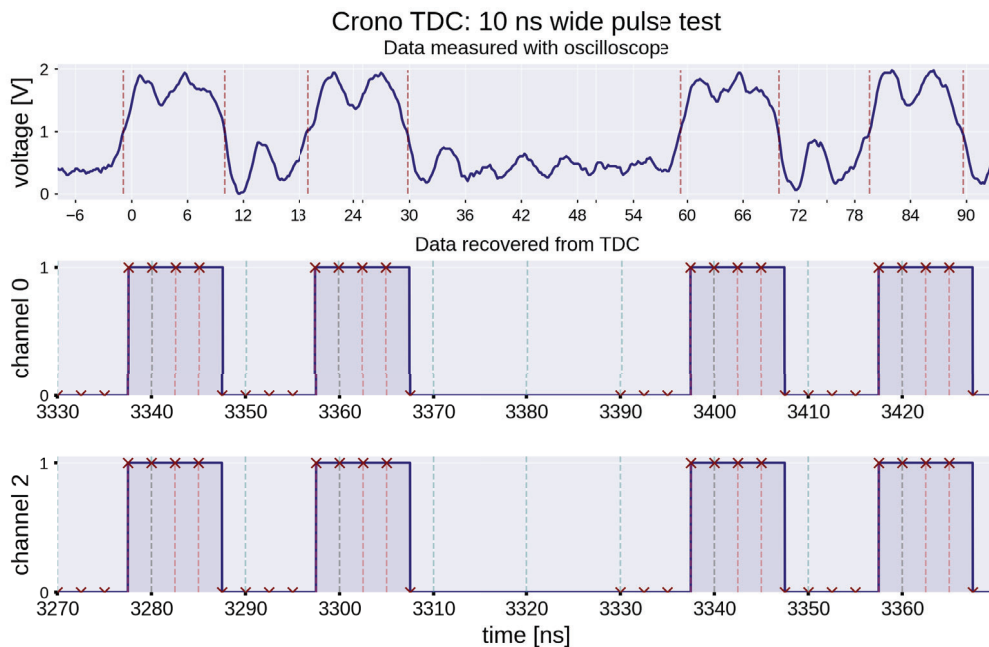


Figura 5.14: Medición de la señal en la entrada de uno de los canales y las formas de onda reconstruidas a partir de los datos del TDC en el ensayo con pulsos de 10 ns.

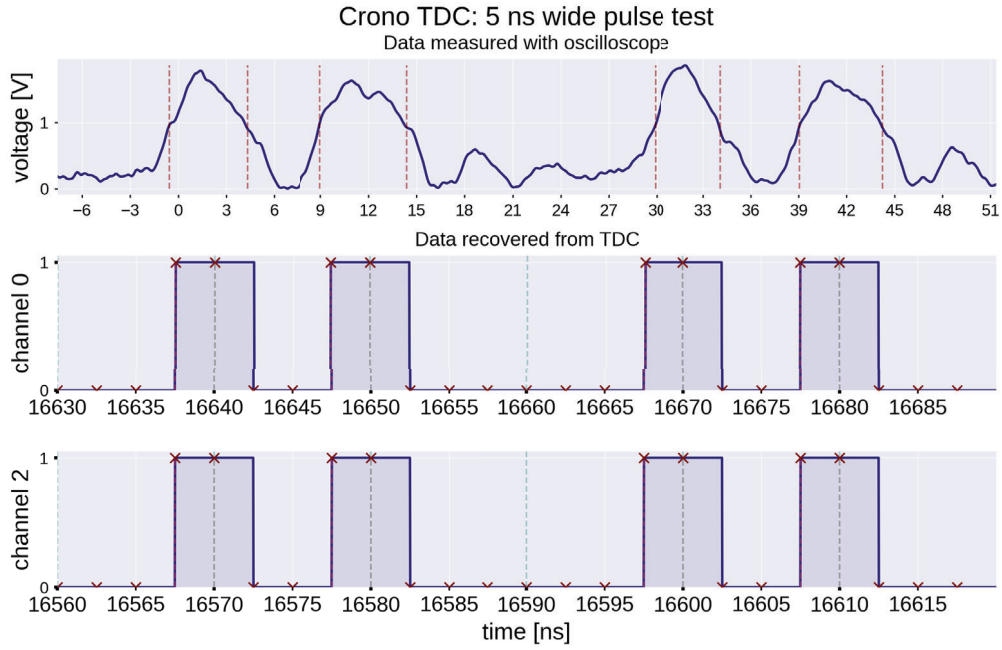


Figura 5.15: Medición de la señal en la entrada de uno de los canales y las formas de onda reconstruidas a partir de los datos del TDC en el ensayo con pulsos de 5 ns.

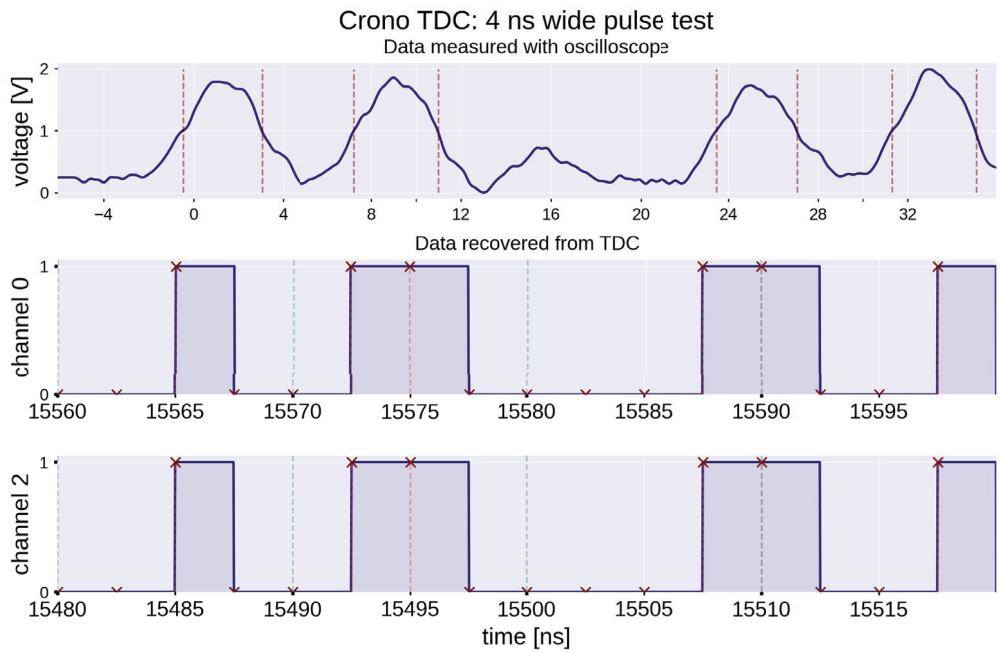


Figura 5.16: Medición de la señal en la entrada de uno de los canales y las formas de onda reconstruidas a partir de los datos del TDC en el ensayo con pulsos de 4 ns.

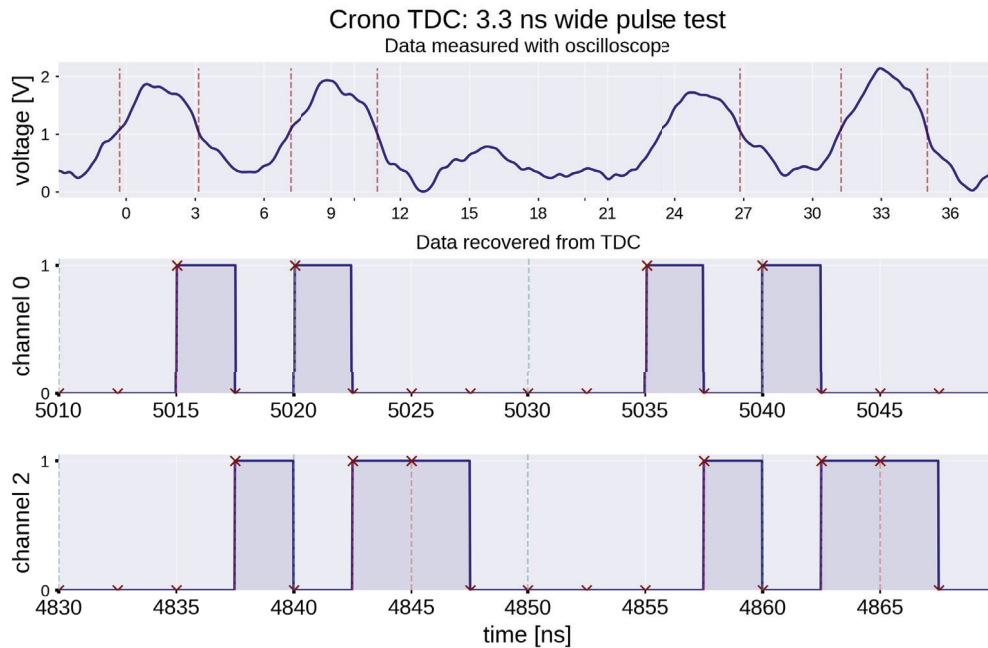


Figura 5.17: Medición de la señal en la entrada de uno de los canales y las formas de onda reconstruidas a partir de los datos del TDC en el ensayo con pulsos de 3,3 ns.

Se puede observar que en el ensayo con pulsos de 20 ns, 10 ns y 5 ns el TDC puede detectar perfectamente la señal de entrada (ver Figuras 5.13, 5.14 y 5.15). La resolución del TDC es un cuarto del período de su señal de reloj, por lo tanto, en este caso fue 2,5 ns y con esta resolución se esperaba medir correctamente las señales de 20 ns, 10 ns y 5 ns.

Pero, en el ensayo con pulsos de 4 ns y 3,3 ns, se esperaba que el TDC encontrara dificultades para poder medirlos correctamente. Se tiene que tener en cuenta que esta arquitectura representa los pulsos de entrada en múltiplos del mínimo intervalo de tiempo que puede medir (el denominado *bin* del TDC). En este caso, el *bin* del TDC es 2,5 ns, por lo tanto, los pulsos pueden ser representados como múltiplos de este número (2,5 ns, 5 ns, 7,5 ns, 10 ns, etc).

Entonces, estos pulsos de 4 ns y 3,3 ns, podrían ser representados con 1 o 2 *bins* del TDC (como se puede apreciar en las Figuras 5.16 y 5.17). Si es 1 o son 2 *bins* dependerá del momento del arribo del flanco ascendente de estos pulsos respecto a los flancos ascendentes de los relojes que alimentan al ISERDESE de cada canal.

A pesar de que el *setup* de prueba no poseía interconexiones cuyas impedancias estuvieran adaptadas, se pudo comprobar el funcionamiento del TDC exitosamente. Se llegó a la conclusión de que el mismo puede detectar pulsos con una incertidumbre atribuida a la arquitectura definida por la Ecuación 1.2, donde $N = 4$.

Cabe destacar que con esta prueba se verifica que el TDC puede reconstruir una señal la cual es asincrónica. Esto se deriva del hecho de que la señal de reloj del generador de pulsos no está en fase con la señal de reloj del TDC. Vale la pena comentar que es notable que no se presentaron signos de metaestabilidades a lo largo del ensayo aun utilizando una señal asincrónica como entrada.

5.3. Utilización de recursos de la FPGA

Con el motivo de lograr una implementación escalable, es crucial conocer la cantidad de recursos que el diseño requiere. Los principales recursos que deben evaluarse son las LUTs ¹, los *flip flops* y las

¹Las LUT son tablas que almacenan una función lógica y pueden ser programadas por el usuario para implementar cualquier función lógica de hasta N entradas (donde N es el tamaño de la LUT).

Block RAMs. La FPGA utilizada (XC7A35T) posee 20800 LUTs de 6 entradas y 41600 *flip flops* [64]. En la Figura 5.18 se puede visualizar el reporte de utilización de cada una de las partes del diseño brindado por Vivado.

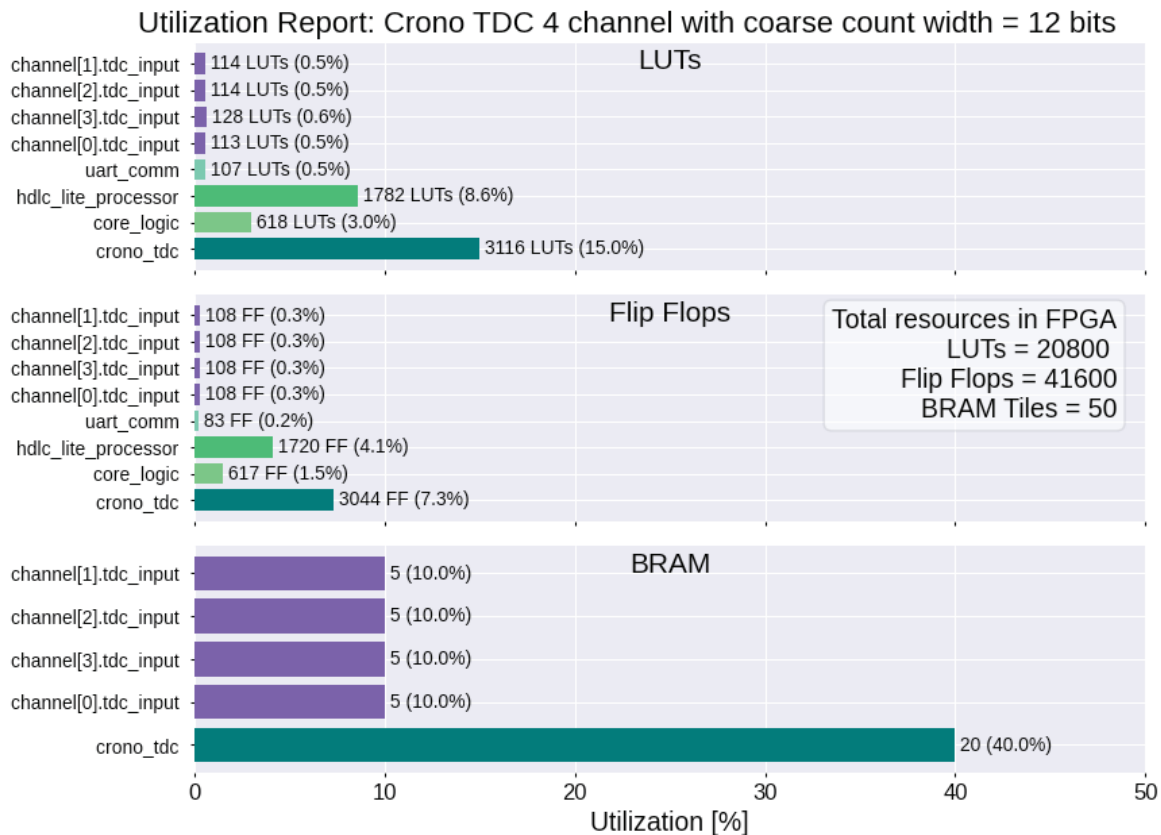


Figura 5.18: Reporte de utilización de LUTs, *flip flops* y BRAMs de cada uno de los módulos del TDC con 4 canales configurados.

Se ha determinado que el *core* diseñado ocupa solamente el 15% de las LUTs y el 7,3% de los *flip flops* disponibles en la FPGA. Además, cada canal de TDC utiliza una fracción muy pequeña de los recursos disponibles. En particular, se ha observado que cada canal de TDC ocupa aproximadamente el 0,5% de las LUTs y el 0,3% de los *flip flops* del chip. En consecuencia, estos recursos lógicos no representan una limitación significativa para la escalabilidad del diseño en términos de la cantidad de canales implementados.

En esta prueba de concepto se utilizó una frecuencia de reloj base de 100 MHz y un tamaño de palabra del contador grueso de 12 bits. Como se puede observar en la Figura 3.10, durante la etapa de diseño se estimó que cada canal de esta configuración ocuparía el 10% de la BRAM total de la FPGA, lo cual se condice con la información del reporte producido por Vivado.

5.4. Resumen

En este capítulo se describió en detalle la prueba de concepto realizada para verificar el funcionamiento del TDC. También se detalló el diseño de un generador de patrones, el cual fue utilizado para introducir señales de alta velocidad en los canales del TDC.

En los ensayos se lograron medir correctamente pulsos de 20 ns, 10 ns y 5 ns, pudiendo recrear su forma de onda solamente a partir de los datos almacenados en el TDC. Sin embargo, se encontraron

dificultades para medir pulsos de 4 ns y $3,3\text{ ns}$, lo cual era esperable, dado que la resolución del TDC en dicha configuración era de $2,5\text{ ns}$.

También, se analizó en detalle la demanda de recursos lógicos y BRAMs de cada módulo individual del *core*. Como resultado, se observó que cada canal del TDC insume solamente el $0,5\%$ de las LUTs y el $0,3\%$ de los *flip flops* de la FPGA.

En el siguiente capítulo se expondrá una discusión integral del desarrollo llevado a cabo en el proyecto, finalizando con su respectiva conclusión.

6. Discusión y conclusión

Este capítulo tiene como objetivo presentar los resultados obtenidos en el proyecto, destacando su relevancia en el área de aplicación correspondiente. Se hace especial énfasis en la importancia de estos resultados y se discuten diferentes alternativas que se consideraron durante el desarrollo del proyecto. Asimismo, se explora el posible trabajo futuro y se exponen las conclusiones finales del proyecto.

6.1. Discusión

En este proyecto se desarrolló un *core* de TDC cuyo objetivo principal consiste en poseer una gran cantidad de canales de entrada. Como se vió en el Capítulo 1, estos dispositivos son cruciales en aplicaciones en las cuales se desea medir el ToF y en física de partículas, incluyendo el PET, el LiDAR y la detección de fotones individuales.

Se escogió una arquitectura de TDC la cual funciona a partir de distintas fases de una misma señal de reloj. Como fue descrito en el Capítulo 2, esta arquitectura presenta algunas ventajas respecto a otras. Por un lado, una fortaleza es que no presenta los errores causados por no linealidades (DNL) encontradas en las arquitecturas más clásicas. Hay que tener en cuenta que los errores causados por las mismas se incrementan en ambientes en los cuales haya grandes fluctuaciones de temperatura. Por otro lado, la resolución de esta arquitectura se encuentra limitada por la frecuencia de reloj utilizada.

La aplicación en cuestión no requiere resoluciones elevadas (alcanza con que sea del orden de las decenas de nanosegundos) y es probable que el TDC se enfrente a ciclos térmicos. Por lo tanto, aquí se encuentra una relación de compromiso en la cual, para la aplicación en cuestión, la arquitectura basada en múltiples fases de reloj resultó la escogida.

Cabe destacar que no se encuentran trabajos de referencia de TDCs que tengan estas características. Durante el desarrollo del proyecto, no se encontró bibliografía que describa esta arquitectura en detalle, sino que se fue reconstruyendo a partir de artículos científicos y reportes técnicos de grupos de investigación. Por lo tanto, el *core* diseñado sienta las bases de un TDC el cual puede ser utilizado en las aplicaciones mencionadas anteriormente, siempre y cuando la resolución necesaria esté dentro de las decenas de nanosegundos. Como el desarrollo es abierto, cualquier grupo de investigación o grupo de trabajo puede consultar el RTL en los repositorios de GitLab [17] [18] [19] para utilizar como referencia. Además del RTL, en estos repositorios se encuentran los *testbenchs* del mismo, para que los interesados puedan validar el trabajo de verificación realizado.

Por otro lado, la aplicación introdujo la necesidad de que este *core* posea una interfaz con la cual un usuario pueda comandarlo a través de un puerto serie. Sin embargo, dado que el diseño resalta por su modularidad, se podría reemplazar el puerto serie y el procesador de tramas por una interfaz AXI esclava para que un microcontrolador embebido en la FPGA pueda comandar directamente al TDC. Como se vió en el análisis de uso de recursos, cada entrada de TDC insume pocos recursos lógicos de la FPGA, por lo tanto, se podría explorar la idea de embeber un Microblaze o un RISC-V.

El módulo que implementa las entradas del TDC, cuya descripción se presentó en detalle en el Capítulo 3.1, resultó ser una solución eficiente y escalable en términos de la cantidad de canales de entrada, gracias a su bajo costo en recursos lógicos. Además, brinda flexibilidad en la configuración de los canales, ya que, mediante la modificación de un parámetro del *top level* y el archivo de *constraints* es posible agregar múltiples canales en el diseño.

La arquitectura seleccionada se centra en el bloque de *hardware* ISERDESE de la familia de FPGAs *Series 7* de Xilinx. Configurado en su modo de sobremuestreo y utilizado en su forma más básica, este bloque permite muestrear cuatro veces en un mismo ciclo de reloj. Como se mencionó anteriormente, la resolución de esta arquitectura se ve limitada por la frecuencia de reloj utilizada y la cantidad de muestras por ciclo de reloj tomadas. La resolución puede mejorarse si se incluyen dos bloques ISERDESE en cada canal de entrada del TDC. El primero se alimentaría con la señal de

reloj base y sus respectivas señales desfasadas 90° , 180° y 270° , mientras que el segundo se alimentaría con las señales de reloj desfasadas 45° , 135° , 225° y 315° . Con esta configuración se tomarían ocho muestras de la señal de entrada en un mismo ciclo de reloj, lo que duplicaría la resolución del TDC sin tener que aumentar la frecuencia del reloj base. Utilizando una frecuencia de reloj base de 100 MHz , se podría lograr una resolución de $1,25\text{ ns}$ con una incertidumbre de cuantización de $2,5\text{ ns}$ con este arreglo.

El sistema de control basado en comandos resultó efectivo en la práctica porque permitió configurar la ventana de medición y decidir con que canales medir sin necesidad de sintetizar un *bitstream* nuevo. Además, con la biblioteca de Python [23] desarrollada para comunicarse con el TDC a través de comandos se otorga repetibilidad de los experimentos ejecutados en este proyecto, dado que los *scripts* utilizados forman parte de la misma. Estos *scripts* están listos para ser utilizados si se quiere comandar al TDC desde un *host* que pueda ejecutar Python, o pueden ser tomados como referencia en el caso en que se lo quiera operar desde un microcontrolador que no esté ejecutando un sistema operativo de propósito general.

El protocolo de entramado HDLC-lite también se vio aprovechado durante los ensayos del TDC. Sin embargo, el escenario donde brillaría sería cuando el TDC se utilizase como carga útil de un Cubesat. En este caso, el TDC se podría comunicar con un microcontrolador que ejecutaría tareas programadas. Por lo tanto, si hubiera una falla, el CRC del protocolo resultaría de gran utilidad.

Finalmente, y considerando todo lo desarrollado anteriormente, durante la prueba de concepto se pudo validar la capacidad del TDC para medir simultáneamente dos señales con pulsos de hasta 5 ns . En esta prueba, el TDC estaba siendo sincronizado con una señal de reloj de 100 MHz , por lo que su resolución era $2,5\text{ ns}$ y su error de cuantización 5 ns . La forma de onda de las señales que poseían pulsos de hasta 5 ns , pudieron ser reconstruidas a partir de los datos almacenados en el TDC. Sin embargo, las señales que poseían pulsos de $3,3\text{ ns}$ y 4 ns no pudieron ser medidas correctamente, lo cual era esperable por las características del desarrollo.

En dicho ensayo, no se encontraron metaestabilidades en los datos almacenados por el TDC, a pesar de que las señales de entrada eran asíncronas. Una técnica usual para disminuir la ocurrencia de metaestabilidades en un circuito digital consiste en colocar 2 o 3 *flip-flops* sincronizadores entre la entrada del circuito secuencial y la señal asíncrona. Dado que el ISERDESE es un arreglo de *flip-flops* conectados de una manera inteligente (Figura 3.2), se asume que es posible que estén cumpliendo el mismo rol que estos 2 o 3 *flip-flops* sincronizadores.

Cabe destacar que el análisis de utilización de recursos de la prueba de concepto arrojó resultados interesantes. Cada canal del TDC insume pocos recursos lógicos de la FPGA, haciendo uso del 0,5% de las LUTs y el 0,3% de los *flip flops*. Sin embargo, existe una relación de compromiso entre el rango dinámico del TDC y el tamaño de la memoria de cada uno de sus canales. Para poder implementar más canales en una misma FPGA sin perder rango dinámico se debería utilizar una que posea más recursos de memoria. Cabe destacar que la FPGA utilizada es pequeña en comparación con otras de la familia *Series 7*. Otra alternativa sería utilizar una memoria externa a la FPGA, lo cual puede ser útil e incluso necesario si se necesita un rango dinámico muy elevado.

Por último, el PCB sienta las bases de un diseño que puede ser integrado en un Cubesat, debido a que respeta el factor de forma y las interconexiones eléctricas de la especificación PC/104-Plus. Esta placa puede ser utilizada para implementar un TDC con 19 de entrada LVDS en una carga útil, actuando de interfaz entre una OBC y un sensor de interés. Sin embargo, para lograr esto se debería realizar un pequeño rediseño, reemplazando algunos componentes comerciales por industriales o *automotive*, para que se encuentren en un rango de temperatura operativo apto para el espacio. También, al respetar (en su mayor medida) la especificación VITA 57.1, el PCB puede utilizarse con sensores comerciales FMC del mercado, lo cual no resulta menor, debido a la gran variedad de módulos comerciales de sensores de alta velocidad disponibles en el mercado.

6.2. Trabajo futuro

En el trabajo realizado se logró alcanzar los objetivos planteados para este proyecto. Durante el desarrollo del mismo se encontraron ciertas propuestas de trabajos futuros para continuar con la línea de investigación.

En primer lugar, se ve una posibilidad de duplicar la resolución de esta arquitectura de TDC si se utilizan dos bloques ISERDESE en sus canales de entrada. Por lo tanto, se propone realizar una variante de diseño de los canales de entrada utilizando dos ISERDESE y validarlo contra las mismas señales de pruebas que se utilizaron en la prueba de concepto. Con esta nueva variante, la resolución sería $1,25 \text{ ns}$, por lo tanto, se podrían llegar a medir los pulsos de $3,3 \text{ ns}$ y 4 ns .

Por otro lado, también se podría dividir el *core* en dos dominios de reloj, sincronizando los ISERDESE de cada entrada con un reloj de mayor velocidad y el resto del diseño con relojes de menor velocidad. Con esto se podría aumentar la frecuencia de reloj de los ISERDESE, aumentando la resolución del TDC, sin tener que incrementar la frecuencia de operación del resto de los bloques del diseño.

Otro trabajo pendiente consiste en embeber un microcontrolador en la FPGA el cual opere al TDC a través de una interfaz AXI. Aquí, se deberá evaluar si el *core* diseñado y el microcontrolador caben en la FPGA utilizada, sin perder de vista de utilizar otra con mayores prestaciones.

Por último, pero no menos importante, el próximo paso debería consistir en la fabricación de un primer prototipo del PCB diseñado. Esta tarea involucra no solo el trato con los fabricantes y proveedores, sino también la puesta en marcha y validación del *hardware*.

6.3. Conclusión

Los objetivos del presente proyecto consistían en desarrollar un TDC que fuera capaz de detectar y medir señales cuya duración temporal sea del orden de las decenas de nanosegundos. Durante el desarrollo del mismo, se logró cumplir con el objetivo propuesto originalmente, dado que se diseñó, verificó y validó el RTL de un *core* de TDC basado en múltiples fases de reloj. La idea clave de esta arquitectura es que el dispositivo puede tomar muestras de la señal de entrada N veces en un ciclo de reloj, logrando una resolución de $\frac{T_{clk}}{N}$. En la implementación, este sobremuestreo se logró con un bloque ISERDESE existente en la FPGA, el cual permite tomar cuatro muestras por ciclo de reloj. Por lo tanto, en los ensayos realizados, utilizando una frecuencia de reloj base de 100 MHz se logró una resolución de $2,5 \text{ ns}$ y una incertidumbre de cuantización de 5 ns . Con esta configuración, se lograron medir señales las cuales contenían pulsos asíncronos cuya duración mínima era 5 ns .

El *core* diseñado también posee una lógica de control para ser operado a través de comandos por un usuario. Se añadió esto al diseño teniendo en mente las futuras aplicaciones, donde se quiera utilizar al TDC a través de un microcontrolador. Esto permite que se puedan configurar ciertos parámetros de las mediciones (como el ancho de la ventana de medición) sin tener que generar un *bitstream* distinto.

Por otro lado, este *core* resuelve la comunicación a través de un puerto serie utilizando un protocolo de entramado el cual provee confiabilidad a la comunicación. Esto sienta la base de un futuro protocolo de comunicación entre un TDC utilizado en una carga útil con una OBC que quiera comandarlo para realizar mediciones y recuperar datos.

En la etapa final del proyecto también se logró el diseño de un PCB que contiene la FPGA XC7A35T-2CSG324I de Xilinx. Esta placa expone 19 entradas LVDS a través de un conector FMC y cumple con la especificación *CubeSat Kit PCB* [1]. Este diseño promete ser un prototipo de un TDC que puede ser utilizado en la carga útil de un Cubesat. Por ejemplo, en este tipo de aplicación el TDC puede medir las señales entregadas por sensores de fotones (SiPM), detectando así fotones individuales.

Finalmente, el trabajo resultó una posibilidad para incorporar herramientas y conceptos nuevos para el estudiante, los cuales resultaron desafiantes por sus características innovadoras y complejidad técnica. La oportunidad de realizar diseños de electrónica digital de alta complejidad y de PCBs de

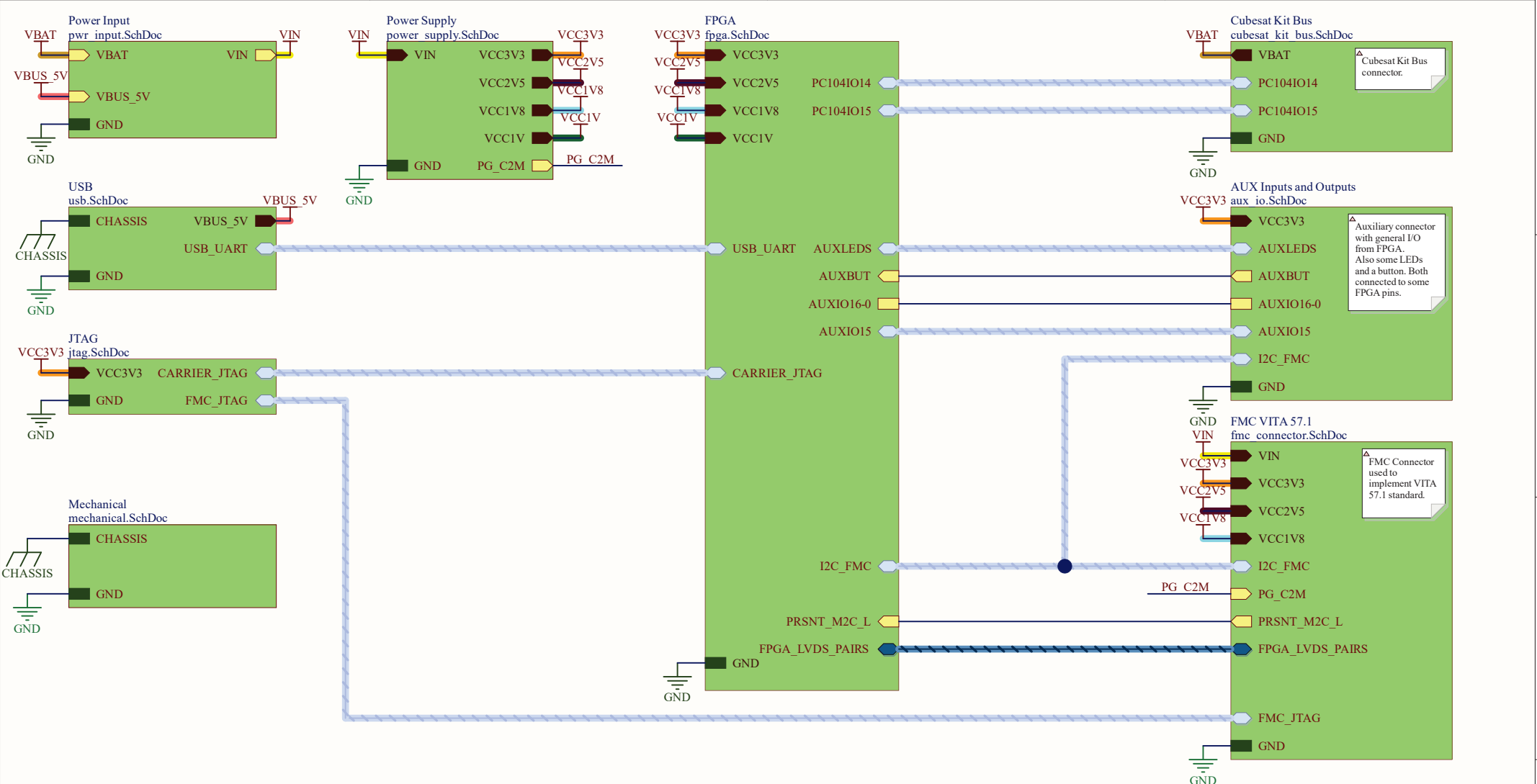
alta velocidad en esta etapa de la carrera del estudiante resulta valioso para su crecimiento profesional, preparándolo para afrontar nuevos desafíos que estén relacionados con esta temática.

A. Esquemáticos e instrucciones de fabricación





En este anexo se encuentran los esquemáticos del PCB desarrollado durante el proyecto y otros esquemáticos tomados como referencia durante el diseño.

A.1. Esquemáticos del PCB desarrollado

A continuación se encuentran los esquemáticos correspondientes al PCB desarrollado durante el proyecto.

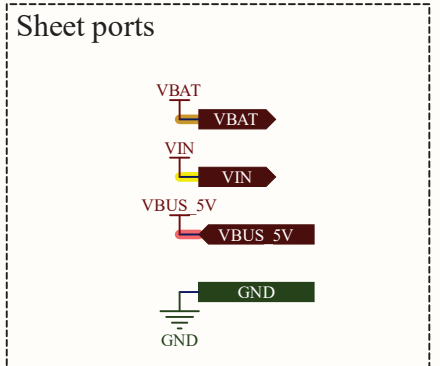
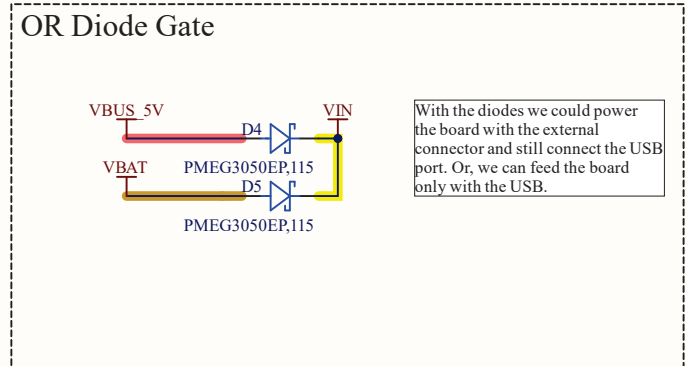
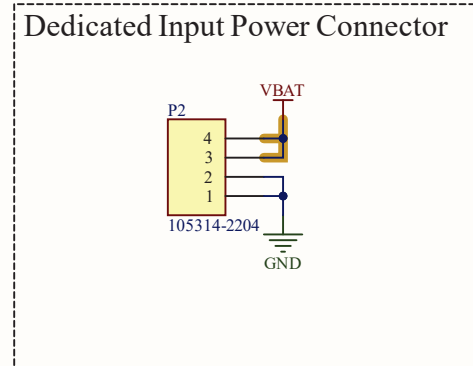



Reference

-  GND
-  Power
-  High Frequency Signals
-  Low Frequency Signals



Project:	CRONO_TDC	
Sheet Name:	crono_tdc.SchDoc	
Revision:	A.00	
Author:	RODRIGUEZ JULIAN	
Reviewer:		
Date:	1/18/2023	
Size:	A4	Sheet 1 of 24

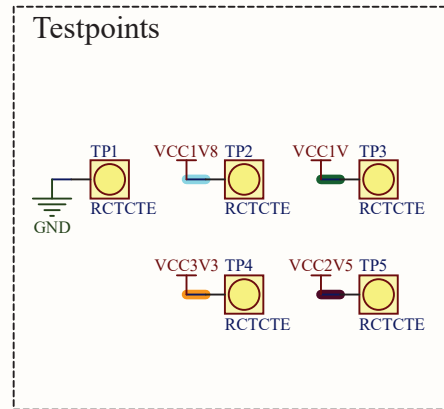
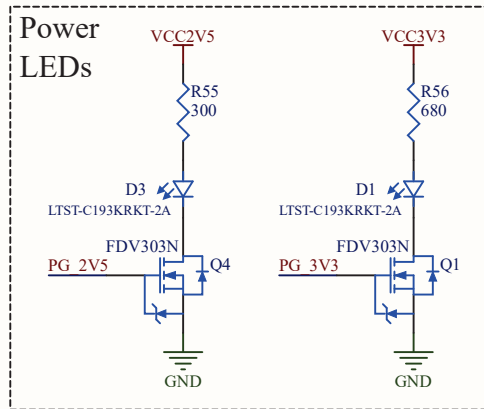
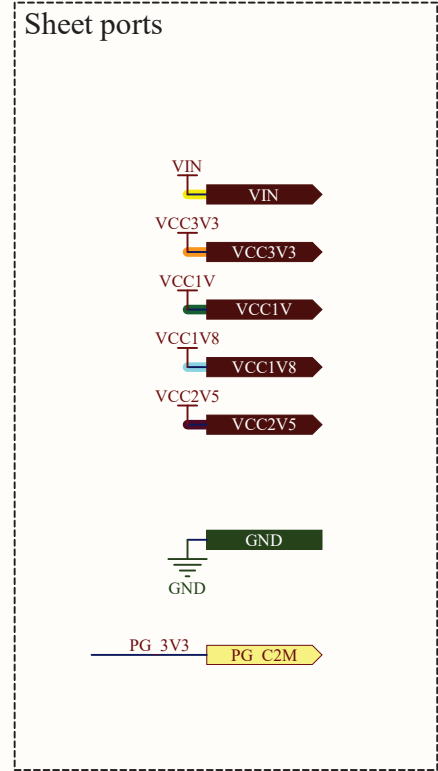
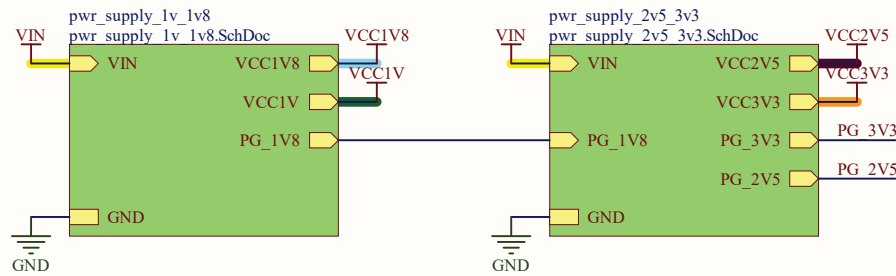


 UNSAM	
Project:	CRONO_TDC
Sheet Name:	pwr_input.SchDoc
Revision:	A.00
Author:	RODRIGUEZ JULIAN
Reviewer:	
Date:	
Size: A4	Sheet 2 of 24

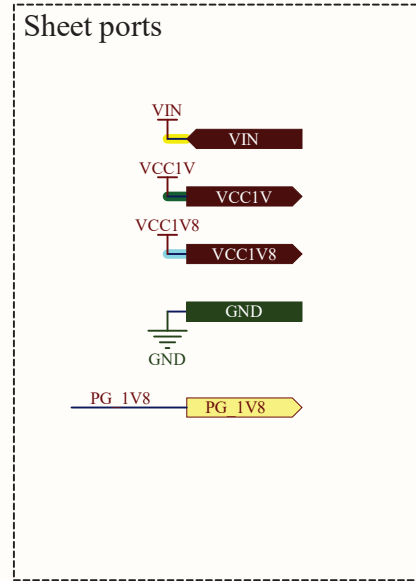
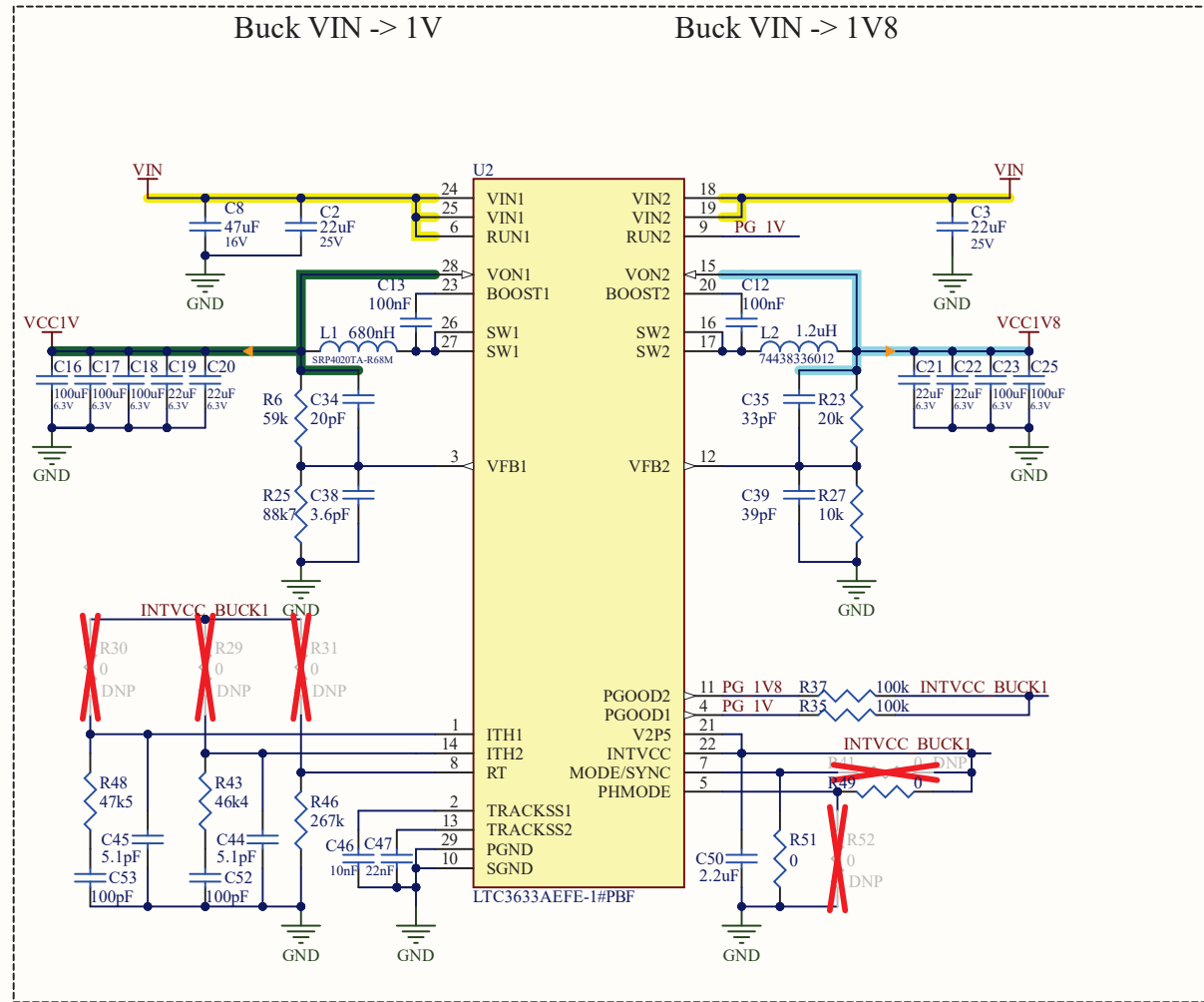
The power supplies follows a determined startup sequence:
 Buck DC-DC 1V -> Buck DC-DC 1V8 -> Buck DC-DC 3V3 & 2V5

This is configured connecting the power good of a power supply to an enable signal of another power supply.

Every power supply (VCC1V, VCC1V8, VCC2V5 and VCC3V3) have a soft-start configured of some milli-seconds.



Project:	CRONO_TDC
Sheet Name:	power_supply.SchDoc
Revision:	A.00
Author:	RODRIGUEZ JULIAN
Reviewer:	
Date:	1/18/2023
Size:	A4



-INTVCC: Internal 3.3V regulator output.

-V2P5: Internal 2.5V regulator output (max 10mA). Tied to INTVCC because it is not used.

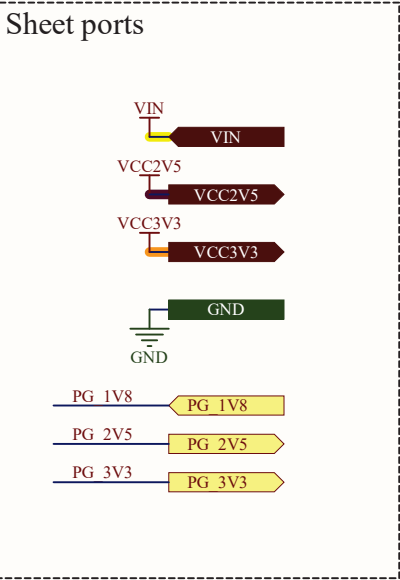
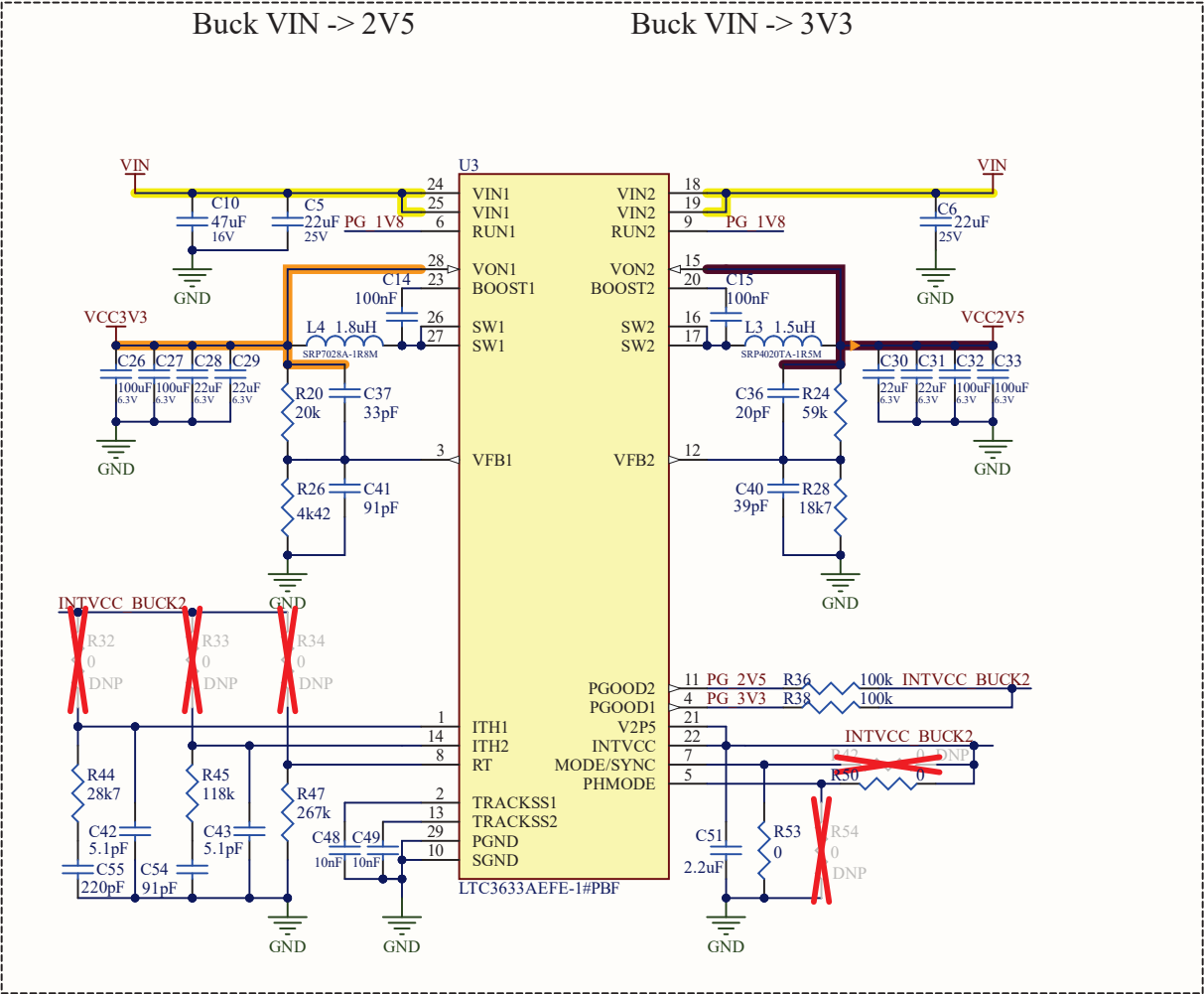
-MODE/SYNC: Force continuous synchronous operation tying to GND. Select "Burst Mode" tying to INTVCC or floating it. By default, "Continuous Sync" is selected.

-PHMODE: Tie this pin to GND to force both channels to switch in phase. Tie this pin to INTVCC to force both channels to switch 180° out of phase.

-RT: By default, leave a 267kOhm pull-down to config 1.2MHz switching frequency. But, leave a 0 Ohm pull-up to INTVCC to select 2MHz switching frequency if needed.

-ITH1/2: Use custom compensation network. Or drive this pins to INTVCC with a 0 Ohm pull-up to use internal compensation. If internal compensation is used, external compensation network components must be removed.

Project:	CRONO_TDC
Sheet Name:	pwr_supply_1v_1v8.SchDoc
Revision:	A.00
Author:	RODRIGUEZ JULIAN
Reviewer:	
Date:	
Size:	A4
Sheet 4 of 24	



-INTVCC: Internal 3.3V regulator output.

-V2P5: Internal 2.5V regulator output (max 10mA). Tied to INTVCC because it is not used.

-MODE/SYNC: Force continuous synchronous operation tying to GND. Select "Burst Mode" tying to INTVCC or floating it. By default, "Continuous Sync" is selected.

-PHMODE: Tie this pin to GND to force both channels to switch in phase. Tie this pin to INTVCC to force both channels to switch 180° out of phase.

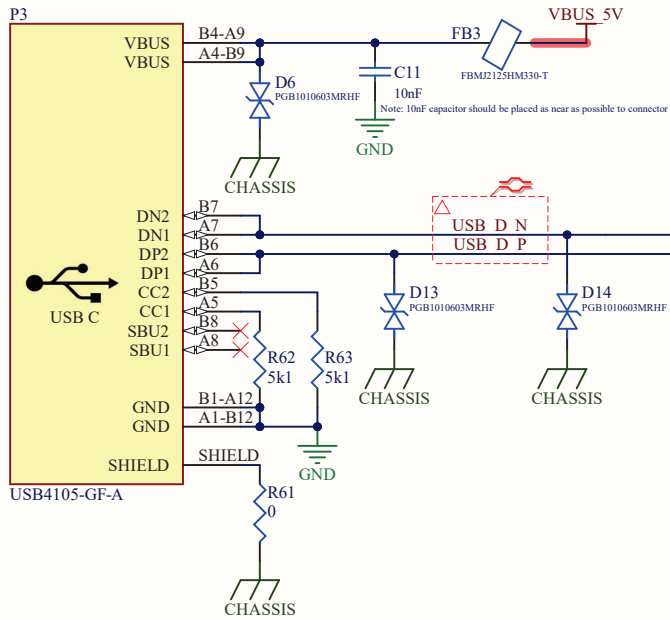
-RT: By default, leave a 267kOhm pull-down to config 1.2MHz switching frequency. But, leave a 0 Ohm pull-up to INTVCC to select 2MHz switching frequency if needed.

-ITH1/2: Use custom compensation network. Or drive this pins to INTVCC with a 0 Ohm pull-up to use internal compensation. If internal compensation is used, external compensation network components must be removed.

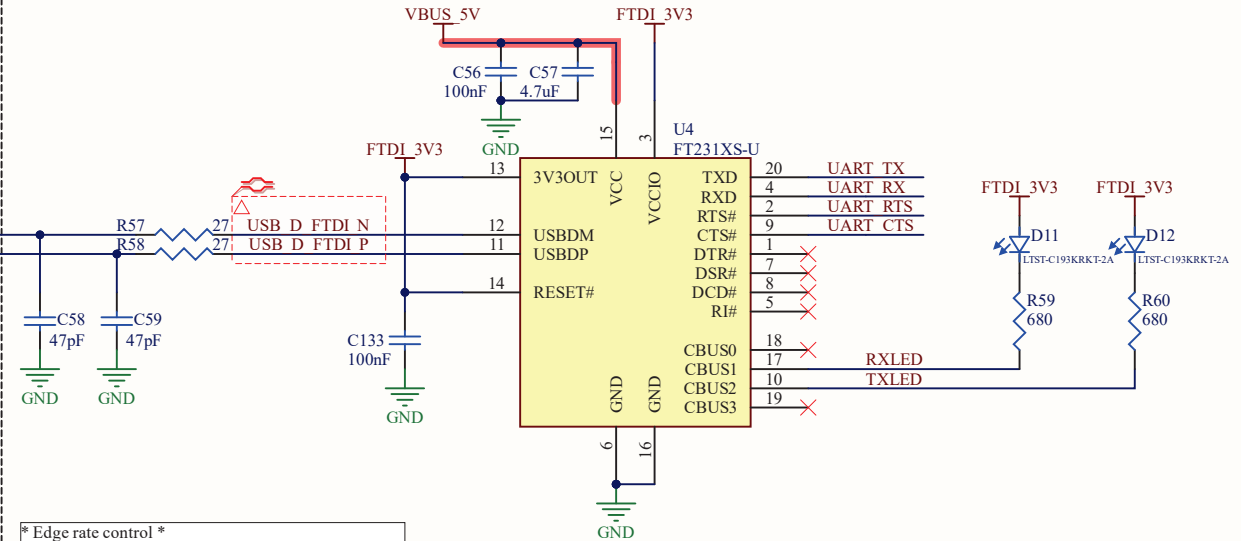


Project:	CRONO_TDC
Sheet Name:	pwr_supply_2v5_3v3.SchDoc
Revision:	A.00
Author:	RODRIGUEZ JULIAN
Reviewer:	
Date:	
Size:	A4

USB C Connector



USB - UART Bridge



*** Edge rate control ***
The capacitors (47 pF) in data lines are used to adjust the rise/fall time of the signals (see AN146).

*** Line termination ***
This family of chips needs a 27 Ω resistor in each data line (see AN146).

*** VCC ***
3.3V or 5V power supply input for the overall IC.

*** VCCIO ***
1.8V - 3.3V power supply input for I/O cells.

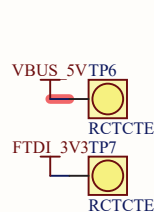
*** 3V3OUT ***
3V3OUT is the internal 3.3V regulator output. This regulator can be used to power VCCIO.

*** CBUS ***
These pins could be configured to be used in several ways. These configurations could be done using a software utility provided by FTDI. But, every pin has a default configuration:

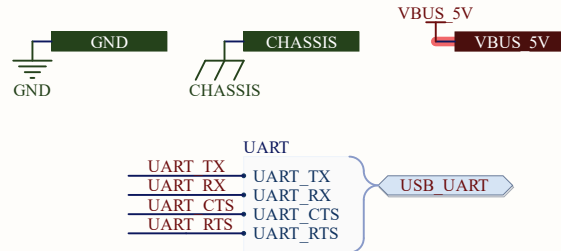
- CBUS0: This pin has TXDEN as default configuration. This pin acts as an output.
- CBUS1: This pin has RXLED# as default configuration. In this configuration, it will pulse low when the device is receiving data through USB.
- CBUS2: This pin has TXLED# as default configuration. In this configuration, it will pulse low when the device is sending data through USB.
- CBUS3: This pin has SLEEP# as default configuration. This pin acts as an output.

*** Unused pins ***
- DSR#, DCD# and RI# are inputs that are pulled up internally. So, we did not need an external pull-up.
- DTR# is an output pin.

Testpoints



Sheet ports



*** SBU1/2 ***
Corresponds to the low-speed signal path used only in standby mode, so we are ignoring them here.

*** CC1/2 ***
The channel configuration pins on the device side (this board is a device) tell the host (a PC) that a connection has been made with a device and VBUS should be turned on.

This pin is used to determine how much current is available from the upstream port (host). This voltage is determined with a voltage divider between the host and the device.

USB C standard requires that CC1 and CC2 be pulled down to ground with 5.1kΩ resistors. The host will have a different value of pull-up that determines the amount of current that can supply.

*** DP1/2 ***
Should be shorted together (are redundant to make reversible the insertion).

*** DN1/2 ***
Should be shorted together (are redundant to make reversible the insertion).

*** Shield ***
Case shield is connected to the chassis through a 0 Ω resistor to allow flexibility in the best component selection to minimize signal noise while providing EMC compatibility.

*** TVS ***
These TVS diodes must be connected as close as possible to the USB connector.

*** Ferrite Bead ***
The ferrite bead and the 10nF capacitor should be connected as close as possible to the USB connector.



Project: **CRONO_TDC**

Sheet Name: **usb.SchDoc**

Revision: **A.00**

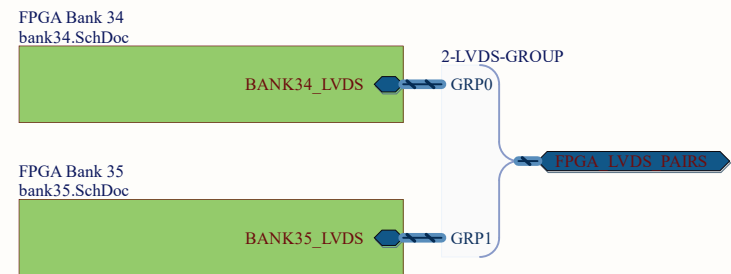
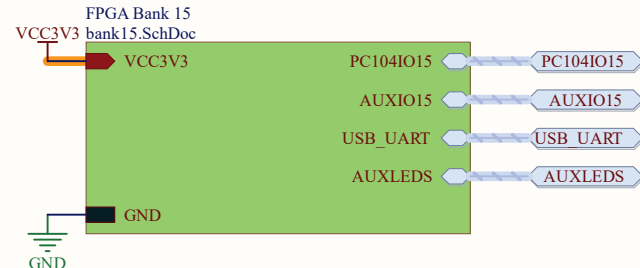
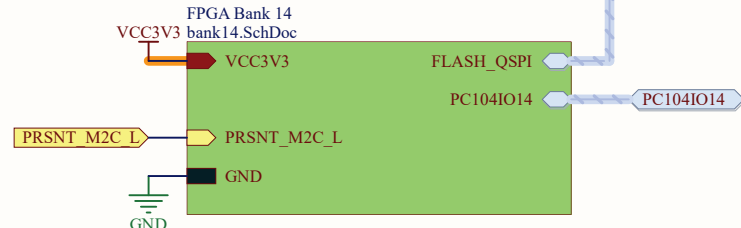
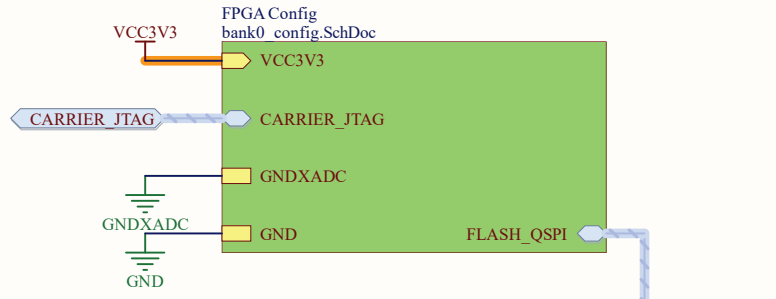
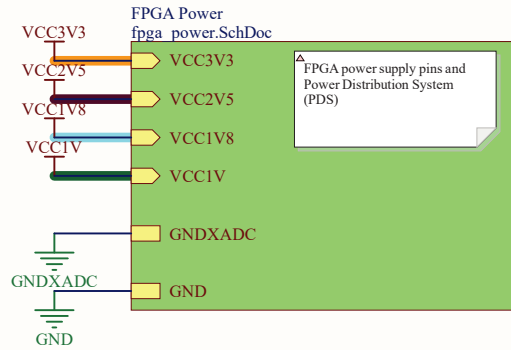
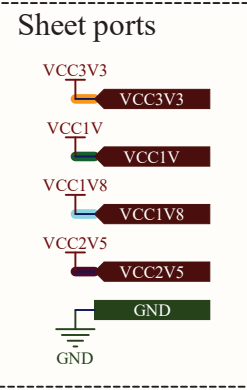
Author: **RODRIGUEZ JULIAN**

Reviewer:

Date: **1/18/2023**

Size: **A4**

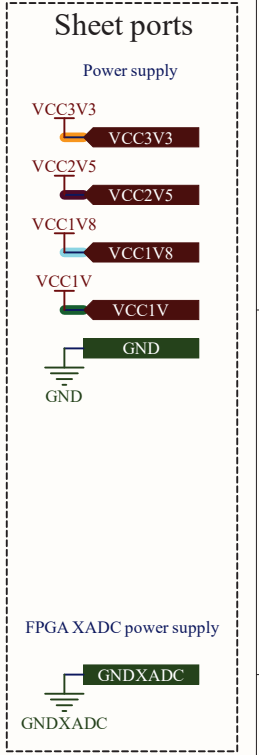
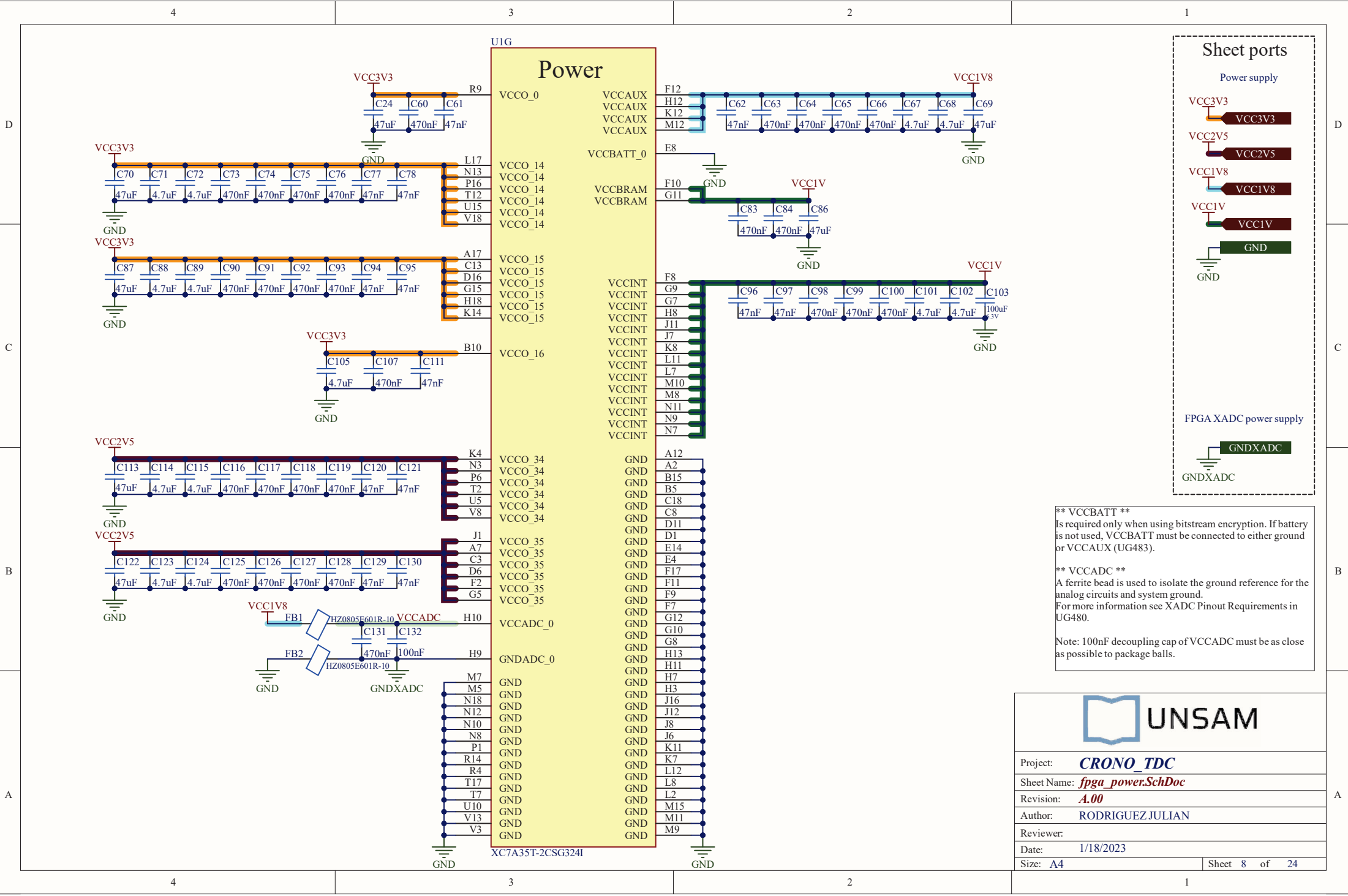
Sheet **6** of **24**



UNSAM

Project:	CRONO_TDC
Sheet Name:	fpga.SchDoc
Revision:	A.00
Author:	RODRIGUEZ JULIAN
Reviewer:	
Date:	
Size:	A4

Sheet 7 of 24



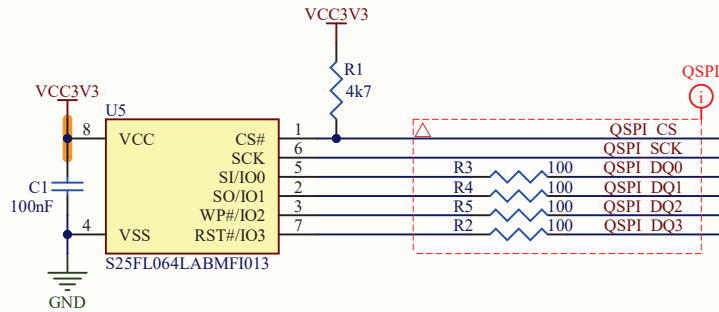
**** VCCBATT ****
 Is required only when using bitstream encryption. If battery is not used, VCCBATT must be connected to either ground or VCCAUX (UG483).

**** VCCADC ****
 A ferrite bead is used to isolate the ground reference for the analog circuits and system ground. For more information see XADC Pinout Requirements in UG480.

Note: 100nF decoupling cap of VCCADC must be as close as possible to package balls.

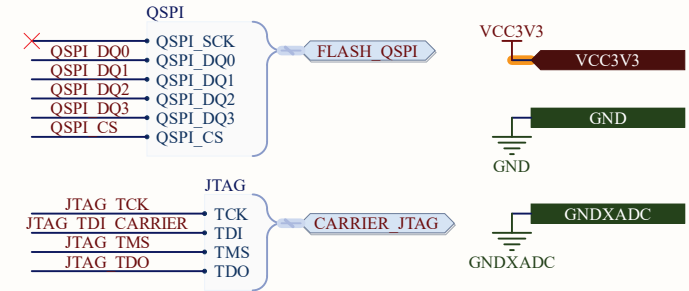
Project:	CRONO_TDC
Sheet Name:	fpga_power.SchDoc
Revision:	A.00
Author:	RODRIGUEZ JULIAN
Reviewer:	
Date:	1/18/2023
Size:	A4

QSPI Flash Memory



QSPI Flash Memory is used to store FPGA configuration between reboots. When M[2:0] = 001, the FPGA will load the bitfile from the QSPI flash memory.

Sheet Ports



FPGA BANK 0

*** JTAG ***
Xilinx recommends pull-up TCK, TDI and TMS to improve signal integrity.

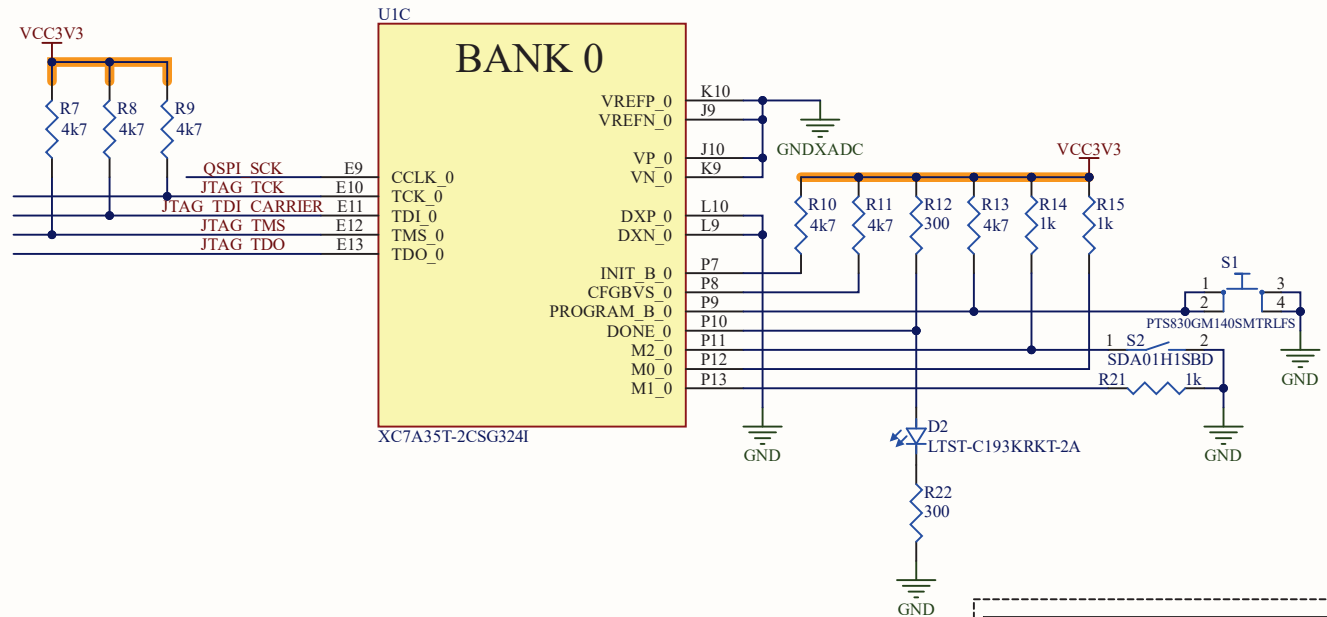
*** FPGA Configuration Selection mode ***
M[2:0] = 101 -> JTAG: FPGA will load bitfile from QSPI flash memory.
M[2:0] = 001 -> Master SPI mode: FPGA will load bitfile from QSPI flash memory.
Mode pins have internal pull-ups. Could be connected either directly or via a 1kΩ resistor (or smaller) to GND or VCC0_0.

*** DONE ***
This pin is open drain. When a configuration sequence is finished, this pin is set high and the LED will turn on.

*** PROGRAM_B ***
When PROGRAM_B is driven low, the FPGA erases its bitstream and a new configuration sequence is started. The method of this bitstream load depends on M[2:0] pins.

*** CFGBVS ***
Configuration Banks Voltage Select pin must be set to a high if bank 0, bank 14 and bank 15 are operating at 2.5V or 3.3V. Otherwise, must be set low if operation is at 1.5V or 1.8V.

*** INIT_B ***
The FPGA drives hits this pin low when:
1) It is in a configuration reset state
2) Is clearing its configuration memory
3) A configuration error has been detected
The FPGA expects a High here after the initialization process to continue with the configuration sequence defined by M[2:0].



*** DXP, DXN ***
Temperature-sensing diode pins. If we are not using it, we can tie them to GND (see UG-475).

*** VREFP, VREFN ***
Because this board is purely digital and we do not intend to use XADX of the FPGA, we use the internal voltage reference. To do this, we connect VREFP and VREFN to analog GND.

*** VP, VN ***
They are a dedicated differential analog input that can be used to measure analog signals surrounded by noise. In this design we are not using them so we tied them to GND.



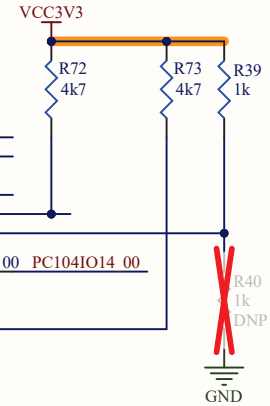
Project:	CRONO_TDC
Sheet Name:	bank0_config.SchDoc
Revision:	A.00
Author:	RODRIGUEZ JULIAN
Reviewer:	
Date:	1/18/2023
Size:	A4

UID

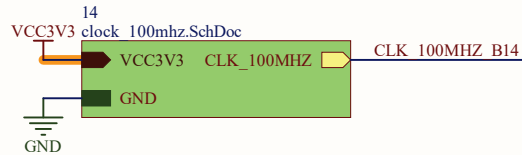
BANK 14

IO_L1P_T0_D00_MOSI_14	K17	QSPI DQ0
IO_L1N_T0_D01_DIN_14	K18	QSPI DQ1
IO_L6P_T0_FCS_B_14	L13	QSPI CS
IO_L2P_T0_D02_14	L14	QSPI DQ2
IO_L3P_T0_DQS_PUDC_B_14	L15	
IO_L3N_T0_DQS_EMCCLK_14	L16	
IO_L4P_T0_D04_14	L18	R71 100 PC104IO14 00
IO_L6N_T0_D08_VREF_14	M13	QSPI DQ3
IO_L2N_T0_D03_14	M14	
IO_L10P_T1_D14_14	M16	
IO_L10N_T1_D15_14	M17	
IO_L4N_T0_D05_14	M18	
IO_L8P_T1_D11_14	N14	
IO_L11P_T1_SRCC_14	N15	
IO_L11N_T1_SRCC_14	N16	
IO_L9P_T1_DQS_14	N17	
IO_L8N_T1_D12_14	P14	
IO_L13P_T2_MRCC_14	P15	
IO_L12P_T1_MRCC_14	P17	CLK 100MHZ B14
IO_L9N_T1_DQS_D13_14	P18	
IO_25_14	R10	
IO_0_14	R11	
IO_L5P_T0_D06_14	R12	
IO_L5N_T0_D07_14	R13	
IO_L13N_T2_MRCC_14	R15	
IO_L15P_T2_DQS_RDWR_B_14	R16	
IO_L12N_T1_MRCC_14	R17	
IO_L7P_T1_D09_14	R18	
IO_L24P_T3_A01_D17_14	T9	
IO_L24N_T3_A00_D16_14	T10	
IO_L19P_T3_A10_D26_14	T11	
IO_L23P_T3_A03_D19_14	T13	
IO_L14P_T2_SRCC_14	T14	
IO_L14N_T2_SRCC_14	T15	
IO_L15N_T2_DQS_DOUT_CSO_B_14	T16	
IO_L7N_T1_D10_14	T18	R66 100 PC104IO14 01
IO_L19N_T3_A09_D25_VREF_14	U11	
IO_L20P_T3_A08_D24_14	U12	
IO_L23N_T3_A02_D18_14	U13	
IO_L22P_T3_A05_D21_14	U14	
IO_L18P_T2_A12_D28_14	U16	
IO_L17P_T2_A14_D30_14	U17	
IO_L17N_T2_A13_D29_14	U18	R67 100 PC104IO14 02
IO_L21P_T3_DQS_14	V10	
IO_L21N_T3_DQS_A06_D22_14	V11	
IO_L20N_T3_A07_D23_14	V12	R98 100 PRSNT M2C L
IO_L22N_T3_A04_D20_14	V14	
IO_L16P_T2_CSI_B_14	V15	R106 100 PC104IO14 03
IO_L16N_T2_A15_D31_14	V16	R70 100 PC104IO14 04
IO_L18N_T2_A11_D27_14	V17	R69 100 PC104IO14 05

XC7A35T-2CSG324I

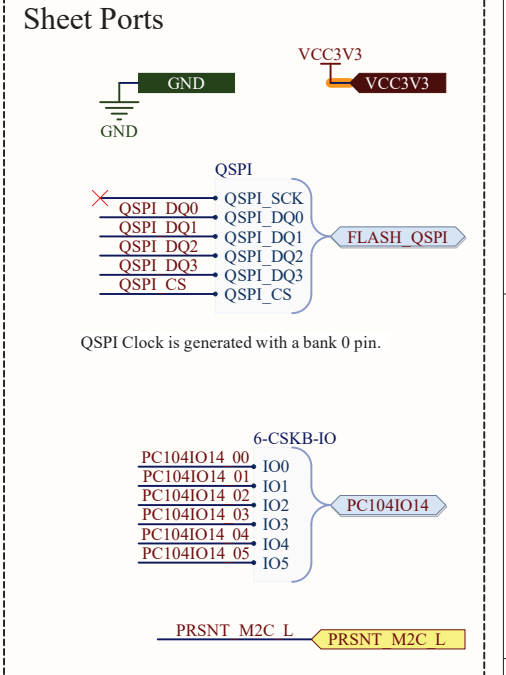


Clock Source



*** Flash QSPI Interface ***
 This FPGA can read and write from a QSPI flash memory using a special set of pins of bank 14.
 Xilinx recommends to use 4k7 pull-ups in D02 and D03 (see UG470).

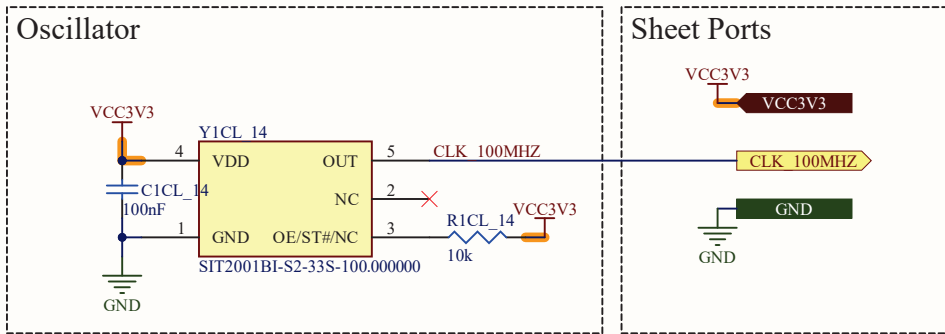
*** PUDC ***
 Pull-Up During Configuration (IO_L3P_T0_DQS_PUDC_B_14).
 - When PUDC_B is tied low the internal pull-ups are enabled on each SelectIO pin after power-up and during configuration.
 - When PUDC_B is tied high the internal pull-ups are disabled on each SelectIO pin.




QSPI Clock is generated with a bank 0 pin.



Project:	CRONO_TDC
Sheet Name:	bank14.SchDoc
Revision:	A.00
Author:	RODRIGUEZ JULIAN
Reviewer:	
Date:	1/18/2023
Size:	A4
	Sheet 10 of 24



 UNSAM	
Project:	CRONO_TDC
Sheet Name:	clock_100mhz.SchDoc
Revision:	A.00
Author:	RODRIGUEZ JULIAN
Reviewer:	
Date:	
Size: A4	Sheet 11 of 24

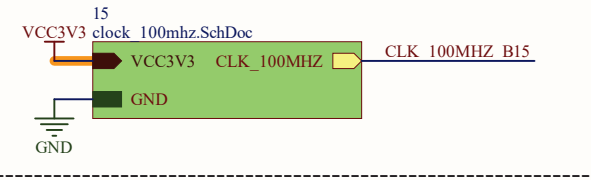
U1E

BANK 15

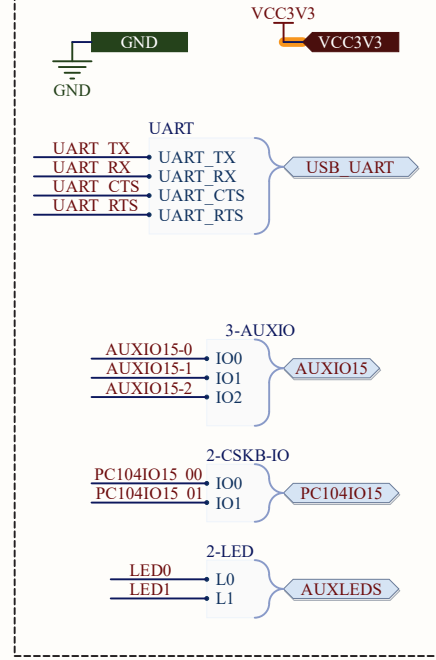
IO_L4N_T0_15	A11	R88	100	AUXIO15-2
IO_L9P_T1_DQS_AD3P_15	A13			LED1
IO_L9N_T1_DQS_AD3N_15	A14			LED0
IO_L8P_T1_AD10P_15	A15			
IO_L8N_T1_AD10N_15	A16	R87	100	AUXIO15-1
IO_L10N_T1_AD11N_15	A18	R86	100	AUXIO15-0
IO_L4P_T0_15	B11			
IO_L3N_T0_DQS_AD1N_15	B12			
IO_L2P_T0_AD8P_15	B13			
IO_L2N_T0_AD8N_15	B14			
IO_L7P_T1_AD2P_15	B16			
IO_L7N_T1_AD2N_15	B17			
IO_L10P_T1_AD11P_15	B18	R81	100	UART TX
IO_L3P_T0_DQS_AD1P_15	C12			
IO_L1N_T0_AD0N_15	C14			
IO_L12N_T1_MRCC_15	C15			
IO_L20P_T3_A20_15	C16			
IO_L20N_T3_A19_15	C17			
IO_L6P_T0_15	D12			
IO_L6N_T0_VREF_15	D13			
IO_L1P_T0_AD0P_15	D14			
IO_L12P_T1_MRCC_15	D15			CLK 100MHZ B15
IO_L16N_T2_A27_15	D17			
IO_L21N_T3_DQS_A18_15	D18	R84	100	UART RTS
IO_L11P_T1_SRCC_15	E15			
IO_L11N_T1_SRCC_15	E16			
IO_L16P_T2_A28_15	E17			
IO_L21P_T3_DQS_15	E18	R82	100	UART RX
IO_L5P_T0_AD9P_15	F13			
IO_L5N_T0_AD9N_15	F14			
IO_L14P_T2_SRCC_15	F15			
IO_L14N_T2_SRCC_15	F16			
IO_L22N_T3_A16_15	F18	R83	100	UART CTS
IO_0_15	G13			
IO_L15N_T2_DQS_ADV_B_15	G14			
IO_L13N_T2_MRCC_15	G16			
IO_L18N_T2_A23_15	G17			
IO_L22P_T3_A17_15	G18	R68	100	PC104IO15_00
IO_L15P_T2_DQS_15	H14			
IO_L19N_T3_A21_VREF_15	H15			
IO_L13P_T2_MRCC_15	H16			
IO_L18P_T2_A24_15	H17			
IO_L17N_T2_A25_15	J13			
IO_L19P_T3_A22_15	J14			
IO_L24N_T3_RS0_15	J15			
IO_L23P_T3_FOE_B_15	J17			
IO_L23N_T3_FWE_B_15	J18	R65	100	PC104IO15_01
IO_L17P_T2_A26_15	K13			
IO_L24P_T3_RS1_15	K15			
IO_25_15	K16			

XC7A35T-2CSG324I

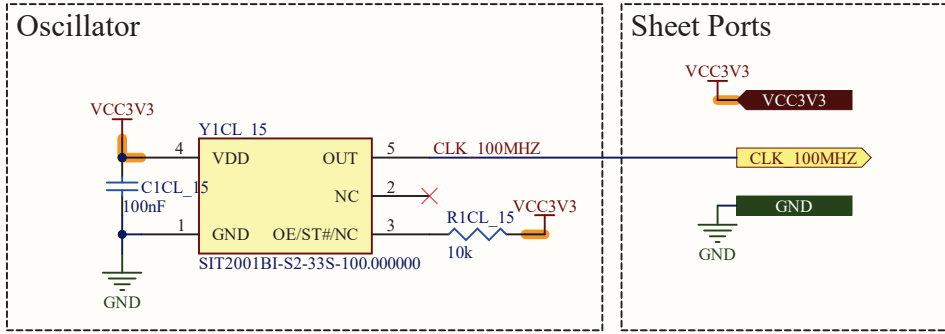
Clock Source




Sheet Ports



Project:	CRONO_TDC
Sheet Name:	bank15.SchDoc
Revision:	A.00
Author:	RODRIGUEZ JULIAN
Reviewer:	
Date:	
Size:	A4
	Sheet 12 of 24



 UNSAM	
Project:	CRONO_TDC
Sheet Name:	clock_100mhz.SchDoc
Revision:	A.00
Author:	RODRIGUEZ JULIAN
Reviewer:	
Date:	
Size: A4	Sheet 13 of 24

U1B

BANK 16

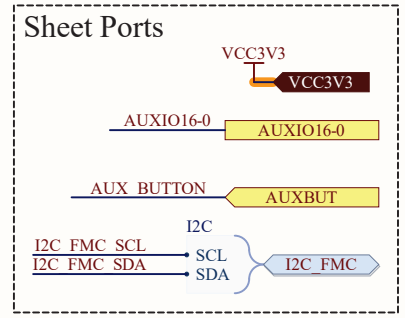
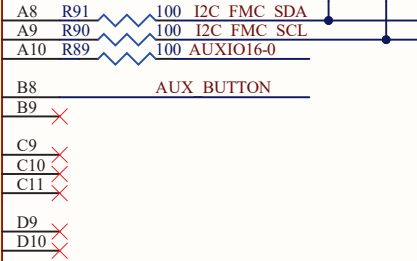
IO_L12N_T1_MRCC_16
 IO_L14N_T2_SRCC_16
 IO_L14P_T2_SRCC_16

IO_L12P_T1_MRCC_16
 IO_L11N_T1_SRCC_16

IO_L11P_T1_SRCC_16
 IO_L13N_T2_MRCC_16
 IO_L13P_T2_MRCC_16

IO_L6N_T0_VREF_16
 IO_L19N_T3_VREF_16

XC7A35T-2CSG324I



Project:	CRONO_TDC
Sheet Name:	bank16.SchDoc
Revision:	A.00
Author:	RODRIGUEZ JULIAN
Reviewer:	
Date:	1/18/2023
Size: A4	Sheet 14 of 24

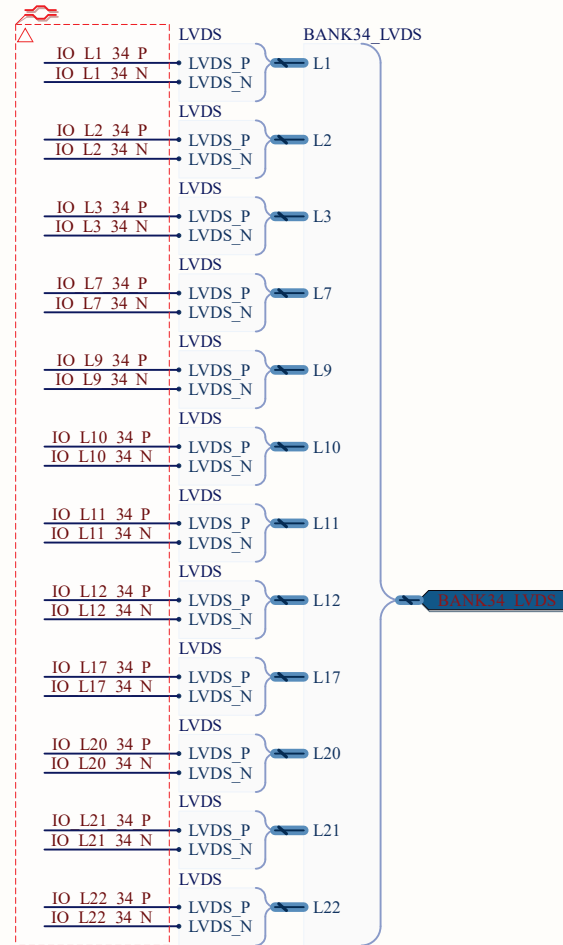
U1A

BANK 34

IO_L2P_T0_34	K3	IO L2 34 P
IO_L5P_T0_34	K5	
IO_0_34	K6	
IO_L1P_T0_34	L1	IO L1 34 P
IO_L2N_T0_34	L3	IO L2 34 N
IO_L5N_T0_34	L4	
IO_L6N_T0_VREF_34	L5	
IO_L6P_T0_34	L6	
IO_L1N_T0_34	M1	IO L1 34 N
IO_L4N_T0_34	M2	
IO_L4P_T0_34	M3	
IO_L16P_T2_34	M4	
IO_L18P_T2_34	M6	
IO_L3N_T0_DQS_34	N1	IO L3 34 N
IO_L3P_T0_DQS_34	N2	IO L3 34 P
IO_L16N_T2_34	N4	
IO_L13P_T2_MRCC_34	N5	
IO_L18N_T2_34	N6	
IO_L15P_T2_DQS_34	P2	
IO_L14N_T2_SRCC_34	P3	
IO_L14P_T2_SRCC_34	P4	
IO_L13N_T2_MRCC_34	P5	
IO_L17P_T2_34	R1	IO L17 34 P
IO_L15N_T2_DQS_34	R2	
IO_L11P_T1_SRCC_34	R3	IO L11 34 P
IO_L19N_T3_VREF_34	R5	
IO_L19P_T3_34	R6	
IO_L23P_T3_34	R7	
IO_L24P_T3_34	R8	
IO_L17N_T2_34	T1	IO L17 34 N
IO_L11N_T1_SRCC_34	T3	IO L11 34 N
IO_L12N_T1_MRCC_34	T4	IO L12 34 N
IO_L12P_T1_MRCC_34	T5	IO L12 34 P
IO_L23N_T3_34	T6	
IO_L24N_T3_34	T8	
IO_L7P_T1_34	U1	IO L7 34 P
IO_L9P_T1_DQS_34	U2	IO L9 34 P
IO_L8N_T1_34	U3	
IO_L8P_T1_34	U4	
IO_L22N_T3_34	U6	IO L22 34 N
IO_L22P_T3_34	U7	IO L22 34 P
IO_25_34	U8	
IO_L21P_T3_DQS_34	U9	IO L21 34 P
IO_L7N_T1_34	V1	IO L7 34 N
IO_L9N_T1_DQS_34	V2	IO L9 34 N
IO_L10N_T1_34	V4	IO L10 34 N
IO_L10P_T1_34	V5	IO L10 34 P
IO_L20N_T3_34	V6	IO L20 34 N
IO_L20P_T3_34	V7	IO L20 34 P
IO_L21N_T3_DQS_34	V9	IO L21 34 N

XC7A35T-2CSG324I

Sheet Ports



Project:	CRONO_TDC
Sheet Name:	bank34.SchDoc
Revision:	A.00
Author:	RODRIGUEZ JULIAN
Reviewer:	
Date:	1/18/2023
Size:	A4
Sheet 15 of 24	

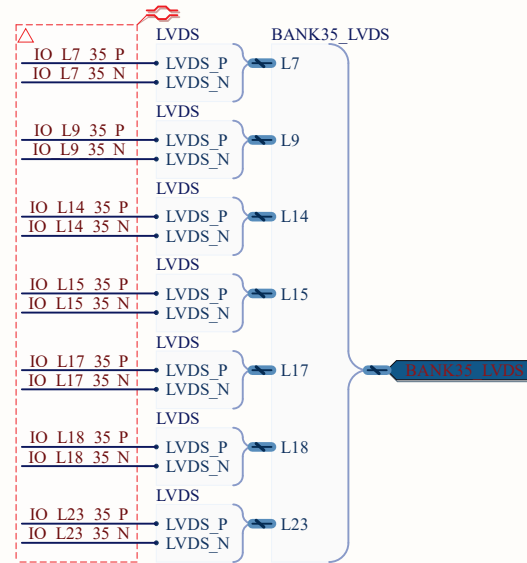
UIF

BANK 35

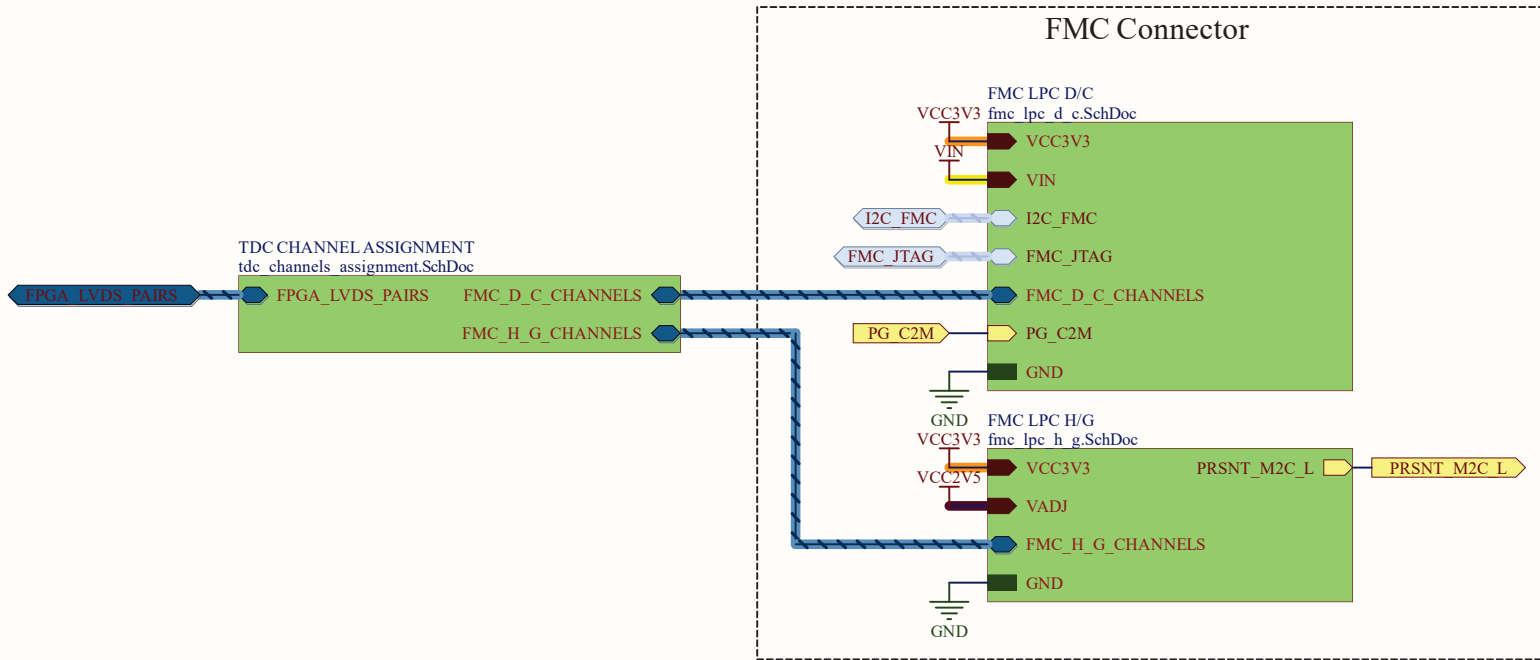
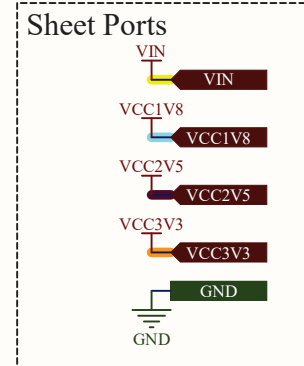
IO_L9N_T1_DQS_AD7N_35	A1	IO L9 35 N
IO_L8N_T1_AD14N_35	A3	X
IO_L8P_T1_AD14P_35	A4	X
IO_L3N_T0_DQS_AD5N_35	A5	X
IO_L3P_T0_DQS_AD5P_35	A6	X
IO_L9P_T1_DQS_AD7P_35	B1	IO L9 35 P
IO_L10N_T1_AD15N_35	B2	X
IO_L10P_T1_AD15P_35	B3	X
IO_L7N_T1_AD6N_35	B4	IO L7 35 N
IO_L2N_T0_AD12N_35	B6	X
IO_L2P_T0_AD12P_35	B7	X
IO_L16N_T2_35	C1	X
IO_L16P_T2_35	C2	X
IO_L7P_T1_AD6P_35	C4	IO L7 35 P
IO_L1N_T0_AD4N_35	C5	X
IO_L1P_T0_AD4P_35	C6	X
IO_L4N_T0_35	C7	X
IO_L14N_T2_SRCC_35	D2	IO L14 35 N
IO_L12N_T1_MRCC_35	D3	X
IO_L11N_T1_SRCC_35	D4	X
IO_L11P_T1_SRCC_35	D5	X
IO_L6N_T0_VREF_35	D7	X
IO_L4P_T0_35	D8	X
IO_L18N_T2_35	E1	IO L18 35 N
IO_L14P_T2_SRCC_35	E2	IO L14 35 P
IO_L12P_T1_MRCC_35	E3	X
IO_L5N_T0_AD13N_35	E5	X
IO_L5P_T0_AD13P_35	E6	X
IO_L6P_T0_35	E7	X
IO_L18P_T2_35	F1	IO L18 35 P
IO_L13N_T2_MRCC_35	F3	X
IO_L13P_T2_MRCC_35	F4	X
IO_0_35	F5	X
IO_L19N_T3_VREF_35	F6	X
IO_L17N_T2_35	G1	IO L17 35 N
IO_L15N_T2_DQS_35	G2	IO L15 35 N
IO_L20N_T3_35	G3	X
IO_L20P_T3_35	G4	X
IO_L19P_T3_35	G6	X
IO_L17P_T2_35	H1	IO L17 35 P
IO_L15P_T2_DQS_35	H2	IO L15 35 P
IO_L21N_T3_DQS_35	H4	X
IO_L24N_T3_35	H5	X
IO_L24P_T3_35	H6	X
IO_L22N_T3_35	J2	X
IO_L22P_T3_35	J3	X
IO_L21P_T3_DQS_35	J4	X
IO_25_35	J5	X
IO_L23N_T3_35	K1	IO L23 35 N
IO_L23P_T3_35	K2	IO L23 35 P

XC7A35T-2CSG324I

Sheet Ports

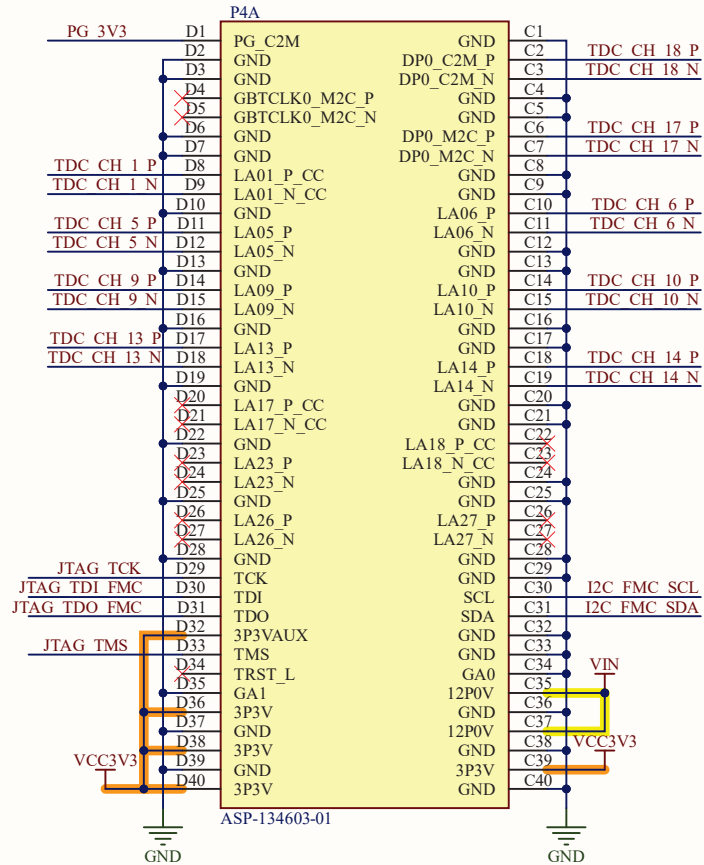


Project:	CRONO_TDC
Sheet Name:	bank35.SchDoc
Revision:	A.00
Author:	RODRIGUEZ JULIAN
Reviewer:	
Date:	
Size:	A4
Sheet 16 of 24	



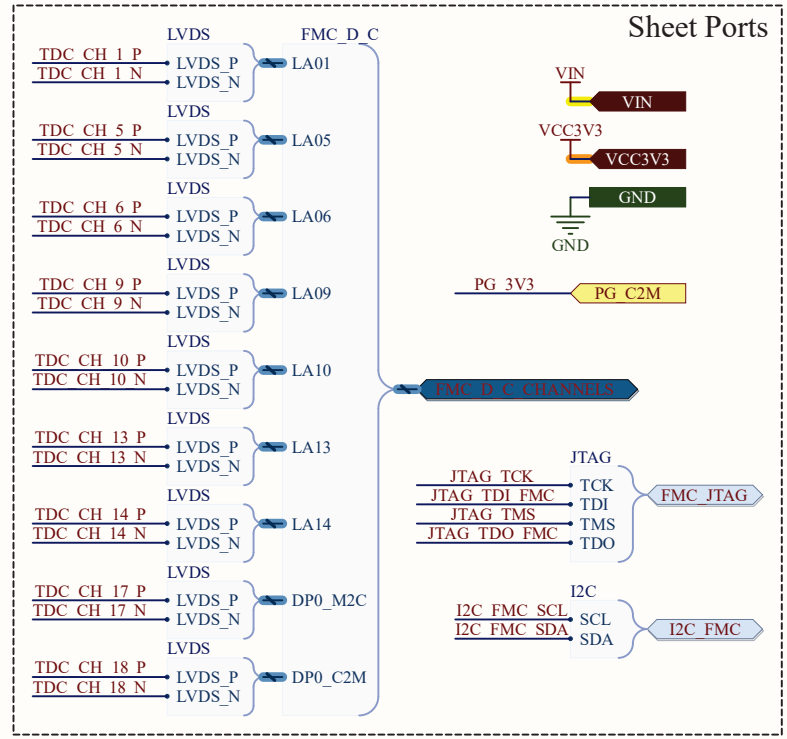
Project:	CRONO_TDC
Sheet Name:	fmc_connector.SchDoc
Revision:	A.00
Author:	RODRIGUEZ JULIAN
Reviewer:	
Date:	1/18/2023
Size:	A4

FMC Connector D/C



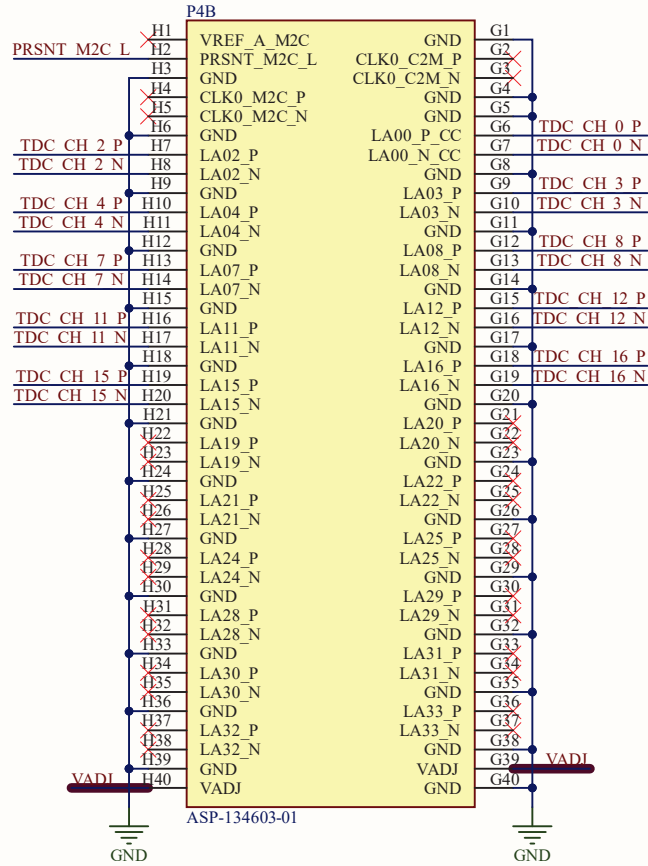
* TRST_L *
This JTAG signal is not used by Xilinx tools. Because this, is unconnected here.

* GA[1:0] *
This PCB is prepared to connect only one FMC module. Because this, the GA pins are connected directly to GND.

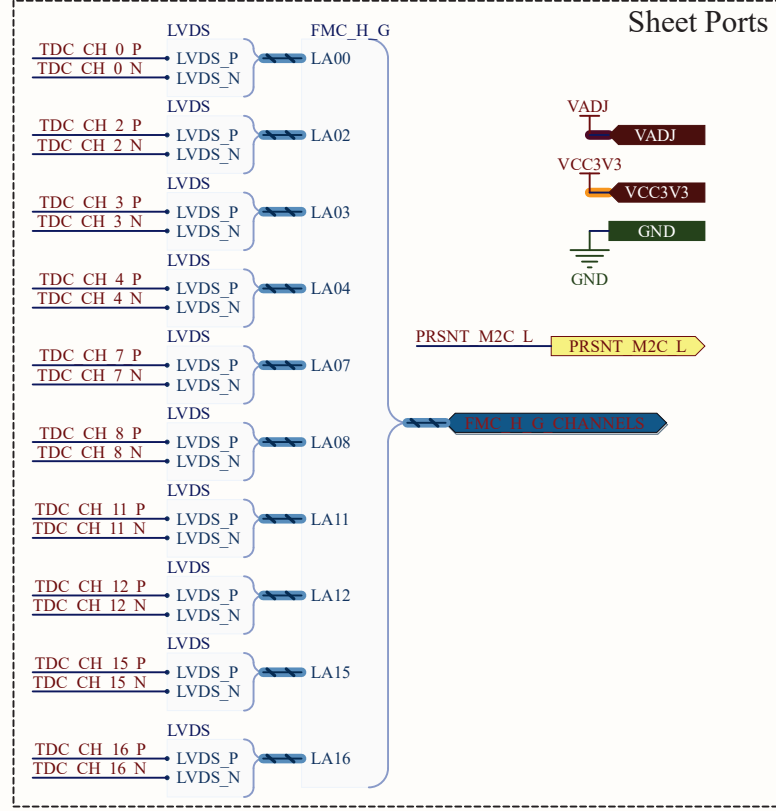


Project:	CRONO_TDC
Sheet Name:	<i>fmc_lpc_d_c.SchDoc</i>
Revision:	A.00
Author:	RODRIGUEZ JULIAN
Reviewer:	
Date:	
Size:	A4
Sheet 18 of 24	

FMC Connector D/C

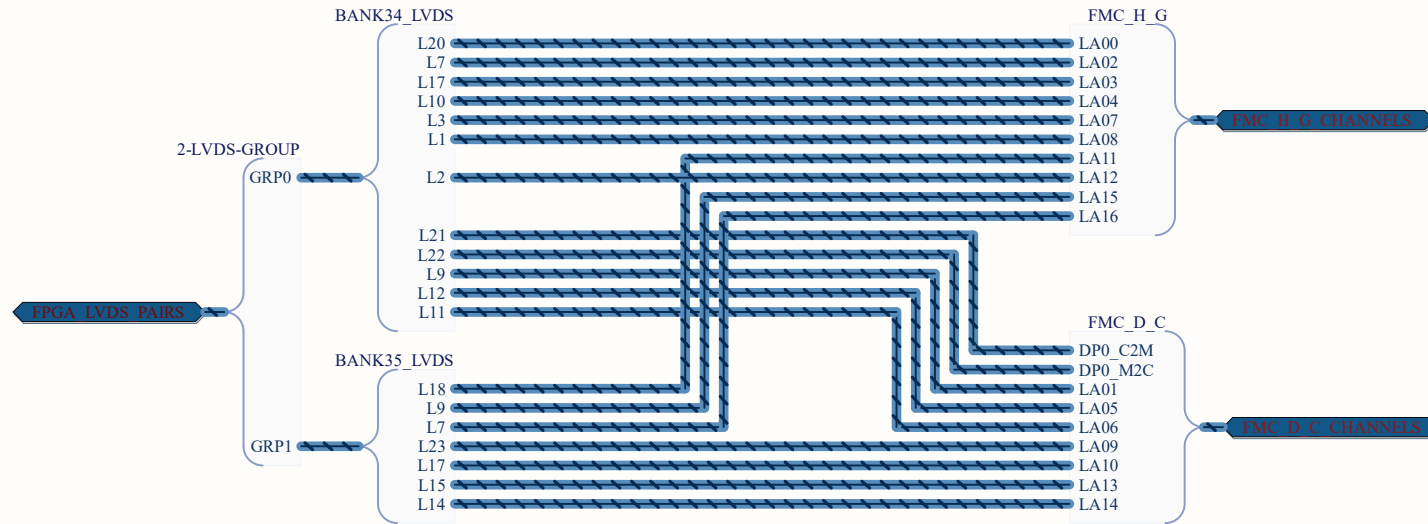


Sheet Ports



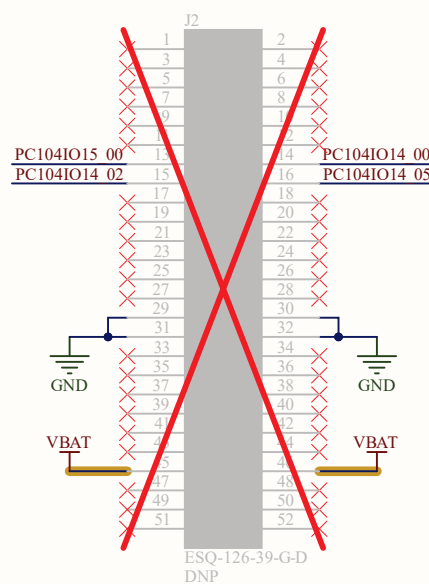
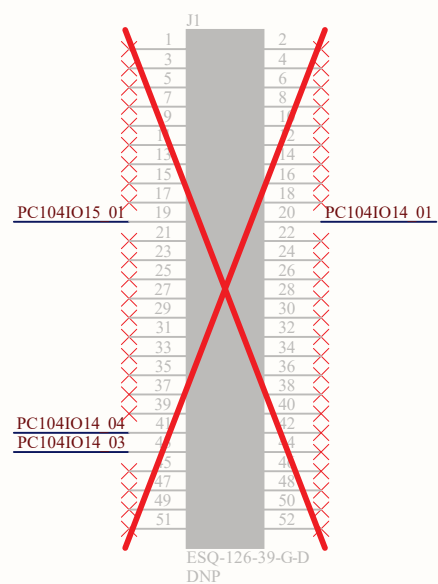
Project:	CRONO_TDC
Sheet Name:	fmc_lpc_h_g.SchDoc
Revision:	A.00
Author:	RODRIGUEZ JULIAN
Reviewer:	
Date:	
Size:	A4

TDC Channels Assignment

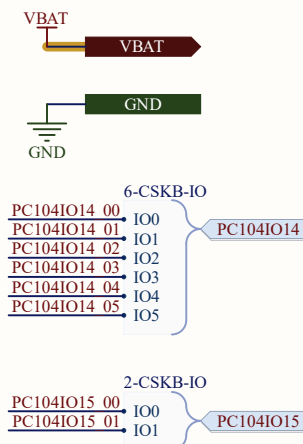


Project:	CRONO_TDC
Sheet Name:	tdc_channels_assignment.SchDoc
Revision:	A.00
Author:	RODRIGUEZ JULIAN
Reviewer:	
Date:	
Size:	A4
Sheet 20 of 24	

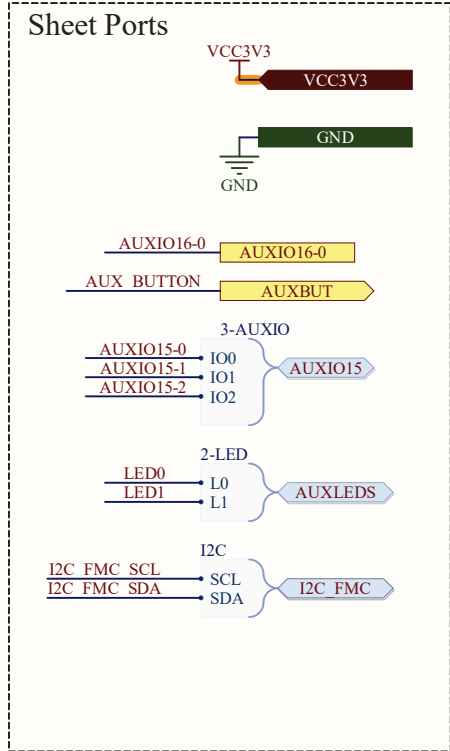
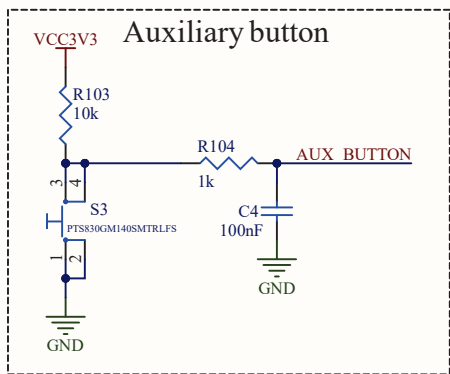
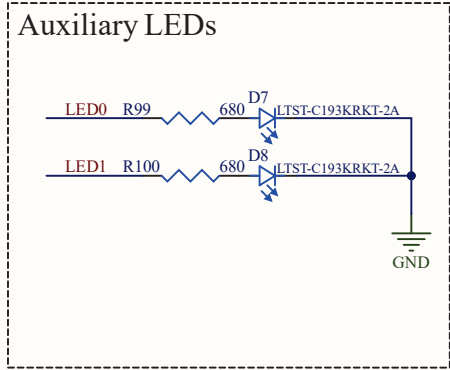
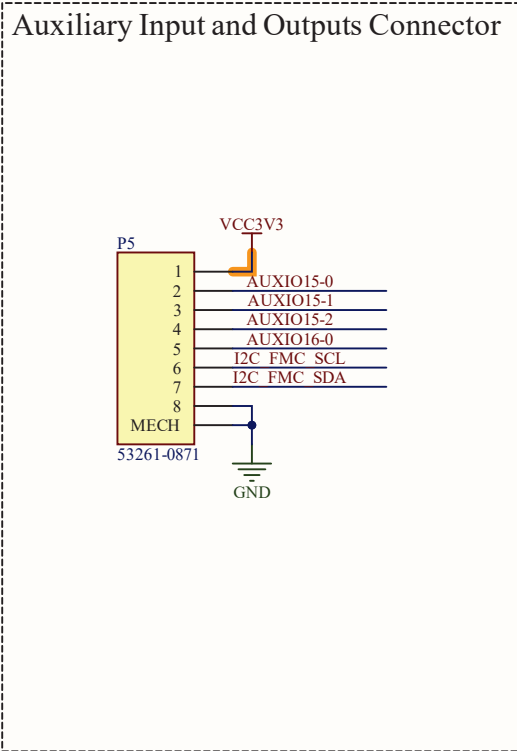
Cubesat Kit Bus Connector



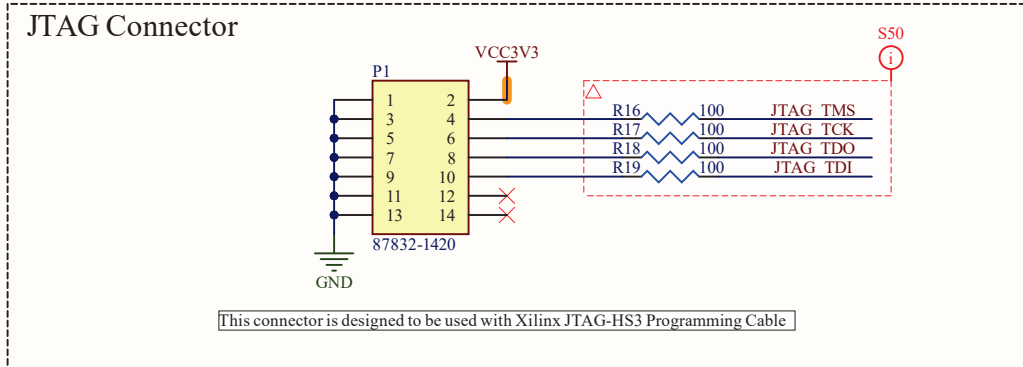
Sheet Ports



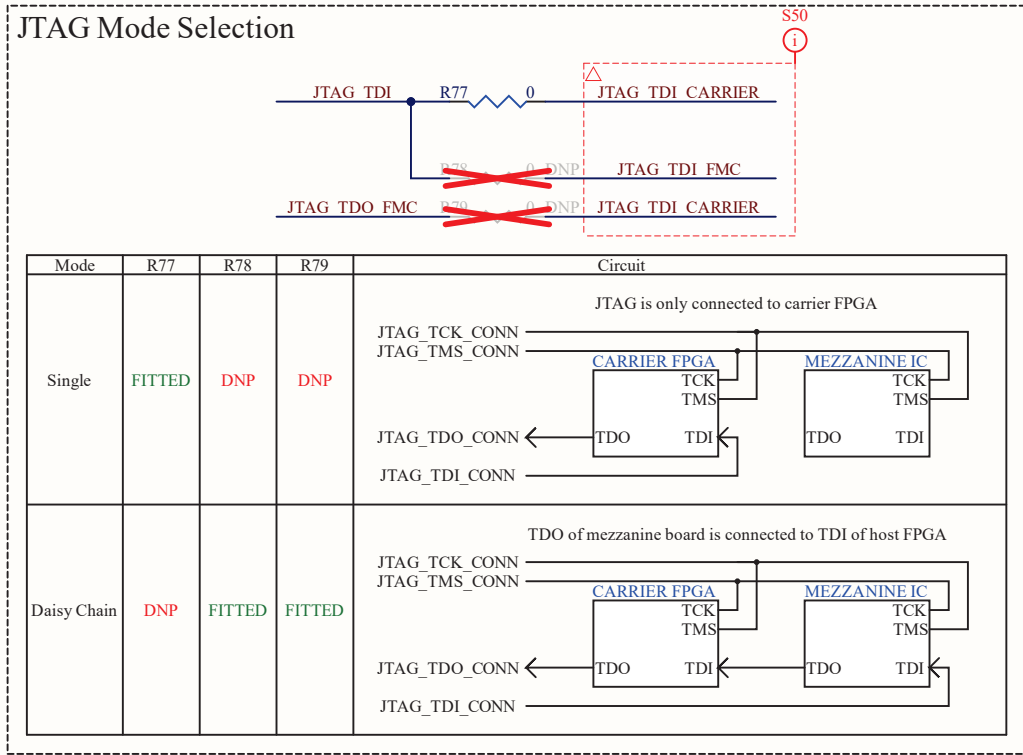
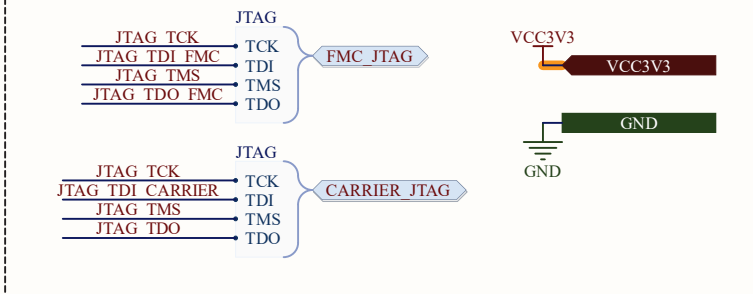
Project:	CRONO_TDC
Sheet Name:	cubesat_kit_bus.SchDoc
Revision:	A.00
Author:	RODRIGUEZ JULIAN
Reviewer:	
Date:	
Size:	A4
Sheet 21 of 24	



Project:	CRONO_TDC
Sheet Name:	aux_io.SchDoc
Revision:	A.00
Author:	RODRIGUEZ JULIAN
Reviewer:	
Date:	1/18/2023
Size:	A4

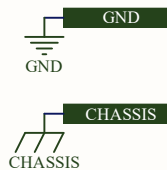


Sheet Ports

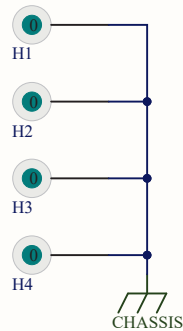


Project:	CRONO_TDC
Sheet Name:	jtag.SchDoc
Revision:	A.00
Author:	RODRIGUEZ JULIAN
Reviewer:	
Date:	1/18/2023
Size:	A4

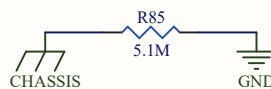
Sheet ports



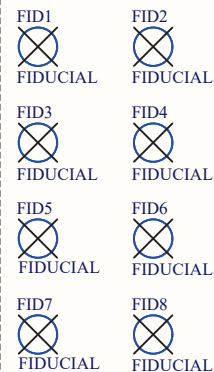
PC-104 Mounting Holes



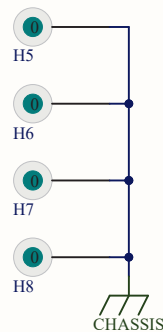
GND-Chassis Connection



Fiducials



Vita 57.1 Mounting Holes



Project:	CRONO_TDC
Sheet Name:	mechanical.SchDoc
Revision:	A.00
Author:	RODRIGUEZ JULIAN
Reviewer:	
Date:	1/18/2023
Size: A4	Sheet 24 of 24

A.2. Instrucciones de fabricación del PCB

A continuación se encuentran las instrucciones de fabricación del PCB. En las mismas se puede visualizar el *stackup* y otras detalles de la fabricación.

Board specification

Parameter	Specification
Layer Count	8
Solder Mask Color	BLUE
Silkscreen Color	WHITE
Surface Finishing	ENIG
Fabrication Specification	IPC-6012 CLS3
Assembly Specification	IPC-610 CLS3
Solder Material	Sn62 or Sn63 (Tin/Lead)
Board Outline	Mechanical 2
Tented Vias	Yes
Solder Mask Material	PSR-4000LDI or similar with TML<1% and CVC<0.10%
Board Edge Finish	Routed

Controlled Impedance

Target Impedance [Ω]	Layer	Trace Width [mil]	GAP [mil]	Reference Layers
50 \pm 10%	L1-SIG TOP	10	-	L2-GND
50 \pm 10%	L3-PWR/SIG	5	-	L4-GND
50 \pm 10%	L5-PWR/SIG	5	-	L6-GND
50 \pm 10%	L8-SIG TOP	10	-	L7-GND
90 \pm 10%	L1-SIG TOP	10	6.5	L2-GND
90 \pm 10%	L8-SIG BOT	10	6.5	L7-GND
100 \pm 10%	L1-SIG TOP	5	4	L2-GND
100 \pm 10%	L5-PWR/SIG	5	8.5	L6-GND
100 \pm 10%	L8-SIG BOT	5	4	L7-GND

Layer Stack Legend

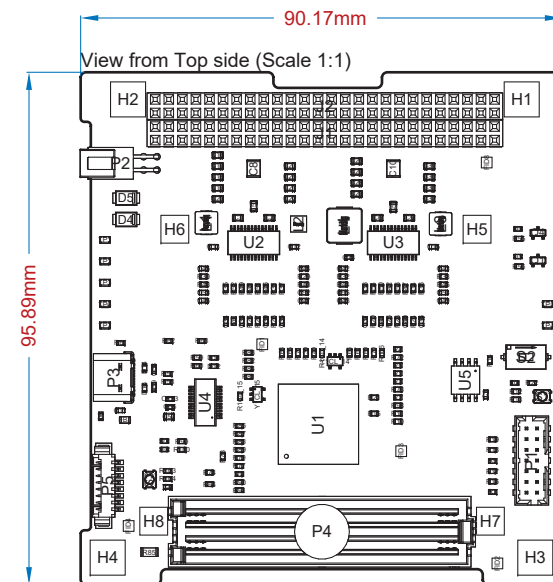
Material	Layer	Thickness	Dielectric Material	Type
	Top Overlay			Legend
Surface Material	Top Solder	0.010mm	Solder Resist	Solder Mask
Nickel, Gold	Top Surface Finish	0.017mm		Surface Finish
Copper	L1-SIG TOP	0.018mm		Signal
Prepreg		0.180mm	PP 0.18mm(7628)	Dielectric
Copper	L2-GND	0.035mm		Signal
Core		0.300mm	Core 0.3mm	Dielectric
Copper	L3-PWR/SIG	0.035mm		Signal
Prepreg		0.180mm	PP 0.18mm(7628)	Dielectric
Copper	L4-GND	0.035mm		Signal
Core		0.300mm	Core 0.3mm	Dielectric
Copper	L5-PWR/SIG	0.035mm		Signal
Prepreg		0.180mm	PP 0.18mm(7628)	Dielectric
Copper	L6-GND	0.035mm		Signal
Core		0.300mm	Core 0.3mm	Dielectric
Copper	L7-GND	0.035mm		Signal
Prepreg		0.180mm	PP 0.18mm(7628)	Dielectric
Copper	L8-SIG BOT	0.018mm		Signal
Nickel, Gold	Bottom Surface Finish	0.017mm		Surface Finish
Surface Material	Bottom Solder	0.010mm	Solder Resist	Solder Mask
	Bottom Overlay			Legend

Total thickness: 1.920mm

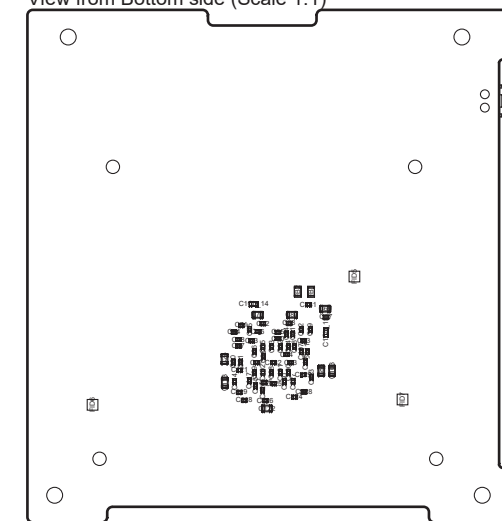
Notes:

1 Be free to modify solder mask expansion of components that was set to zero (P3 and U4).

2 L1-SIG TOP and L8-SIG BOT layer thickness is copper FINISHED (Copper + Plating).



View from Bottom side (Scale 1:1)



B. Diseño de la fuente de alimentación de la FPGA

En este anexo se encuentra el diseño en detalle de la fuente de alimentación de la FPGA. La FPGA necesita 4 rieles de tensión: V_{CCINT} y V_{CCBRAM} (1 V), V_{CCAUX} (1,8 V), V_{CCO2V5} (2,5 V) y V_{CCO3V3} (3,3 V) y el fabricante recomienda cumplir una secuencia específica de encendido de estos rieles para minimizar la magnitud de la corriente de arranque (esto mismo se describe en más detalle en la nota de aplicación [65]). Primero debe encenderse V_{CCINT} , luego V_{CCBRAM} , luego V_{CCAUX} y por último las alimentaciones de los bancos (V_{CCO}). Pero, si V_{CCINT} posee el mismo valor que V_{CCBRAM} ambas pueden ser encendidas en simultáneo (lo mismo sucede con V_{CCAUX} y V_{CCO}).

El consumo de la FPGA es determinado por varios factores, entre ellos, la frecuencia de reloj interna y el uso de los puertos de IO. El fabricante brinda información y herramientas para dimensionar el consumo de la FPGA en estadios tempranos del diseño.

En la hoja de datos [33] se encuentran los valores de corriente de reposo de cada una de las fuentes de alimentación, los cuales brindan una idea de la magnitud del consumo mínimo esperado que puede poseer la FPGA. Estos consumos están especificados a la tensión nominal, una temperatura de juntura de 85°C, para FPGAs desconfiguradas, sin carga alguna en sus puertos, sin *pull-ups* activos y con sus pines en alta impedancia y flotando. Por otro lado, Vivado posee una herramienta llamada *Report Power* la cual, si es ejecutada luego de que haya sucedido el *route and placement*, nos brinda información detallada del consumo esperado de la FPGA. Se ejecutó dicho análisis con el diseño del TDC con 18 canales de entrada sincronizados con un reloj de 100 MHz y un generador de patrones sincronizado con un reloj de 400 MHz. En el diseño se simuló un peor caso, donde todos los elementos de la FPGA trabajan al 100 % de la frecuencia de reloj. Esto es una estimación grosera para entender el consumo que podría poseer la FPGA una vez configurada. El resultado de dicho análisis se puede ver en las Figuras B.1 y B.2.

Power Supply				
Supply Source	Voltage (V)	Total (A)	Dynamic (A)	Static (A)
Vccint	1.000	0.433	0.421	0.012
Vccaux	1.800	0.159	0.146	0.013
Vcco33	3.300	0.006	0.005	0.001
Vcco25	2.500	0.017	0.016	0.001
Vcco18	1.800	0.000	0.000	0.000
Vcco15	1.500	0.000	0.000	0.000
Vcco135	1.350	0.000	0.000	0.000
Vcco12	1.200	0.000	0.000	0.000
Vccaux_io	1.800	0.000	0.000	0.000
Vccbram	1.000	0.005	0.004	0.001
MGTAVcc	1.000	0.000	0.000	0.000
MGTAVtt	1.200	0.000	0.000	0.000
Vccadc	1.800	0.020	0.000	0.020

Figura B.1: Consumo estimado por riel de tensión de una implementación del *core* con 18 canales de entrada sincronizados con un reloj de 100 MHz y un generador de patrones sincronizado con un reloj de 400 MHz.

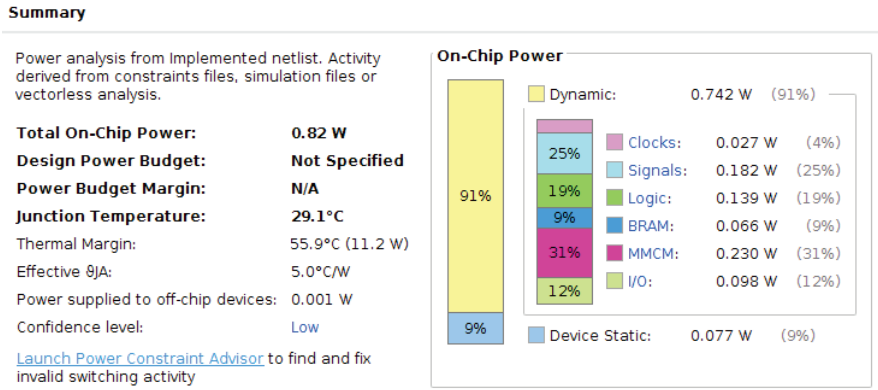


Figura B.2: Porcentaje de consumo de cada elemento de la FPGA de una implementación del *core* con 18 canales de entrada sincronizados con un reloj de 100 MHz y un generador de patrones sincronizado con un reloj de 400 MHz.

B.1. Diseño de la fuente de alimentación

Se optó por un diseño similar al que puede ser observado en placas de desarrollo comerciales que involucran FPGAs de la misma familia [45] [46]. En dichas placas se utilizan reguladores *switching* para generar las tensiones de alimentación de la FPGA.

Como se menciona en [66], las FPGAs requieren que sus tensiones de alimentación crezcan con una pendiente suave hasta su valor de estado estacionario. Esto se debe a que las FPGAs pueden implementar circuitos denominados *power-on-reset circuits*, los cuales reinician otros circuitos sensibles internos luego de ser energizada para que inicien en un estado bien conocido. Con respecto a la FPGA utilizada, en su guía de configuración [40] el fabricante aclara que las alimentaciones de la FPGA deben crecer en forma de rampa monótona y en su hoja de datos [33] se encuentra que estos rieles deben llegar al 90 % de su estado estacionario entre 200 μs y 50 ms. Hay que tener en cuenta que las FPGAs pueden poseer corrientes de arranque mucho mayores a la corriente de estado estacionario, dado que, al energizarse deben cargar las capacitancias de los millones de componentes internos. Además, cada riel de alimentación de la FPGA posee una gran cantidad de capacitores de desacople y de *bypass* para poder suministrar a la FPGA la energía necesaria durante transitorios de consumos elevados, lo cual sucede cuando ocurren una gran cantidad de conmutaciones simultáneas. Por lo tanto, el regulador utilizado debe suministrar una corriente de arranque alta y mantener una rampa creciente de tensión con una cierta pendiente.

Una manera de proveer esta rampa y minimizar la corriente de arranque de la FPGA es que los reguladores utilizados posean un encendido suave. En el mercado, los reguladores conmutados poseen circuitos que permiten que la tensión de salida crezca monótonamente en forma de rampa, llamados *soft start circuits*. Es por ello que es una ventaja utilizar reguladores conmutados frente a los reguladores lineales para alimentar FPGAs, dado que son pocos los reguladores lineales que poseen encendido suave.

La arquitectura finalmente utilizada se puede visualizar en la Figura B.3, donde los reguladores deben poder proveer por lo menos el doble de la corriente obtenida en la estimación inicial de consumo (ver Figuras B.1 y B.2). Se puede visualizar en dicha arquitectura que la señal de *power good* del regulador de 1 V habilita al regulador de 1,8 V. Luego, la señal de *power good* del regulador de 1,8 V habilita al regulador de 2,5 V y 3,3 V. Con esto se obtiene el secuenciado en las alimentaciones recomendado por el fabricante.

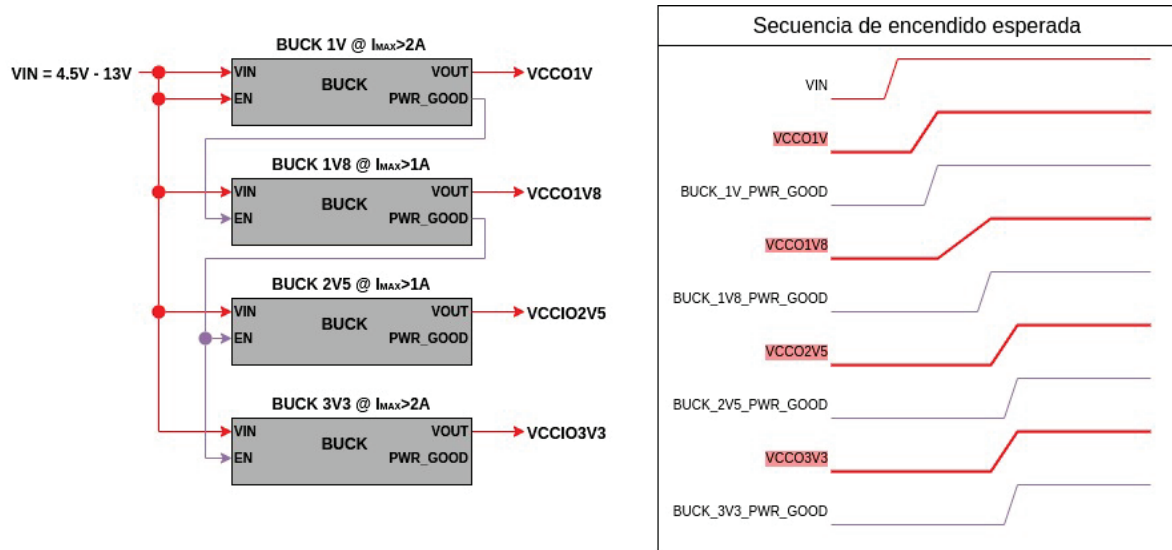


Figura B.3: Arquitectura del circuito de alimentación para generar las tensiones necesarias para la FPGA.

Los requerimientos de diseño de estas fuentes se desprenden de los requerimientos de estabilidad de la FPGA y de los requerimientos generales de la placa, estos son listados en la Tabla B.1.

Todas las fuentes se diseñaron para soportar una tensión de entrada entre 4,5 V y 13 V, aceptando un *ripple* máximo de entrada (ΔV_{IN}) de 50 mV.

El *ripple* máximo en la tensión de salida (ΔV_{OUT}) de cada fuente se determinó como la mitad de la variación de tensión de alimentación especificada para la FPGA. Por ejemplo, como el riel de 3,3 V de la FPGA permite tensiones entre 3,135 V y 3,465 V, entonces la tensión de *ripple* máxima se calculó como $\frac{3,465\text{ V} - 3,135\text{ V}}{2} = 165\text{ mV}$.

Las corrientes máximas ($I_{OUT(MAX)}$) se pensaron en función de las corrientes que reportó la herramienta de Vivado. Se determinó que los reguladores donde se espera mayor consumo posean, al menos, una corriente de salida máxima de 2 A y en los que se espera menos posean una de 1 A.

Tabla B.1

Requerimientos de diseño de las fuentes Buck.

Parámetro	Riel de tensión	Valor
Rango de tensión de entrada	1 V, 1,8 V, 2,5 V y 3,3 V	4,5 V - 13 V
Tensión de <i>ripple</i> de entrada (ΔV_{IN})	1 V, 1,8 V, 2,5 V y 3,3 V	50mV
Tensión de <i>ripple</i> de salida maximo (ΔV_{OUT})	1 V	50 mV
	1,8 V	90 mV
	2,5 V	125 mV
	3,3 V	165 mV
Corriente de salida máxima ($I_{OUT(MAX)}$)	1 V y 3,3 V	> 2 A
	1,8 V y 2,5 V	> 1 A

Se eligió el chip LTC3633 [67] para implementar dichas fuentes *step down*, el cual es utilizado en la placa de desarrollo Basys 3 [46]. Este integrado posee dos reguladores Buck, los cuales pueden entregar hasta 3 A cada uno y admite una tensión de entrada entre 3,6 V y 20 V. Dado que cada regulador puede entregar hasta 3 A cada uno, se lo diseñó para que entregue la máxima corriente posible. Estos chips incluyen una señal de *power good* y un pin de habilitación en cada canal, los cuales se utilizaron para implementar la secuencia de encendido ilustrada en la Figura B.3.

Se tomó como referencia de diseño el circuito provisto en la hoja de datos del integrado [67]. En las hojas *power_supply.SchDoc*, *power_supply-1v-1v8.SchDoc* y *power_supply-2v5-3v3.SchDoc* (ver

Anexo A.1) se puede observar el circuito diseñado con las fuentes de alimentación. En los siguientes apartados de este anexo se detalla el procedimiento de diseño de las mismas.

B.1.1. Frecuencia de conmutación

Este regulador ofrece la posibilidad de utilizar una red de compensación interna para reducir la cantidad de componentes y agilizar el diseño, pero el fabricante recomienda dicha red solamente para frecuencias de conmutación mayores a 1 MHz . En la nota de aplicación [68] se discute como la frecuencia de un regulador *step down* afecta al rendimiento del mismo. En particular, la frecuencia de conmutación afecta al *ripple* de salida, a la eficiencia, a la respuesta transitoria y al tamaño de los componentes externos (inductor y capacitores).

Se pueden identificar dos grandes tipos de pérdidas de energía en un regulador *switching*: una debida a la conmutación de los transistores (pérdidas de Corriente Alterna (AC)) y otra debida a las pérdidas de conducción sobre los transistores cuando están conduciendo (pérdidas de Corriente Directa (DC)).

Las pérdidas de DC no son dependientes de la frecuencia de conmutación, solamente dependen del ciclo de trabajo en el cual opera el regulador. Pero las pérdidas de AC están íntimamente relacionadas con la frecuencia, dado que se originan durante la conmutación misma de los transistores. Estas pérdidas se generan en un intervalo de tiempo en el cual el transistor (o transistores en fuentes síncronas, como las que incluye el LTC3633) no está del todo prendido o apagado pero todavía circula corriente por el mismo. Cabe recordar que cuando un transistor no está del todo activado, su resistencia es mayor y por lo tanto, la potencia que disipará será mayor. Cuanto mayor sea la frecuencia de conmutación, mayor cantidad de veces se apagarán y prenderán los transistores del regulador en un mismo intervalo de tiempo, aumentando las pérdidas de AC.

Al utilizar una frecuencia de conmutación alta, la inductancia y capacitancia requerida en el filtro de salida es menor. Por lo tanto, se pueden escoger capacitores e inductores mas pequeños, reduciendo el tamaño final del circuito. También, al utilizar capacitores e inductores mas pequeños, la respuesta del lazo de control es más rápida.

En la nota de aplicación [69] se propone un modelo en el cual el *ripple* de salida de la fuente se debe a la suma lineal del *ripple* debido a la capacitancia de salida y debido a la resistencia de carga. La Ecuación B.1 es la expresión del *ripple* en la tensión de salida derivada de dicho modelo. Este modelo es útil para realizar cálculos aproximados, donde se puede visualizar que a mayor frecuencia, menor *ripple* habrá en la tensión de salida.

$$V_{out,p2p,linear} = \frac{I_{p2p}}{8C_{OUT}f_{SW}} + I_{p2p}R_{OUT} \quad (\text{B.1})$$

Pero, como se menciona en esta otra nota de aplicación [70], este tipo de reguladores introducen diversos tipos de ruidos en el sistema. Estos ruidos se pueden llegar a acoplar al *ripple* en la tensión de salida. El otro fenómeno que puede encontrarse son los transitorios de conmutación, los cuales ocurren cuando los transistores del regulador se prenden y apagan.

Por lo tanto, la elección de la frecuencia de conmutación es una relación de compromiso en la cual el diseñador debe elegir qué característica es más importante para la aplicación en cuestión. En este diseño, se priorizó minimizar el *ripple* en la tensión de salida sobre la eficiencia, por lo que se escogieron frecuencias más bien altas, por encima de 1MHz .

Se decidió dejar configurable la selección de la frecuencia de conmutación y de la red de compensación con resistores de $0\ \Omega$. De esta manera se puede ajustar la red de compensación luego de que la placa sea manufacturada en el caso en el cual la fuente sea inestable. Por lo tanto, con estos resistores de $0\ \Omega$ se pueden escoger dos configuraciones:

- Utilizar la frecuencia de conmutación por defecto del regulador, la cual es 2 MHz , y el lazo de compensación interno del mismo.

- Fijar la frecuencia de conmutación a $1,2\text{ MHz}$ y utilizar una red de compensación externa diseñada particularmente para esta aplicación. Esta opción permite iterar sobre el lazo de compensación y la frecuencia de conmutación en el caso en el cual la fuente sea inestable. Para diseñar la compensación del regulador, el fabricante recomienda estudiar un modelo de primer orden para aumentar el entendimiento del lazo de control y luego obtener los valores de la red de compensación con el simulador LTPowerCAD que ellos mismos proveen. En el anexo C se encuentran notas de estudio sobre el lazo de compensación de esta fuente.

B.1.2. Secuenciado

Este integrado provee señales de *power good* y de habilitación para cada canal. La señal de *power good* se fijará a 0 V si la tensión de salida del canal se encuentra por fuera del $\pm 8\%$ del punto de regulación configurado. Para que dicha señal vuelva a un nivel lógico alto, la tensión de salida deberá moverse hasta un valor dentro del $\pm 5\%$ del punto de regulación.

Cada canal es habilitado cuando la tensión en la señal de habilitación supera $1,22\text{ V}$. Por lo tanto, para cumplir con la secuencia de encendido requerida por Xilinx para los rieles de alimentación se conectó la señal de *power good* de la fuente de 1 V a la señal de habilitación de la fuente de $1,8\text{ V}$. Luego, se conectó la señal de *power good* de la fuente de $1,8\text{ V}$ a las señales de habilitación de las fuentes de $2,5\text{ V}$ y $3,3\text{ V}$. Las salidas de *power good* son *open drain*, así que se colocaron *pull-ups* a la salida de un regulador de tensión de $3,3\text{ V}$ interno que tiene cada chip. Esto permite que la señal de *power good* del regulador de 1 V pueda superar el nivel mínimo de $1,22\text{ V}$ para habilitar al regulador de $1,8\text{ V}$.

En los reguladores de $2,5\text{ V}$ y $3,3\text{ V}$ se colocaron leds para indicar cuando las señales de *power good* se activan para poseer un indicador rápido de que las fuentes encendieron correctamente.

B.1.3. Configuración de punto de regulación

La tensión de salida del regulador se ajusta con un divisor resistivo entre el pin FB y la salida del regulador. Con la Ecuación B.2 se pueden calcular los valores de dichos resistores.

$$V_{OUT} = 0,6(1 + \frac{R_2}{R_1}) \quad (\text{B.2})$$

En la Tabla B.2 se encuentran los valores de resistores R_1 y R_2 escogidos para cada regulador. La designación R_1 y R_2 respeta el orden provisto en el circuito de referencia de la hoja de datos [67].

Tabla B.2

Resultados de los cálculos de los resistores de realimentación de tensión de cada regulador.

Regulador [V]	R_1 [k Ω]	R_2 [k Ω]
1	88.7	59
1.8	10	20
2.5	18.7	59
3.3	4.42	20

B.1.4. Elección del inductor

El valor mínimo de inductancia requerida puede determinarse con la Ecuación B.3 (dicha ecuación fue obtenida de la hoja de datos del chip [67]).

$$L > \frac{V_{OUT}}{f_{SW}\Delta I_{LPP}} \left(1 - \frac{V_{OUT}}{V_{IN(MAX)}}\right) \quad (\text{B.3})$$

Se tomó como punto de partida una corriente de *ripple* pico a pico Δ_{LPP} igual al 40% de la corriente máxima de salida definida por diseño (3 A) y se normalizó a valores de inductores estándar. Cabe destacar que el fabricante recomienda que la corriente de *ripple* sobre el inductor sea al menos el 30% de la corriente máxima de salida, dado que la corriente tiene que ser lo suficientemente grande como para ser medida y utilizada por el lazo de control.

En la Tabla B.3 se encuentran los valores de inductancia escogidos y el *ripple* en la corriente sobre el inductor para cada regulador.

Tabla B.3

Resultados de los cálculos de inductancia y corriente de *ripple* en el inductor para cada regulador.

Regulador [V]	Inductor [μH]	Corriente de <i>ripple</i> (pico a pico) [mA]	Porcentaje de la corriente máxima [%]
1	0.68	1131.22	37.707%
1.8	1.2	1076.92	35.897%
2.5	1.5	1121.79	37.393%
3.3	1.8	1139.96	37.999%

B.1.5. Ciclo de trabajo

Dado que el regulador debe mantener prendido un tiempo mínimo $t_{OFF(MIN)}$ el MOSFET inferior de salida para luego prender el MOSFET superior, existe una limitación en el máximo ciclo de trabajo al cual puede operar la fuente. Dicho ciclo de trabajo máximo se calcula con la Ecuación B.4, donde t_{DEAD} es el tiempo de retraso entre que el regulador deshabilita el MOSFET superior y habilita el MOSFET inferior (y viceversa). Una explicación más detallada del tiempo muerto en reguladores *buck* se puede encontrar en la nota de aplicación [71]. Para este regulador: $t_{OFF(MIN)} = 45 ns$ y $t_{DEAD} = 10 ns$.

$$D_{MAX} = 1 - f_{sw}(t_{OFF(MIN)} + 2t_{DEAD}) \quad (B.4)$$

Análogamente, el regulador debe mantener prendido un tiempo mínimo t_{ONMIN} el MOSFET superior de salida para luego prender el MOSFET inferior. Este tiempo fija el ciclo de trabajo mínimo con el cual puede trabajar el regulador según la Ecuación B.5. Para este regulador: $t_{ON(MIN)} = 10ns$.

$$D_{MIN} = f_{sw}t_{ON(MIN)} \quad (B.5)$$

Para esta aplicación, operando los reguladores a una frecuencia de 1,2 MHz, el ciclo de trabajo al cual pueden operar debe estar dentro del siguiente rango: $1,2\% < D < 92,2\%$.

El ciclo de trabajo al cual operará cada regulador es descrito por la Ecuación B.6. Se puede observar que, para una misma tensión de salida, cuanto menor sea la tensión de entrada, el regulador deberá operar a un ciclo de trabajo mayor. Por lo tanto, el regulador operará al mayor ciclo de trabajo cuando la tensión de entrada tome el valor más pequeño posible (4,5 V) y, análogamente, operará al menor ciclo de trabajo cuando la tensión de entrada tome el valor mas grande posible (13 V).

$$D = \frac{V_{OUT}}{V_{IN}} \quad (B.6)$$

En la Tabla B.4 se encuentran los ciclos de trabajo a los cuales operarán los reguladores. Se puede observar que están dentro del rango recomendado ($1,2\% < D < 92,2\%$).

Tabla B.4

Rango de ciclos de trabajo posibles de cada regulador.

Regulador [V]	$D_{VIN=13 V}$ [%]	$D_{VIN=4,5 V}$ [%]
1	7.69	22.22
1.8	13.85	40
2.5	19.23	55.56
3.3	25.38	73.33

B.1.6. Elección de los capacitores de entrada

Los reguladores Buck generan pulsos de corriente en su entrada, es por ello que se colocan capacitores allí para que puedan suministrar la corriente durante estos eventos. Si estos capacitores no estuvieran allí, la fuente que alimenta al regulador Buck debería suplir dicha corriente. En la nota de aplicación [72] se detalla este fenómeno y se describe la Ecuación B.7 con la cual se puede calcular la capacitancia mínima requerida para que el *ripple* de entrada esté por debajo de cierto valor.

$$C_{IN} \geq \frac{D(1-D)I_{OUT(MAX)}}{\Delta V_{IN} f_{sw}} \quad (B.7)$$

En esta ecuación, D es el ciclo de trabajo y existe un máximo cuando $D = 0,5$. Para evaluar el peor caso, se tomó como ciclo de trabajo 0,5 para los reguladores los cuales el ciclo de trabajo puede llegar a ser mayor a 50 %. Para los reguladores los cuales su ciclo de trabajo máximo no llega a ser 50 %, se tomó como ciclo de trabajo el máximo que pueden alcanzar. Los resultados de estos cálculos se pueden observar en la Tabla B.5.

Tabla B.5

Resultados de los cálculos de capacitancia de entrada mínima de cada regulador.

Regulador [V]	C_{IN} mínimo [μF]
1	8.642
1.8	12
2.5	12.5
3.3	12.5

Los capacitores de entrada deben poder soportar el estrés térmico generado por la corriente de *ripple* circulando en la entrada. En la hoja de datos del regulador se especifica la Ecuación B.8, la cual describe la corriente Media Cuadrática (o *Root Mean Square* en inglés) (RMS) máxima que soportarán los capacitores.

$$I_{CIN(RMS)} = I_{OUT(MAX)} * \frac{\sqrt{(V_{OUT}(V_{IN(MIN)} - V_{OUT}))}}{V_{IN(MIN)}} \quad (B.8)$$

En la Tabla B.6 se encuentran las estimaciones de corrientes RMS que deberían soportar los capacitores.

Tabla B.6

Corriente RMS máxima que deberán soportar los capacitores de entrada.

Regulador [V]	I_{RMX} [mA]
1	799.41
1.8	1036.15
2.5	1182.34
3.3	1305.63

Se escogieron capacitores de Murata, dado que el fabricante provee una gran cantidad de información de cada uno de ellos. En este caso, provee la variación de la capacitancia en función de la tensión DC entre los terminales del capacitor. También, provee el aumento de temperatura del capacitor en función de la corriente RMS circulando en el.

Se colocaron los capacitores GRM32EC81C476KE15L y GRM21BR61E226ME44 en la entrada de cada regulador para proveer la capacitancia mínima necesaria. En las Figuras B.4 y B.5 se puede visualizar la variación de la capacidad de estos capacitores en función de la tensión DC. Se puede ver que cuando los capacitores se exponen a una tensión de 13 V, no bastaría solamente con el capacitor de 47 μF , en este escenario es donde el capacitor de 22 μF ayuda a cumplir con las capacitancias

mínima expuestas en la Tabla B.5. Por otro lado, también se puede observar que los capacitores no elevarían su temperatura de manera apreciable frente a las corrientes estimadas que se encuentran en la Tabla B.6.

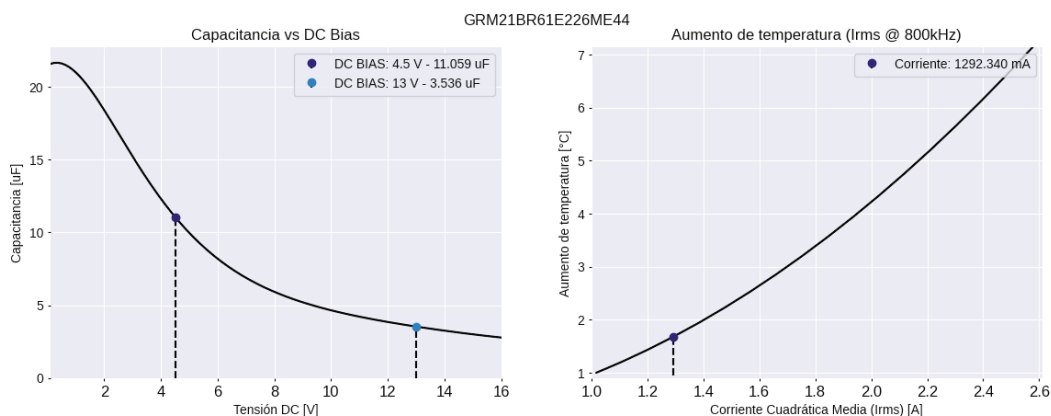


Figura B.4: Izquierda: Capacidad del GRM21BR61E226ME44 en función de la tensión en continua entre sus terminales. Derecha: Aumento de temperatura en dicho capacitor en función de su corriente RMS.

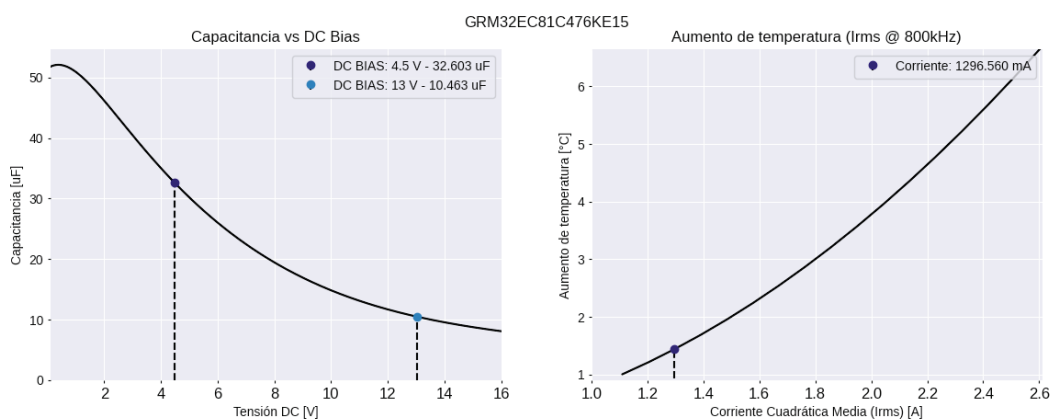


Figura B.5: Izquierda: Capacidad del GRM32EC81C476KE15 en función de la tensión en continua entre sus terminales. Derecha: Aumento de temperatura en dicho capacitor en función de su corriente RMS.

B.1.7. Elección de los capacitores de salida

Cuando el regulador experimenta un aumento en la carga, los capacitores de salida proveen la corriente a la carga hasta que el lazo de control del regulador aumenta lo suficiente la corriente que circula por el transistor de salida. Según la hoja de datos del regulador, necesita entre 3 y 4 ciclos para que el lazo de control pueda responder a este pedido de energía de la carga, pero solo en el primer ciclo la tensión de salida disminuye linealmente.

Por lo tanto, el fabricante recomienda tomar como capacitancia de salida mínima la descrita por la Ecuación B.9, donde ΔI_{OUT} es el incremento de corriente requerido por la carga y V_{DROOP} la caída de tensión máxima permitida tras este incremento de corriente en el primer ciclo del lazo de control. Rápidamente se puede ver que al flexibilizar el requerimiento de V_{DROOP} disminuye la capacitancia mínima requerida.

$$\Delta C_{OUT} > \frac{3\Delta I_{OUT}}{f_{sw}V_{DROOP}} \quad (\text{B.9})$$

Para cada regulador, se tomó ΔI_{OUT} como la corriente máxima ($I_{OUT(MAX)} = \Delta I_{OUT}$) y V_{DROOP} como el 80% de la tensión de *ripple* máxima ($V_{DROOP} = 0,8\Delta V_{OUT}$).

Tabla B.7

Resultados de los cálculos de capacitancia de salida mínima de cada regulador.

Regulador [V]	V_{DROOP} [mV]	C_{OUT} [μF]
1	40	187.5
1.8	72	104.2
2.5	100	75
3.3	132	56.8

Se escogieron dos capacitores de Murata: el GRM21BR60J107ME15L, de 100 μF , y el GRM21BR60J226ME39K, de 22 μF . Se calcularon las capacitancias para las distintas tensiones DC, como se puede visualizar en la Figura B.6, para determinar la cantidad de capacitores necesarios.

Los capacitores colocados y la capacitancia final obtenida se encuentran en la Tabla B.8. Se puede observar que los valores de capacitancia total satisfacen los requeridos en la Tabla B.7.

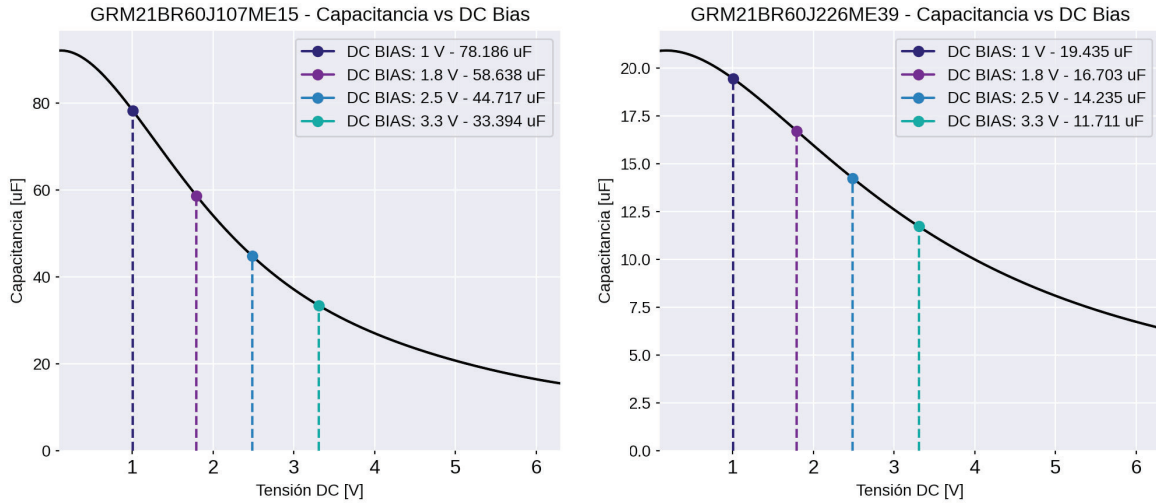


Figura B.6: Capacidad de los capacitores de salida en función de la tensión en continua. Números de parte de los capacitores: GRM21BR60J107ME15L y GRM21BR60J226ME39K.

Tabla B.8

Capacitores de salida utilizados para los reguladores y la capacitancia total.

Regulador [V]	Capacitores de 100 μF	Capacitores de 22 μF	Capacitancia total [μF]
1	3	2	273.43
1.8	2	2	150.7
2.5	2	2	117.96
3.3	2	2	90.21

B.1.8. Estimación del *ripple* en la tensión de salida

El *ripple* en la tensión de salida puede calcularse con la Ecuación B.10. Puede visualizarse que a mayor valor de capacitancia de salida y menor *Equivalent Series Resistance* (ESR) en estos capacitores, menor será el *ripple* en la tensión de salida del regulador.

$$\Delta V_{OUT} = \Delta I_{LPP} \left(ESR + \frac{1}{8f_{sw} C_{OUT}} \right) \tag{B.10}$$

Para calcular este valor, hay que obtener la ESR total que es presentada al regulador. Dado que se están utilizando dos capacitores distintos, la obtención de esta ESR total no es directa. Para esto, hay que obtener el modelo paralelo de los capacitores utilizados y luego calcular el paralelo de las resistencias del modelo paralelo. En el artículo [73] se describe un método para calcular el modelo paralelo de un capacitor. De este método resulta la Ecuación B.11, con la cual se puede calcular la resistencia paralelo del capacitor a partir de su ESR y su capacitancia.

$$R_{mp} = \frac{ESR^2 + (\frac{1}{2\pi Cf})^2}{ESR} \tag{B.11}$$

Como se ve en la Ecuación B.11, la resistencia del modelo paralelo depende de la frecuencia, por lo tanto, en esta estimación se utilizará la frecuencia de conmutación: 1,2 MHz.

La ESR de los capacitores depende de la frecuencia y de la tensión DC en la cual está operando. También, la capacidad depende de la tensión DC a la cual sea expuesto el capacitor. El fabricante de los capacitores escogidos provee curvas con estos valores (dependencia de la ESR en función de la frecuencia y dependencia de la capacidad en función de la tensión DC). Dichos valores se pueden visualizar en la Figuras B.6 y B.7.

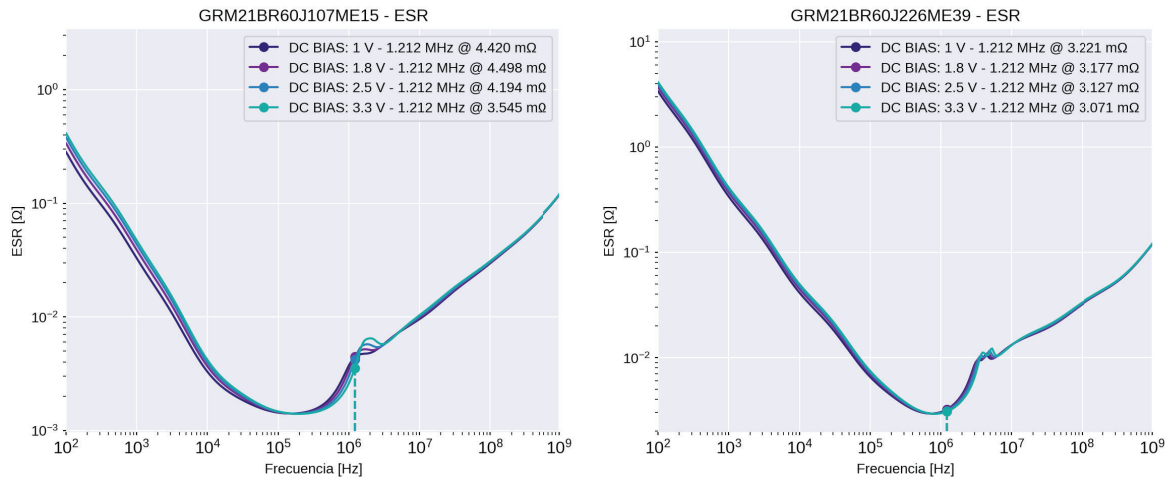


Figura B.7: ESR de los capacitores utilizados en la salida de los reguladores en función de la frecuencia. Números de parte de los capacitores: GRM21BR60J107ME15L y GRM21BR60J226ME39K.

Con el valor de ESR y capacitancia de cada capacitor se calculó la resistencia del modelo paralelo R_{mp} para cada valor de bias DC. Dichos resultados se encuentran en la Tabla B.9.

Tabla B.9

Resistencia del modelo paralelo para los capacitores de salida de los reguladores de tensión. Cálculos realizados para una frecuencia de 1,2 MHz.

DC-BIAS [V]	GRM21BR60J107ME15L (100 μF) - R_{mp} [mΩ]	GRM21BR60J226ME39K (22 μF) - R_{mp} [mΩ]
1	5.071	17.678
1.8	5.635	23.024
2.5	6.291	30.887
3.3	7.995	44.832

En función de la cantidad de capacitores a la salida de cada regulador, la resistencia del modelo paralelo y la capacidad a dicha tensión DC de cada uno de ellos, se calculó el ripple estimado en la tensión de salida con la Ecuación B.10. Los resultados se encuentran en la Tabla B.10. Cabe destacar

que esta es una estimación de primer orden. Por ejemplo, al calcular el *ripple* con la Ecuación B.10, se asumió que los capacitores no varían su capacitancia con la frecuencia, lo cual no es cierto.

Tabla B.10

Ripple estimado en la tensión de salida de cada regulador. Cálculos realizados para una frecuencia de 1,2 MHz.

Regulador [V]	ESR Equivalente [$m\Omega$]	ΔV_{OUT} [mV]
1	1.419	2.036
1.8	2.264	3.182
2.5	2.613	3.923
3.3	3.392	5.184

B.1.9. Encendido suave

Este regulador implementa un encendido suave al cual se le puede configurar el tiempo de subida colocando un capacitor en el pin *TRACKSS*. La relación entre el capacitor colocado en *TRACKSS* y el tiempo de subida se visualiza en la Ecuación B.12, donde C_{SS} es el capacitor en cuestión.

$$t_{ss} = 430000\Omega C_{SS} \quad (\text{B.12})$$

Más allá de que el regulador de 1,8 V se habilite con la señal de *power good* del regulador de 1 V, se escogieron capacitores tales que el tiempo de establecimiento de este último sea menor al del regulador de 1,8 V. Dado que Xilinx recomienda un tiempo máximo en el cual $V_{CCAUx} - V_{CCO} > 2,625$ V, el tiempo de establecimiento de las fuentes de 2,5 V y 3,3 V se escogió de manera que sea menor que el de la fuente de 1,8 V. Los valores de *soft start* de cada regulador se pueden encontrar en la Tabla B.11.

Tabla B.11

Resultados de los cálculos de encendido suave para cada regulador.

Regulador [V]	Capacitor C_{SS} [nF]	Tiempo de establecimiento [ms]
1	10	4.3
1.8	22	9.46
2.5	10	4.3
3.3	10	4.3

B.1.10. Modo de operación

El regulador puede operar en dos modos:

- *Burst Mode*: provee una mayor eficiencia a corrientes de salida pequeñas a expensas de un mayor *ripple* en la tensión de salida.
- *Forced Continuous Mode*: provee un menor *ripple* en la tensión de salida para pequeñas corrientes a expensas de eficiencia.

Se optó por utilizar el modo *Forced Continuous* para asegurar un *ripple* de salida bajo. Pero, se colocó un resistor de 0Ω para elegir el modo *Burst* si se lo desea.

B.1.11. Conmutación en fase y desfase

Cuando los MOSFETs superiores de ambos canales son conmutados al mismo tiempo para permitir el paso de corriente hacia la salida, los capacitores y la fuente de entrada tienen que suplir un pulso de corriente de gran magnitud. Una manera de mitigar este efecto es que los MOSFETs conmuten con un desfase de 180° , de esta manera no van a drenar corriente de la entrada al mismo

tiempo, relajando el requerimiento de capacidad de entrada. Pero, una desventaja de realizar esto es que si alguno de los canales está operando al 50% se puede acoplar ruido generado por la conmutación misma de un canal a otro, generando *jitter* en la frecuencia del canal afectado. Aunque este efecto puede ser mitigado diseñando con cuidado el *layout* de la placa.

Este integrado permite elegir si se quiere conmutar ambos canales en fase o con un desfase de 180°. Por defecto, se deja configurado para que conmuten en desfase, pero se colocaron resistores de 0 Ω para utilizar la otra configuración en caso de que se quiera lo contrario.

B.1.12. Simulación en LTPowerCAD

Los valores de los componentes externos de los reguladores fueron ajustados con el *software* del fabricante, LTPowerCAD. En las figuras B.8 y B.9 se pueden observar las fuentes simuladas en LTPowerCAD.

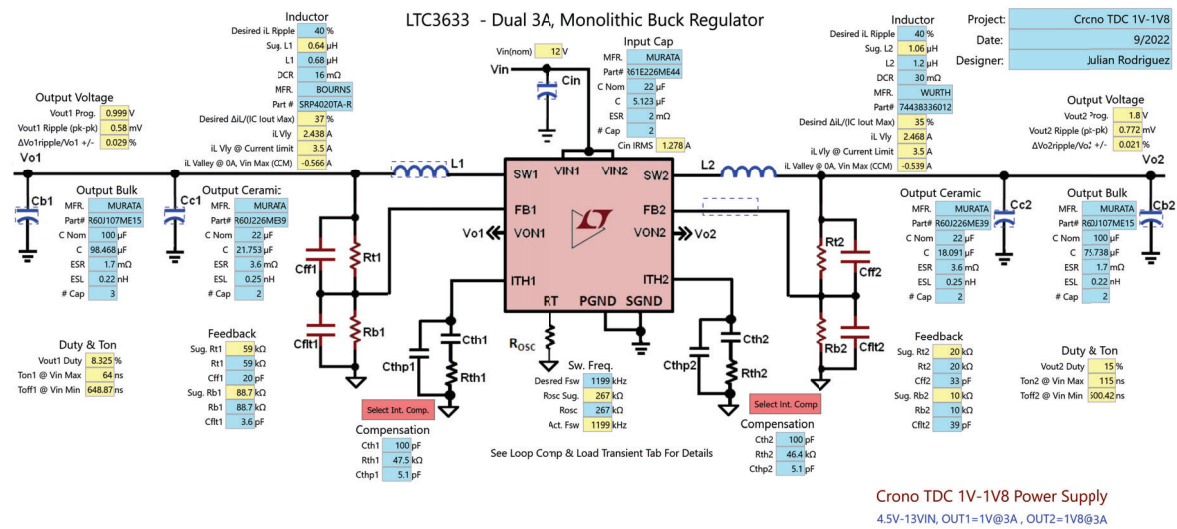


Figura B.8: Fuente de 1 V y 1,8 V simulada en LTPowerCAD. En amarillo se pueden observar los valores sugeridos por el programa y en azul los escogidos finalmente en el diseño.

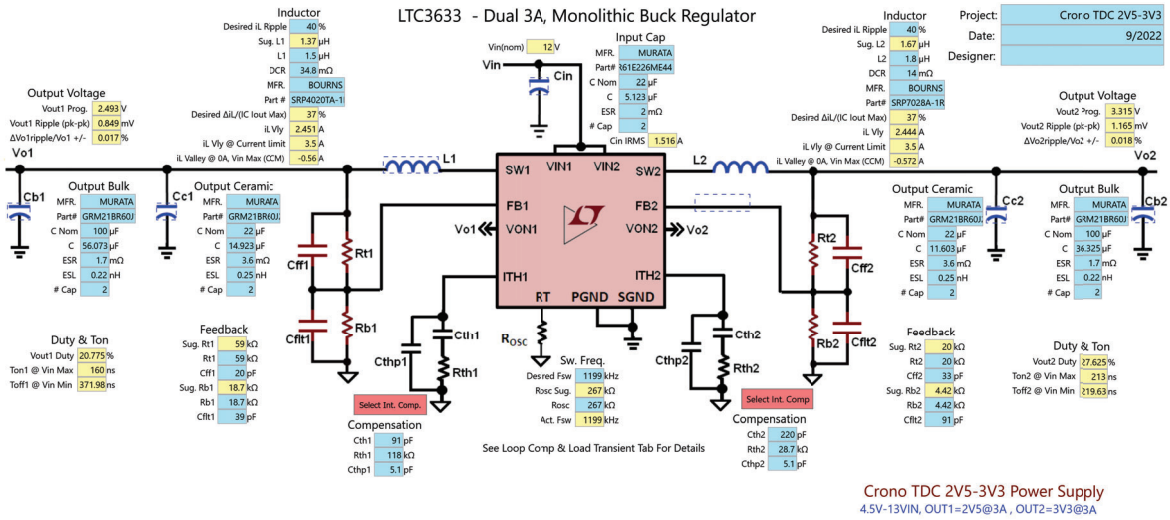
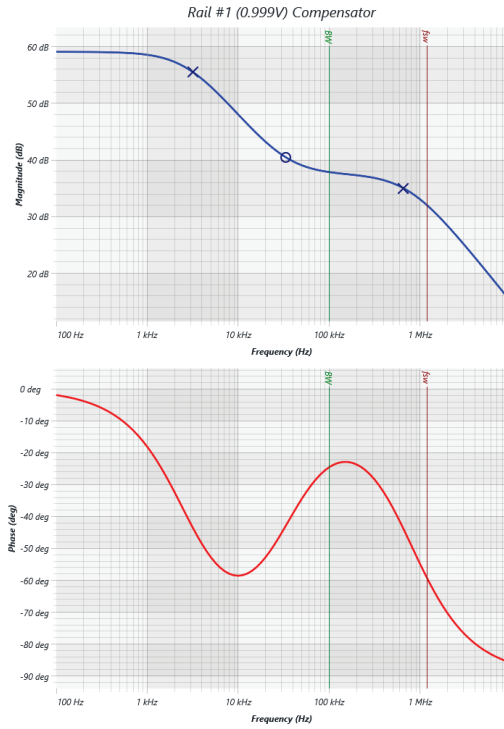
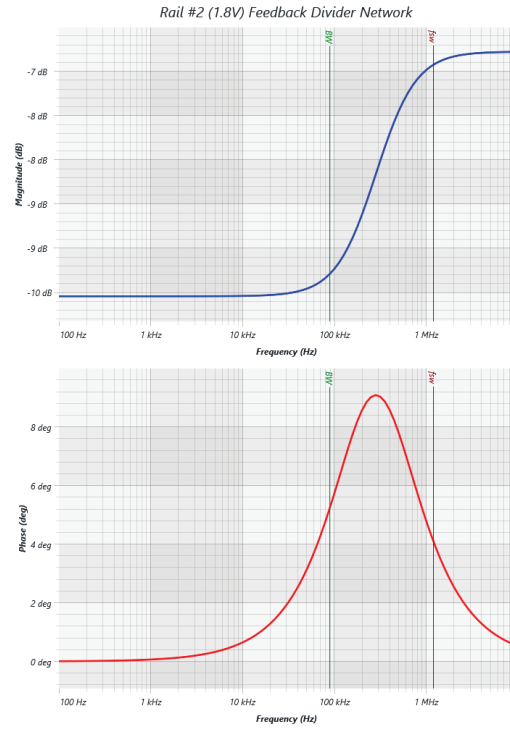


Figura B.9: Fuente de 2,5 V y 3,3 V simulada en LTPowerCAD. En amarillo se pueden observar los valores sugeridos por el programa y en azul los escogidos finalmente en el diseño.

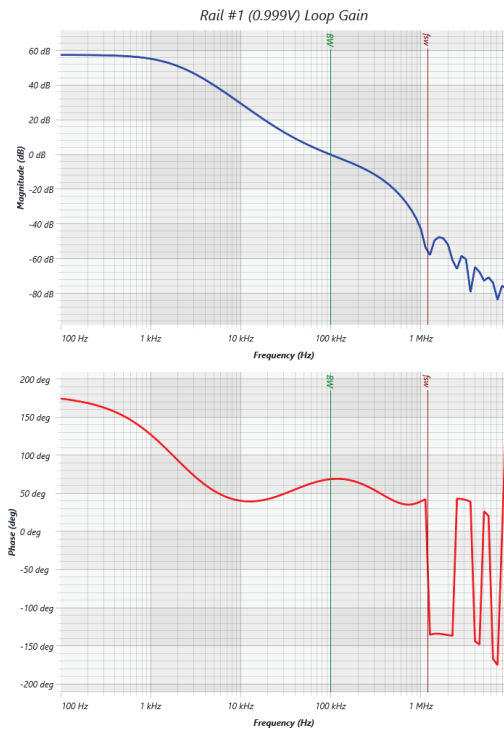
Este *software* permite simular el diagrama de Bode de los compensadores, de las redes de realimentación de tensión y de la ganancia de lazo. También, permite simular la respuesta transitoria de las fuentes frente a un escalón de cierta corriente, en este caso, se simuló para un escalón de 3 A. En las Figuras B.10, B.11, B.12 y B.13 se encuentran las simulaciones de las fuentes.



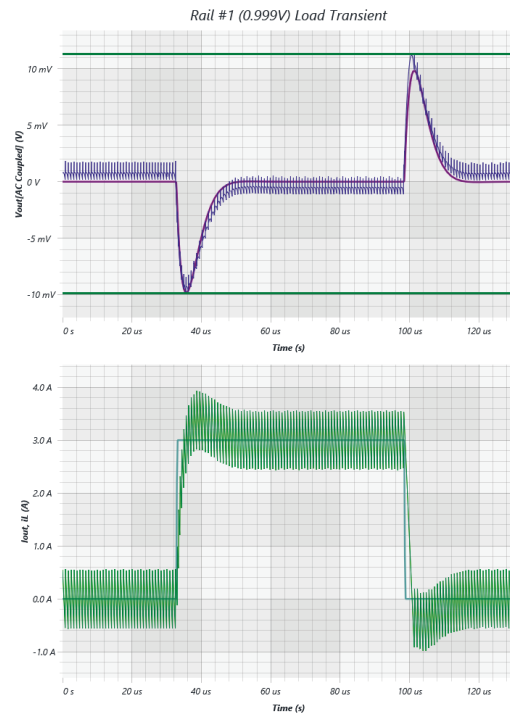
(a) Diagrama de Bode del compensador de la fuente de 1 V.



(b) Diagrama de Bode de la red de realimentación de voltaje de la fuente de 1 V.

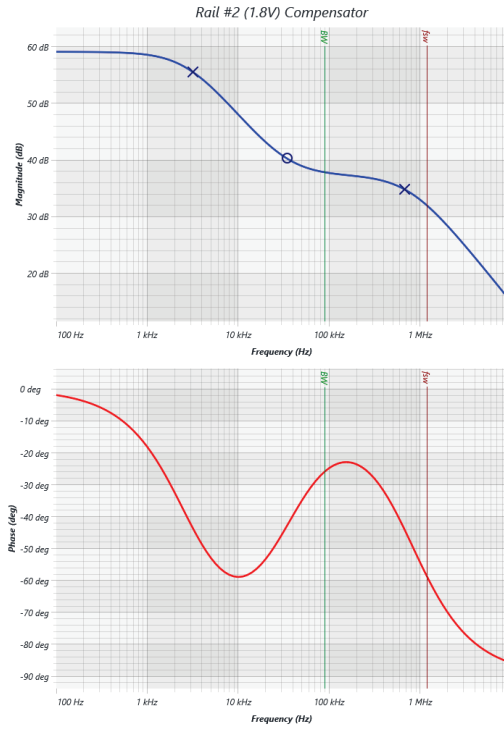


(c) Diagrama de Bode de la ganancia de lazo de la fuente de 1 V.

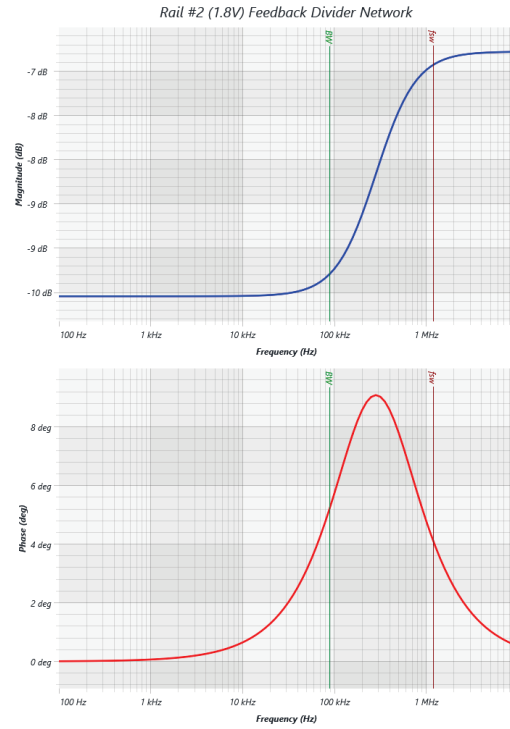


(d) Respuesta transitoria de la fuente de 1 V a un escalón de 3 A.

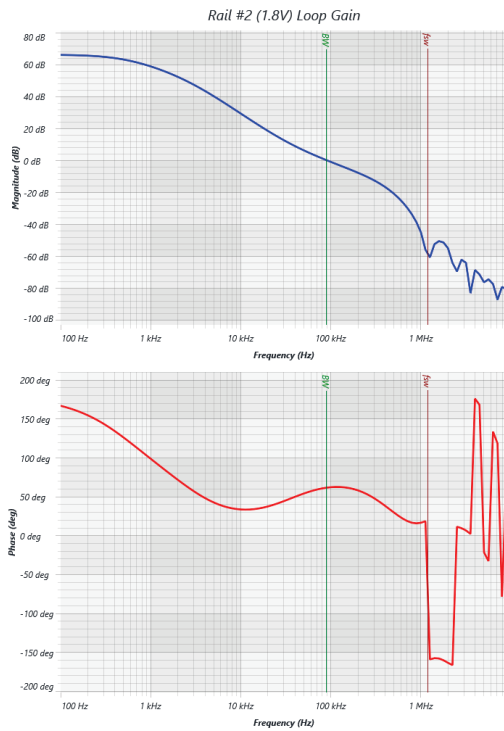
Figura B.10: Simulación de la fuente de 1 V en LTPowerCAD.



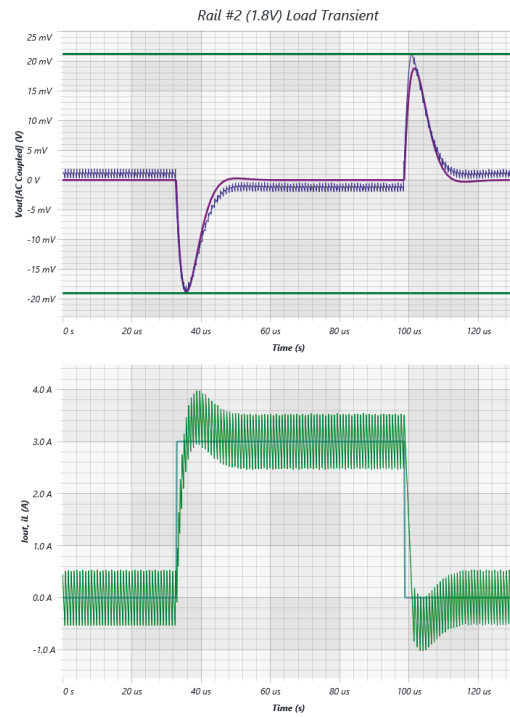
(a) Diagrama de Bode del compensador de la fuente de 1,8 V.



(b) Diagrama de Bode de la red de realimentación de voltaje de la fuente de 1,8 V.

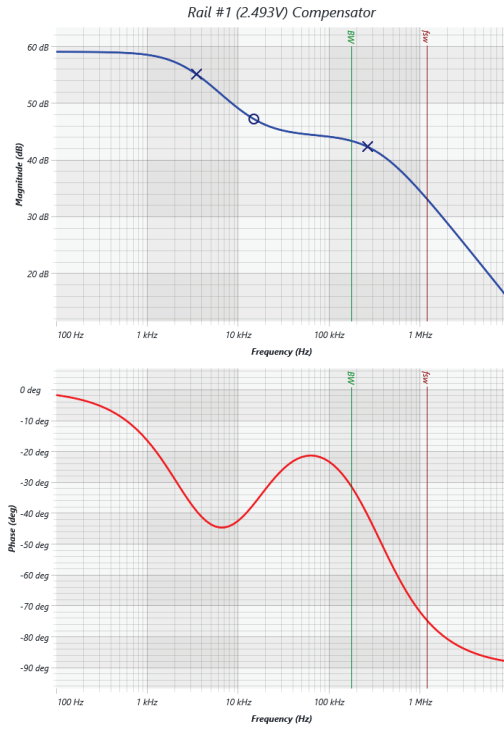


(c) Diagrama de Bode de la ganancia de lazo de la fuente de 1,8 V.

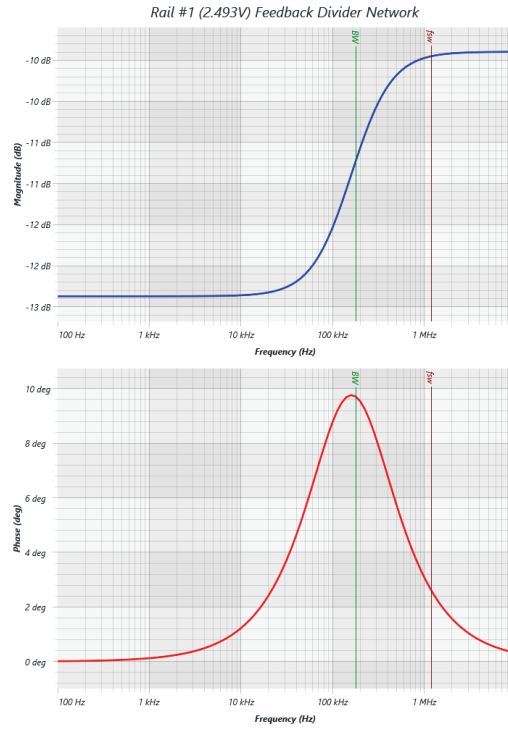


(d) Respuesta transitoria de la fuente de 1,8 V a un escalón de 3 A.

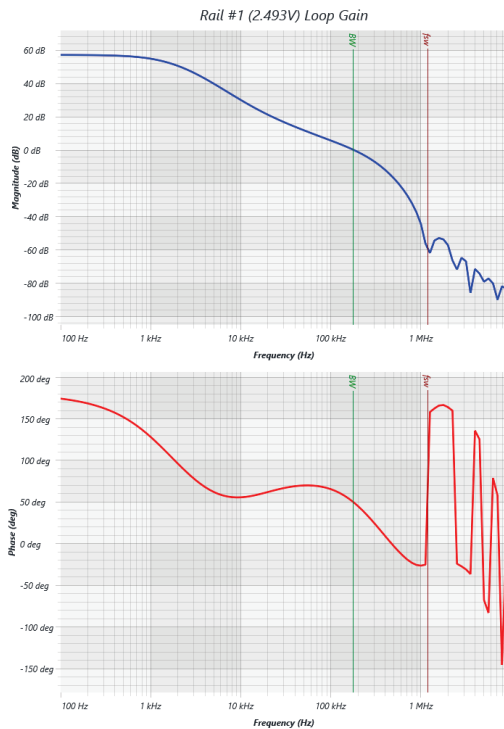
Figura B.11: Simulación de la fuente de 1,8 V en LTPowerCAD.



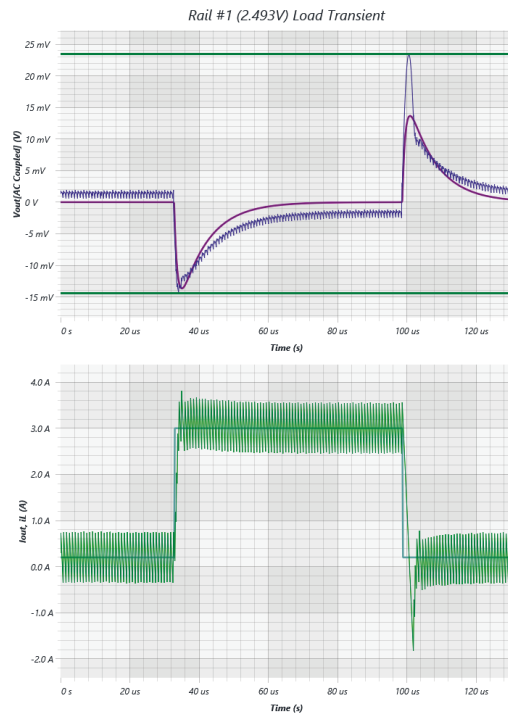
(a) Diagrama de Bode del compensador de la fuente de 2,5 V.



(b) Diagrama de Bode de la red de realimentación de voltaje de la fuente de 2,5 V.

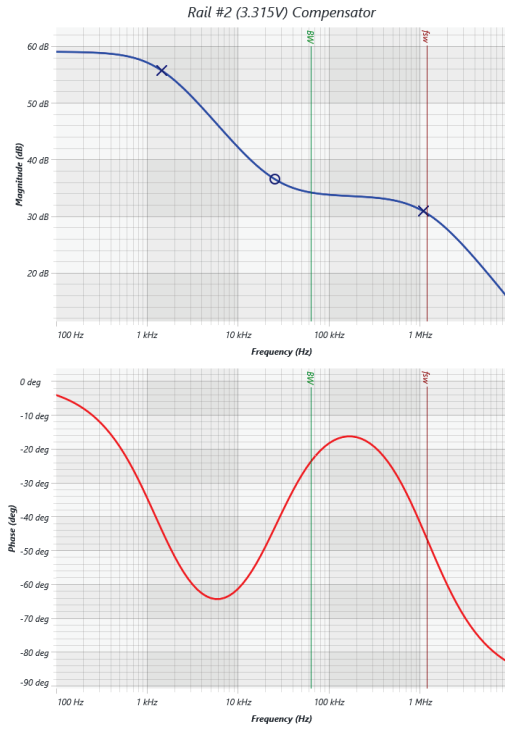


(c) Diagrama de Bode de la ganancia de lazo de la fuente de 2,5 V.

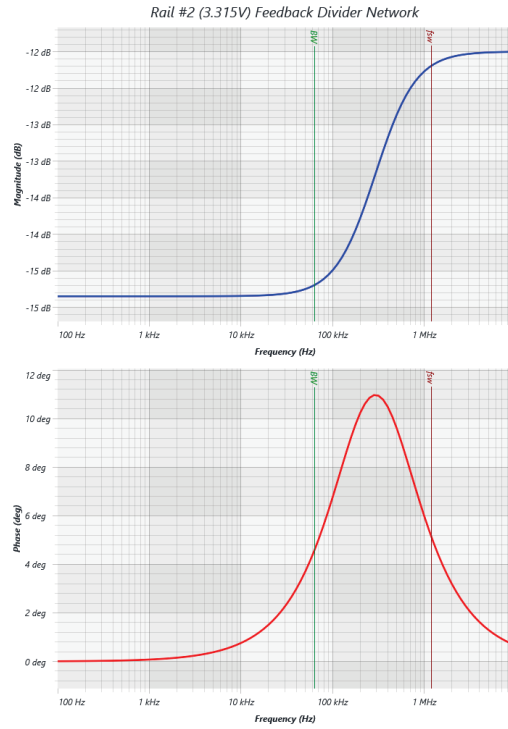


(d) Respuesta transitoria de la fuente de 2,5 V a un escalón de 3 A.

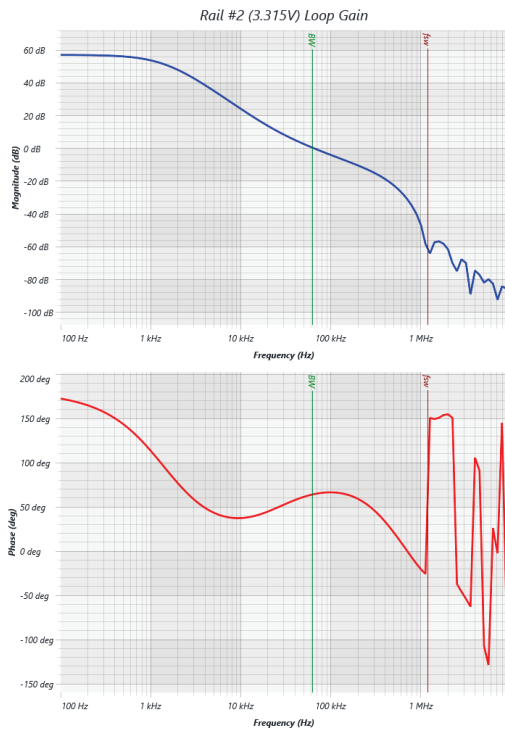
Figura B.12: Simulación de la fuente de 2,5 V en LTPowerCAD.



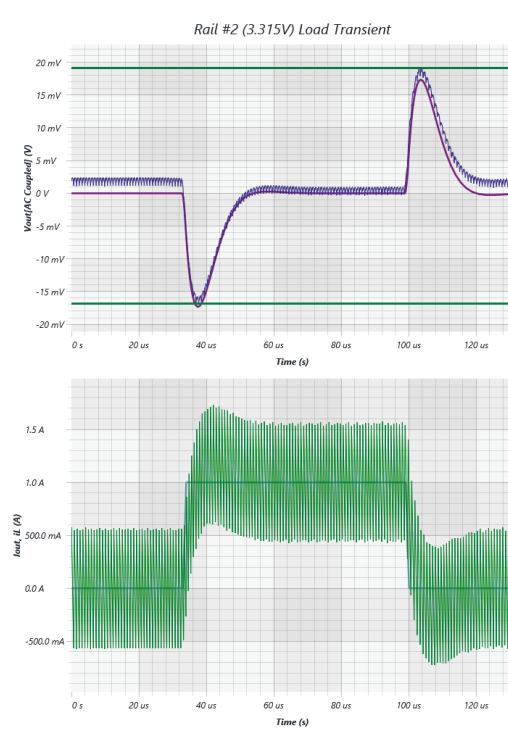
(a) Diagrama de Bode del compensador de la fuente de 3,3 V.



(b) Diagrama de Bode de la red de realimentación de voltaje de la fuente de 3,3 V.



(c) Diagrama de Bode de la ganancia de lazo de la fuente de 3,3 V.



(d) Respuesta transitoria de la fuente de 3,3 V a un escalón de 3 A.

Figura B.13: Simulación de la fuente de 3,3 V en LTPowerCAD.

B.2. FPGA Power Distribution System

Hay varios componentes o interconexiones que están en el camino de la corriente entre el regulador y la FPGA. Nombrando algunos, se encuentra la salida misma del regulador, los capacitores de desacople, las vias, las pistas, los capacitores adicionales agregados a la placa y los pines de la FPGA. Estas interconexiones y componentes componen el PDS, el cual posee una impedancia finita. Este concepto se puede visualizar en la Figura B.14.

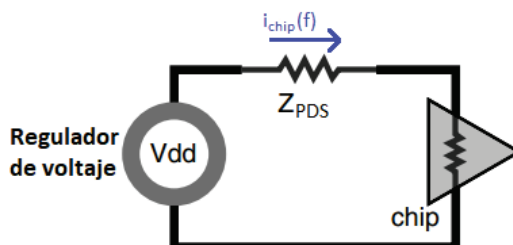


Figura B.14: Corriente consumida por el chip fluyendo desde el regulador de tensión a través del PDS. Adaptada de [39].

La corriente consumida por el chip no es constante y puede variar con determinada frecuencia. Esta corriente, al circular por la impedancia del PDS genera una caída de tensión que fluctúa con el tiempo y a la cual se la denomina comúnmente *ripple*. El objetivo del diseño de un PDS es mantener su impedancia por debajo de un valor máximo dentro de un rango de frecuencias para asegurar que el *ripple* se mantenga por debajo de cierto valor.

Como por lo general no se conoce la información precisa del espectro de consumo de la FPGA, se tomó un acercamiento conservador en el problema, donde se fija como objetivo minimizar la impedancia del PDS desde bajas frecuencias (DC) hasta la frecuencia máxima a la cual se utilizaría la FPGA, lo cual es, 400 MHz en este caso.

Se puede estimar la impedancia máxima del PDS con la corriente transitoria máxima que se espera para cada riel de alimentación con la Ecuación B.13.

$$Z_{max_{PDS}} = \frac{V_{max_{ripple}}}{I_{transitorio}} \quad (B.13)$$

En la Tabla 4.2 se encuentran las especificaciones para cada riel de tensión y en la Figura B.1 se encuentran los consumos dinámicos estimados por Vivado para el diseño que se implementará en la FPGA. El máximo *ripple* utilizado en este cálculo es la mitad del desvío máximo respecto de la tensión nominal especificada en la Tabla 4.2. Por ejemplo, como el riel de 1 V permite un corrimiento máximo de 50 mV se fijó como *ripple* máximo 25 mV (notar que no se están utilizando valores pico a pico). Cabe destacar que se consideró un piso de consumo de 50 mA para los valores de corriente reportados por Vivado. Con esta información se calculó la impedancia máxima de cada PDS, como se puede ver en la Tabla B.12.

Tabla B.12

Impedancia máxima de cada PDS de los rieles de alimentación de la FPGA.

Riel de tensión	Máximo <i>ripple</i> [mV]	Corriente transitoria [mA]	Impedancia máxima [Ω]
1	25	425	0.059
1.8	45	146	0.308
2.5	62.5	50	1.25
3.3	82.5	50	1.65

B.2.1. Composición del PDS

El fabricante de la FPGA recomienda colocar capacitores en paralelo en las entradas de alimentación para minimizar la impedancia del PDS en un cierto rango de frecuencias. El diseño del PDS abarca la selección de estos capacitores de desacople, su ubicación respecto a la FPGA y el ruteo de los mismos.

Merece mención el comportamiento de estos capacitores en el tiempo. Por lo general, los reguladores de tensión pueden mantener la salida de tensión frente a eventos de aumento de carga que llegan hasta unos cientos de kHz. Para eventos mas rápidos, el regulador responde con cierta lentitud y la tensión de salida se ve disminuida por un cierto intervalo de tiempo. Se puede visualizar esto en las Figuras B.10d, B.11d, B.12d y B.13d, en las cuales se simularon respuestas transitorias para cada salida del regulador utilizado. Al presentarse un escalón de corriente abrupto, la tensión del regulador cae y luego se estabiliza.

En estos eventos es donde los capacitores de desacople actúan como una fuente de energía local para la FPGA, manteniendo la tensión estable hasta que el regulador puede proveerla por si mismo. Es por esto que los capacitores de desacople ayudan a mantener el riel de alimentación estable frente a eventos cuya frecuencia se encuentra entre los cientos de kHz y cientos de MHz.

Al elegir estos capacitores y su ubicación respecto a la FPGA es necesario considerar la inductancia parásita de los mismos (la cual se denomina *Equivalent Series Inductance* (ESL)) y de los caminos de corriente en el PCB. Esta inductancia parásita retrasa los cambios en la corriente y son la razón por la cual los capacitores de desacople no pueden responder instantáneamente a transitorios de corriente que ocurren a una frecuencia mayor que su frecuencia efectiva. Por lo tanto, esta inductancia debe ser disminuida escogiendo capacitores con un pequeño valor de ESL y realizando cuidadosamente el ruteo de los mismos. El objetivo del PDS es responder a los transitorios de corriente lo mas rápido posible para mantener el voltaje en un cierto rango.

Un capacitor ideal posee una impedancia que decae a medida que aumenta la frecuencia ($Z = \frac{1}{2\pi fC}$), pero desafortunadamente, los capacitores ideales no existen. Un capacitor *Multi-layer Ceramic Capacitor* (MLCC) real debe ser tratado como un circuito RLC serie. La impedancia de un capacitor real decae hasta una determinada frecuencia llamada **frecuencia de auto resonancia** (f_{res}) y luego aumenta a medida que aumenta la frecuencia. La impedancia a baja frecuencia está dada por la capacitancia y a alta frecuencia, por la ESL. La menor impedancia lograda por el capacitor está limitada por la ESR. Se puede observar la respuesta en frecuencia de un capacitor real en la Figura B.15.

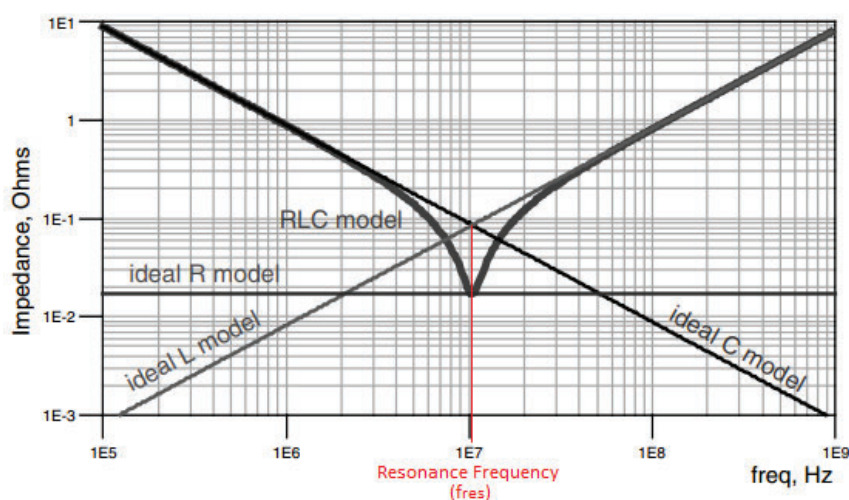


Figura B.15: Respuesta en frecuencia de un capacitor. Adaptada de [39].

Si bien los fabricantes brindan un valor de ESL de sus capacitores, la misma depende mayormente de como es montado el mismo en el PCB respecto a la FPGA. Esto se debe a que la ESL

reportada por el fabricante por lo general está en el orden de 1 nH o menos y la ESL que puede acumular un capacitor debido a su montaje respecto a la FPGA puede ser mucho mayor.

En la guía de usuario [74] se detalla el PDS recomendado para la FPGA utilizada. En la Tabla B.13 se detallan las características de los capacitores recomendados para el PDS. Los capacitores de desacople de mayor valor ($47\ \mu\text{F}$ y $100\ \mu\text{F}$) se utilizan para filtrar ruido de baja frecuencia y los capacitores de menor valor ($470\ \text{nF}$ y $4,7\ \mu\text{F}$) se utilizan para filtrar ruidos en un rango intermedio de frecuencias.

Tabla B.13

Características de los capacitores de desacople recomendadas por Xilinx. Extraído de [74].

Capacitancia [μF]	Tecnología	Empaquetado	ESL máxima [nH]	ESR [$m\Omega$]
100	X5R o X7R	1210	1	< 40
47	X5R o X7R	1210	1	< 40
4.7	X5R o X7R	0805	0.5	< 20
0.47	X5R o X7R	0603	0.5	< 20

B.2.2. Simulación en LTSpice

Se escogieron los capacitores de desacople para el PDS de cada riel de tensión y se simuló en LTSpice para obtener la respuesta en frecuencia de los mismos. Dado que algunas propiedades del capacitor (como la capacitancia) varían con la tensión DC de bias, el fabricante provee modelos SPICE para una variedad de tensiones de bias. En la Figura B.16 se pueden encontrar los circuitos simulados en LTSpice y en la Figura B.17 se encuentran las respuestas en frecuencia.

Se remarcó con líneas horizontales las impedancias máximas que puede poseer cada PDS. También, con una línea vertical se marcaron los $400\ \text{MHz}$, la cual fue la frecuencia de corte definida para cada PDS. Se puede visualizar que la impedancia de cada PDS se coloca por debajo de su impedancia objetivo hasta los $400\ \text{MHz}$ en todos los rieles.

Se agregaron capacitores más allá de los recomendados por Xilinx en la guía de usuario [74] para lograr la impedancia objetivo. Esta práctica también puede visualizarse en la placa de desarrollo Basys 3 [46].

En la hoja *fpga_power.SchDoc* (ver Anexo A.1) se pueden visualizar las alimentaciones de la FPGA y los capacitores que componen el PDS.

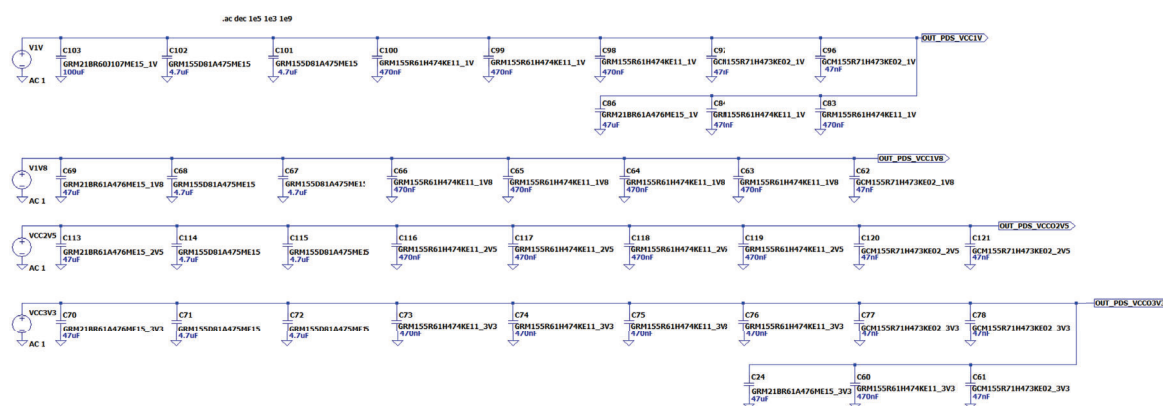


Figura B.16: PDS de cada riel de tensión simulado en LTSpice.

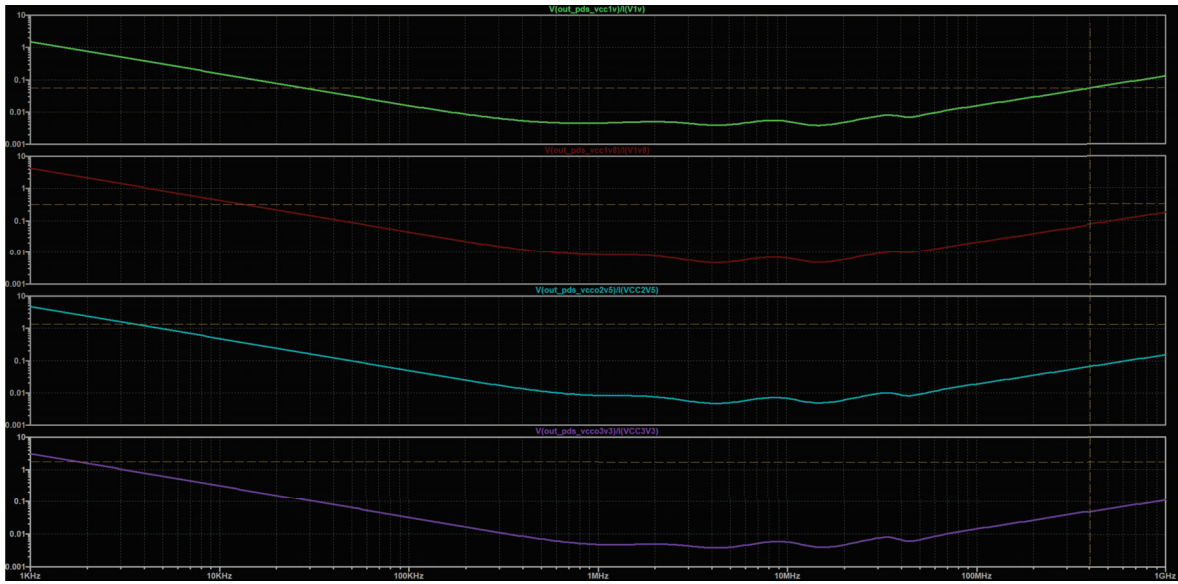


Figura B.17: Respuesta en frecuencia de cada PDS simulado en LTSpice.

C. Notas sobre el lazo de compensación de la fuente *buck*

En este anexo se encuentran notas sobre el lazo de compensación de la fuente *buck* utilizada. Esta investigación se realizó para entender el funcionamiento de la misma y para poder, llegado el caso, ajustar los valores de la red de compensación si esta misma fuera inestable.

El regulador escogido utiliza un control por corriente (denominado *current-mode control*). En este tipo de control, el ciclo de trabajo del regulador es proporcional a una tensión de referencia y a una tensión derivada de la corriente que circula por el inductor o por uno de los elementos conmutables (transistores).

En la Figura C.1 se puede visualizar el principio de funcionamiento de este tipo de reguladores. El amplificador de error genera una tensión de salida que depende de la diferencia entre V_{REF} y la tensión realimentada por el divisor de tensión colocado a la salida (V_{FB}). Un oscilador genera una señal de reloj con una frecuencia f_{osc} y al comienzo de cada ciclo fuerza un estado lógico alto en el *flip-flop* con su señal *set*. Por lo tanto, al inicio de cada ciclo, la salida del *flip flop* se encuentra en un estado lógico alto. Mientras la salida del *flip flop* esté activa, la corriente fluye a través del transistor de salida y la resistencia de sensado R_S , generando una tensión $V_S = R_S I_S$. A partir de este momento, la corriente y la tensión de salida del convertidor empiezan a aumentar hasta que la tensión de sensado V_S es igual a la salida del amplificador de error V_F , cuando esto sucede, la salida del comparador *Pulse Width Modulation* (PWM) se pone en un estado lógico alto, reiniciando el valor de salida del *flip flop* a cero hasta el próximo ciclo. Con este mecanismo el convertidor regula cuando conduce cada uno de los elementos conmutables.

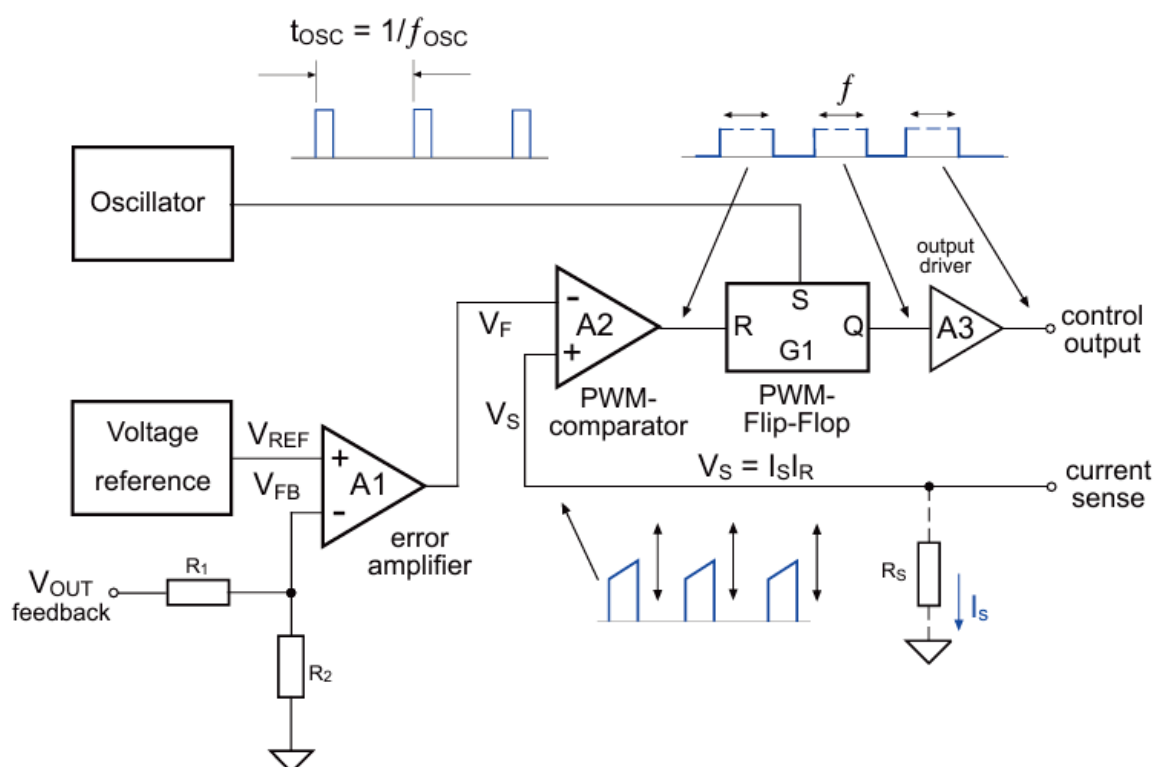


Figura C.1: Concepto de un controlador DC-DC por control de corriente. Donde A_1 es el amplificador de error, A_2 es el comparador PWM y G_1 es un *flip-flop*. La tensión V_{REF} actúa de tensión de referencia y se encuentra compensada contra la temperatura. Extraído de [75].

Como se puede observar en la Figura C.1, estos reguladores poseen dos lazos de realimentación: en uno se realimenta la tensión de salida y en el otro la corriente de salida. Esta separación de lazos permite que se pueda compensar cualquier cambio producido en la corriente y en la tensión de salida rápidamente, lo cual mejora la respuesta transitoria del regulador. En contraste, los reguladores por control de tensión solo pueden responder a los cambios de tensión producidos en la salida, es por ello que para compensar cambios en la corriente deben esperar varios ciclos hasta que se vea una perturbación en la tensión.

En [76] se detalla un modelo de pequeña señal para convertidores DC-DC con control de corriente. Este modelo es preciso pero complejo, por lo cual, termina siendo utilizado solo por simuladores. Por ejemplo, el simulador LTPowerCAD de *Linear Technologies* implementa dicho modelo.

En la Figura C.2 se detalla un modelo de pequeña señal extraído de [77]. Se debe tener en cuenta que, por lo general, lo único que puede personalizar el diseñador es la red de resistores de realimentación, la red de compensación y los componentes pasivos de salida (capacitores e inductor). El resto ya está resuelto dentro del integrado de control del regulador.

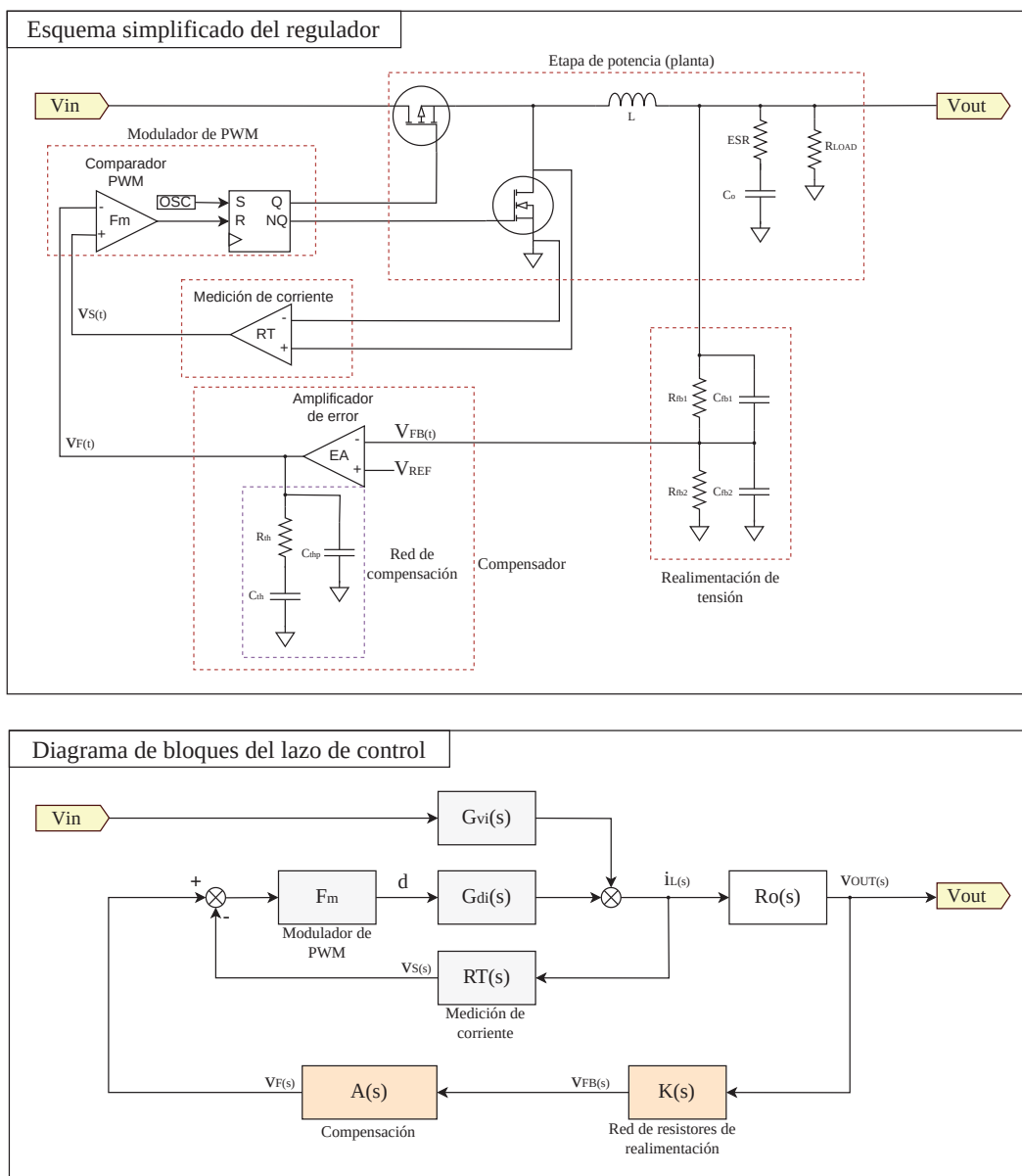


Figura C.2: Esquema simplificado de un regulador *switching* por control de corriente. Basado en [77].

En la nota de aplicación [78] se modela un regulador *step down* y se detallan los pasos para diseñar la red de compensación. Pero, se aclara que el modelo propuesto no es preciso a altas frecuencias y que cualquier diseño debe ser revisado con un simulador que utilice un modelo de mayor complejidad. En la Figura C.3 se puede observar este modelo, donde se encapsula la etapa de potencia y la realimentación de corriente y solo se hace énfasis en la realimentación de tensión.

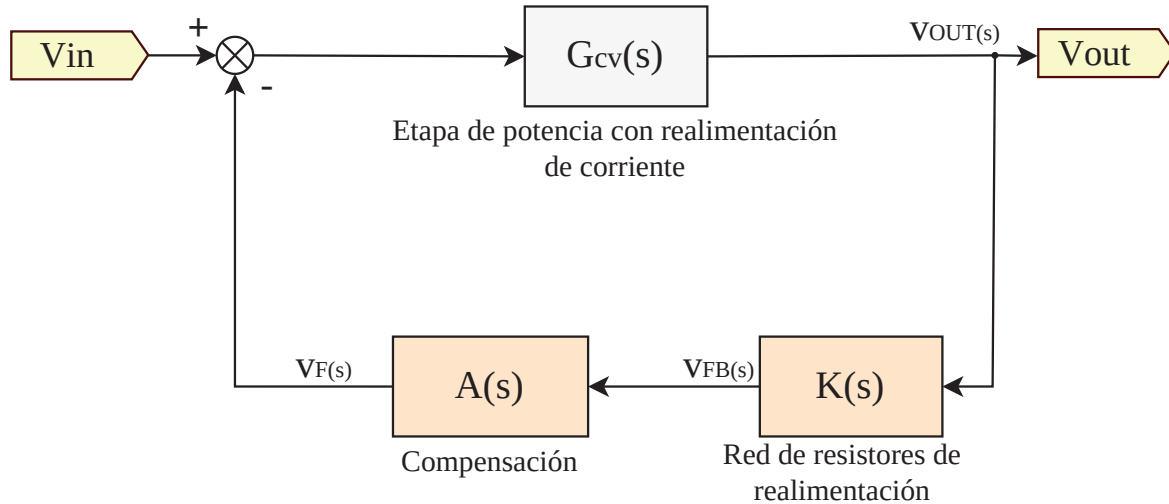


Figura C.3: Esquema simplificado del lazo de control de un regulador Buck por control de corriente, haciendo énfasis en la realimentación de tensión. Basado en [78].

Existen dos variables a monitorear cuando se diseña un lazo de compensación para este tipo de aplicaciones, las cuales se pueden visualizar en la Figura C.4. El **margen de ganancia** es la ganancia cuando la fase cruza por cero, por lo general es un valor negativo. Por otro lado, el **margen de fase** es el valor de fase cuando la ganancia cruza por cero, por lo general es un valor positivo.

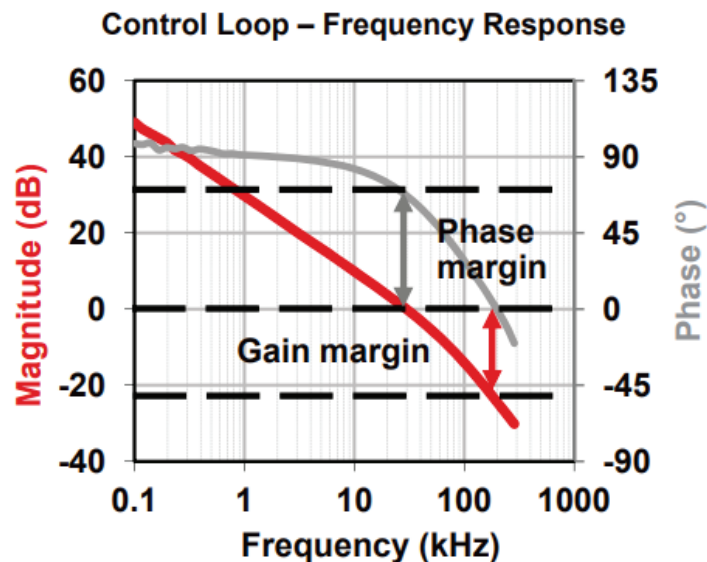


Figura C.4: Margen de fase y margen de ganancia. Extraído de [79].

Un lazo de control cerrado es inestable si la magnitud de la ganancia de lazo abierto es igual a 1 cuando su fase es 0° . Es decir, se debe diseñar con un margen de fase amplio, de manera que el sistema no se vuelva inestable al presentarse alguna perturbación. En la práctica se diseña con un margen de fase de al menos 45° .

La red de compensación $A(s)$ es el elemento más crítico para la compensación del lazo de realimentación porque determina la ganancia en continua, el ancho de banda (frecuencia de corte) y los márgenes de fase y ganancia. La ganancia de compensación se describe con la Ecuación C.1, donde g_m es la ganancia de transconductancia del amplificador de error, $Z_{ith}(s)$ es la impedancia de la red (C_{th} , R_{th} y C_{thp}) y R_O es la impedancia de salida del amplificador de error.

$$A(s) = \frac{v_F(s)}{v_{FB}(s)} = g_m(Z_{ith}(s)/R_O) = g_m R_O \frac{1 + \frac{s}{2\pi f_{thz}}}{(1 + \frac{s}{2\pi f_{po}})(1 + \frac{s}{2\pi f_{thp}})} \quad (C.1)$$

En la Ecuación C.1 se pueden observar dos polos y un cero, los cuales se detallan en las Ecuaciones C.2, C.3 y C.4.

$$f_{po} = \frac{1}{R_O C_{th}} \quad (C.2)$$

$$f_{thz} = \frac{1}{R_{th} C_{th}} \quad (C.3)$$

$$f_{thp} = \frac{1}{R_{th} \frac{C_{th} C_{thp}}{C_{th} + C_{thp}}} \quad (C.4)$$

Por lo general, se escoge $C_{thp} \ll C_{th}$ obteniendo una descripción mas sencilla del polo f_{thp} , la cual se puede encontrar en la Ecuación C.5.

$$f_{thp} \approx \frac{1}{R_{th} C_{thp}} \quad (C.5)$$

El polo f_{po} es un polo de baja frecuencia, dado que la resistencia de salida R_O del amplificador de error es alta. Se contactó al fabricante del regulador (*Linear Technologies*) para obtener la resistencia de salida de dicho amplificador, la cual resulta ser $500 \text{ k}\Omega$. Este polo introduce un atraso de 90° en la fase del compensador y una pendiente de -20 dB/dec a partir de su frecuencia.

El objetivo del cero f_{thz} es aumentar la fase en la frecuencia de corte. De hecho, si este cero se ubica antes de la frecuencia de corte, se puede lograr aumentar hasta 90° la fase en la frecuencia de corte, incrementando el margen de fase y estabilizando el lazo. La desventaja de este cero es que la ganancia a alta frecuencia se ve aumentada.

Por último, el polo f_{thp} se introduce para disminuir la ganancia de alta frecuencia del lazo, por lo que se posiciona mas allá de la frecuencia de corte.

De la función transferencia C.1 se deriva el diagrama de Bode asintótico del compensador, el cual se puede visualizar en la Figura C.5.

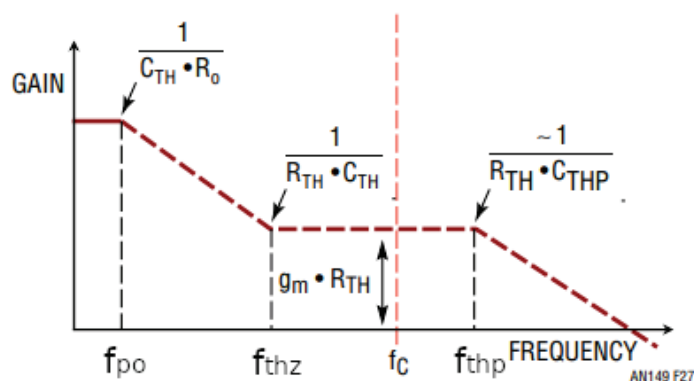


Figura C.5: Diagrama de Bode asintótico de la red de compensación. Extraído de [78].

El capacitor C_{th} afecta la ubicación del polo de baja frecuencia f_{po} y del cero f_{thz} . Un C_{th} pequeño aumenta la ganancia de baja-media frecuencia dado que este polo y cero se mueven hacia frecuencias mayores. Esto provoca que se reduzca el tiempo de establecimiento de los transitorios sin mucho impacto en el *undershoot* y el *overshoot*, pero puede disminuir la fase cerca de la frecuencia de corte elegida, perjudicando la estabilidad del sistema.

El resistor R_{th} afecta la ubicación del cero f_{thz} y del polo de alta frecuencia f_{thp} . Un valor grande de R_{th} aumenta la ganancia entre este cero y polo, aumentando el ancho de banda f_c porque la ganancia va a cruzar por cero a frecuencias mayores. Esto permite reducir el *overshoot* y *undershoot* en los transitorios. Pero, al aumentar el ancho de banda puede que el sistema se vuelva inestable al no poseer el suficiente margen de fase.

El capacitor C_{thp} afecta principalmente a la ubicación del polo de alta frecuencia f_{thp} . Principalmente, este capacitor se usa para filtrar el ruido en el terminal del integrado donde se conecta el compensador para evitar *jitter* en la frecuencia de conmutación. Por lo general, su valor se elige de manera de que el polo f_{thp} esté lejos de la frecuencia de corte, dado que si son cercanos, puede disminuir el ancho de banda y el margen de fase, aumentando el *overshoot* y el *undershoot*.

D. Detalle de las conexiones VITA 57.1

En el conector FMC del PCB se realizaron conexiones adicionales para respetar, en su mayor medida, el estándar VITA 57.1. En este anexo se encuentra el detalle de estas conexiones.

Las señales $GA[1:0]$ son utilizadas por los módulos FMC para determinar la dirección *Inter-Integrated Circuit* (I2C) de su memoria *Electrically Erasable Programmable Read-Only Memory* (EEPROM). El *carrier* puede conectar estas señales a 3P3VAUX o a GND. En el caso del TDC, la intención es que haya solamente un módulo FMC conectado al PCB, por lo tanto, es indiferente la conexión de esta señal y se conectó a GND.

La interfaz JTAG se conectó al conector JTAG de entrada, el cual es utilizado también para programar a la FPGA. Por defecto, las señales TDO y TDI no están conectadas a la interfaz VITA. Pero, a partir de dos resistores de $0\ \Omega$ se puede configurar en modo *daisy chain* la interfaz JTAG, pudiendo así configurar a la FPGA misma del TDC y a algún otro chip programable/reconfigurable que esté ubicado en el módulo FMC. Para mayor detalle de esta conexión, ver Sección 4.5.1.

El estándar define una interfaz I2C en la cual el *carrier* es el maestro y el módulo FMC el esclavo. Para futuro uso, se conectaron a dos pines de un banco de IO de la FPGA.

La señal $PRSNT_M2C_L$ permite al *carrier* determinar si un módulo FMC está conectado. Para futuro uso, se conectó a un pin de un banco de IO de la FPGA.

La señal PG_C2M permite al *carrier* informar al módulo FMC que las fuentes de alimentación VADJ, 12P0V y 3P3V se encuentran en un estado nominal. Por simplicidad, se conecta a la señal de *power good* del regulador de 3,3 V. Cabe destacar que esto no cumple la norma VITA 57.1, pero no es algo crítico para la aplicación actual. Se deja para futuras iteraciones el resolverlo de una manera más elegante.

A través del pin 12P0V el *carrier* entrega 12 V al módulo FMC. Esto se conectó a la entrada de alimentación de 4,5 V-13 V de la placa. Si bien el rango de tensiones de entrada es amplio y por lo tanto no cumple con los 12 V estrictos que indica la norma, la idea de esta entrada es que posea 12 V nominalmente. Este conexionado se realiza para que el módulo FMC también pueda acceder a esta alimentación.

El *carrier* entrega 3,3 V al módulo FMC a través del pin 3P3V. Este pin se conectó directamente a la salida del regulador de 3,3 V.

El estándar también especifica el pin 3P3VAUX por el cual el *carrier* puede brindar una alimentación de 3,3 V auxiliar al módulo FMC. Esta fuente de alimentación auxiliar no fue especificada para alimentar al módulo FMC en sí, su objetivo es poder alimentar a algunos circuitos particulares, como las memorias EEPROM que deben contener. Según el estándar, esta alimentación puede ser originada por la misma fuente que genera 3P3V y en el PCB se conectó a la salida del regulador de 3,3 V.

Por último, a través del pin VADJ el *carrier* puede exponer una alimentación entre 0 V y 3,3 V. Se utiliza generalmente para brindar una alimentación más al módulo FMC, así librándolo de tener que poseer un posible regulador extra. En el estándar se indica que si se van a utilizar interfaces LVDS en las señales LA*, este voltaje debe ser 2,5 V, por lo tanto, se conectó a la salida del regulador de 2,5 V.

Bibliografía

- [1] P. Inc, «CubeSat Kit PCB Specification,» Pumpkin Inc, Especificación, 2007. dirección: http://www.cubesatkit.com/docs/CSK_PCB.Spec-A5.pdf.
- [2] D. R. Schaart, G. Schramm, J. Nuyts y S. Surti, «Time of Flight in Perspective: Instrumental and Computational Aspects of Time Resolution in Positron Emission Tomography,» *IEEE Trans Radiat Plasma Med Sci*, 2021. DOI: 10.1109/trpms.2021.3084539. dirección: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8454900/>.
- [3] P. Dong y Q. Chen, «LiDAR Remote Sensing and Applications,» *LiDAR Remote Sensing and Applications (1st Edition)*, 2018. DOI: 10.4324/9781351233354. dirección: <https://www.taylorfrancis.com/books/mono/10.4324/9781351233354/lidar-remote-sensing-applications-pinliang-dong-qi-chen>.
- [4] T. Instruments, «TDC7200 Time-to-Digital Converter for Time-of-Flight Applications in LiDAR, Magnetostrictive and Flow Meters,» Texas Instruments, Hoja de Datos, 2016. dirección: <https://www.ti.com/lit/ds/symlink/tdc7200.pdf>.
- [5] W. Jang, Y. Chalich y M. J. Deen, «Sensors for Positron Emission Tomography Applications,» *Multidisciplinary Digital Publishing Institute*, 2019. DOI: 10.3390/s19225019.
- [6] S. Vandenberghe, E. Mikhaylova, E. D’Hoe, P. Mollet y J. S. Karp, «Recent developments in time-of-flight PET,» *EJNMMI Physics*, 2016. DOI: 10.1186/s40658-016-0138-3.
- [7] N. Lusardi, F. Garzetti, G. Bulgarini, R. B. M. Gourgues, J. W. N. Los y A. Geraci, «Single photon counting through multi-channel TDC in programmable logic,» *2012 18th IEEE-NPSS Real Time Conference*, 2016. DOI: 10.1109/NSSMIC.2016.8069673. dirección: <https://ieeexplore.ieee.org/document/8069673>.
- [8] S. A. Wadood, «Measuring the lifetime of cosmic ray Muons through an FPGA based Time to Digital Converter,» 2016. dirección: <https://www.physlab.org/wp-content/uploads/2018/07/Thesis-Sultan.compressed.pdf>.
- [9] Swabian Instruments, *Time Tagger Series - Brochure*, <https://www.swabianinstruments.com/static/downloads/TimeTaggerSeries.pdf>, Último acceso: 2023-06-04, 2023.
- [10] S. Instruments, «Time-correlated single-photon counting,» Swabian Instruments, Nota de aplicación, 2022. dirección: https://www.swabianinstruments.com/static/app_notes/appnote_SI-0001_web.pdf.
- [11] S. Quantum, «Single Quantum EOS - SNSPD Closed-Cycle System,» Single Quantum, inf. téc., 2023. dirección: https://qdusa.com/siteDocs/productBrochures/Single_Quantum_Eos_datasheet.pdf.
- [12] I. Shavrin, H. Fedder, M. Schlagmüller et al., «Characterization of Silicon Photomultipliers,» Swabian Instruments, Nota de aplicación, 2019. dirección: https://www.swabianinstruments.com/static/app_notes/appnote_SI-0002_web.pdf.
- [13] M. Nahuel, C. Matias, S. Gabriel, G. Federico y L. Pablo, «SiPM Analog Front-End Electronics For Space-Borne Applications,» *2020 Argentine Conference on Electronics (CAE)*, 2020. DOI: 10.1109/CAE48787.2020.9046371.
- [14] M. Storm, H. Cao, D. Engin y M. Albert, «Cubesat Lidar Concepts for Ranging, Topology, Sample Capture, Surface, and Atmospheric Science,» 2017. dirección: <https://digitalcommons.usu.edu/smallsat/2017/all2017/250>.
- [15] X. Sun, D. R. Cremons, E. Mazarico et al., «Small All-range Lidar for Asteroid and Comet Core Missions,» 2021. DOI: 10.3390/S21093081. dirección: https://ntrs.nasa.gov/api/citations/20210011319/downloads/SunX_SALi_Sensors_v8.pdf.
- [16] F. Yong, Z. Li, G. Hui, C. Bincai, G. Li y H. Haiyan, «Spaceborne LiDAR Surveying and Mapping,» 2022. DOI: 10.5772/intechopen.108177. dirección: <https://www.intechopen.com/online-first/84590>.

-
- [17] Julian Rodriguez, *Crono TDC Core*, ver. 1.1.0, 7 de abr. de 2023. dirección: https://gitlab.com/1895/tdc_hdl/crono_tdc_core/-/tree/v1.1.0.
- [18] Julian Rodriguez, *Crono Signal Generator*, ver. 1.0.0, 7 de abr. de 2023. dirección: https://gitlab.com/1895/tdc_hdl/crono_signal_generator/-/tree/v1.0.0.
- [19] Julian Rodriguez, *Crono HDL Common*, ver. 1.0.0, 7 de abr. de 2023. dirección: https://gitlab.com/1895/tdc_hdl/crono_hdl_common/-/tree/v1.0.0.
- [20] Julian Rodriguez, *Crono HDL CI*, ver. 1.0.0, 7 de abr. de 2023. dirección: https://gitlab.com/1895/tdc_hdl/crono_hdl_ci/-/tree/v1.0.0.
- [21] Julian Rodriguez, *Crono Testbench Tools*, ver. 1.0.0, 7 de abr. de 2023. dirección: https://gitlab.com/1895/tdc_hdl/crono_tb_tools/-/tree/v1.0.0.
- [22] Julian Rodriguez, *Crono TDC Vivado*, ver. 1.0.0, 7 de abr. de 2023. dirección: https://gitlab.com/1895/tdc_hdl/crono_tdc_vivado/-/tree/v1.0.0.
- [23] Julian Rodriguez, *Crono TDC Py*, ver. 1.0.0, 7 de abr. de 2023. dirección: https://gitlab.com/1895/tdc_hdl/crono_tdc_py/-/tree/v1.0.0.
- [24] Julian Rodriguez, *Crono TDC Board*, ver. RevA.00.00, 7 de abr. de 2023. dirección: https://gitlab.com/1895/tdc_hdl/crono_tdc_board/-/tree/RevA.00.00.
- [25] C. Ko, K. Pun y A. Gothenberg, «Vernier parallel delay-line based time-to-digital converter,» *Analog Integrated Circuits and Signal Processing* 71, 2012. DOI: 10.1007/s10470-011-9766-7.
- [26] M. T. Pasha, «All-Digital PWM Transmitters,» Division of Integrated Circuits y Systems, Department of Electrical Engineering, Linköping University, inf. téc., 2019. DOI: 10.3384/diss.diva-153729. dirección: https://www.researchgate.net/publication/348740228_All-Digital_PWM_Transmitters.
- [27] H. Wang, M. Zhang e Y. Liu, «High-Resolution Digital-to-Time Converter Implemented in an FPGA Chip,» *Applied Sciences* 7(1):52, 2017. DOI: 10.3390/app7010052.
- [28] P. Dudek, S. Szczepanski y J. V. Hatfield, «A High-Resolution CMOS Time-to-Digital Converter Utilizing a Vernier Delay Line,» *IEEE TRANSACTIONS ON SOLID-STATE CIRCUITS, VOL. 35*, 2000. DOI: 10.1109/4.823449.
- [29] S. Henzler, *Time-to-Digital Converters*. Springer, 2010.
- [30] K. Cui, Z. Ren, X. Li, Z. Liu y R. Zhu, «A high-linearity, ring-oscillator-based, Vernier time-to-digital converter utilizing carry chains in FPGAs,» *IEEE Transactions on Nuclear Science*, 2016. DOI: 10.1109/TNS.2016.2632168. dirección: <https://ieeexplore.ieee.org/document/7755750>.
- [31] W. Pan, G. Gong y J. Li, «A 20-ps Time-to-Digital Converter (TDC) Implemented in Field-Programmable Gate Array (FPGA) with Automatic Temperature Correction,» *IEEE TRANSACTIONS ON NUCLEAR SCIENCE, VOL. 61, NO. 3*, 2014. DOI: 10.1109/TNS.2014.2320325.
- [32] Q. Zhong, M. Xiangting, L. Deyuan, Y. Lei, Y. Zeen y L. Dongcang, «A high precision TDC based on a multi-phase clock,» *2012 18th IEEE-NPSS Real Time Conference*, 2012. DOI: 10.1109/RTC.2012.6418217. dirección: <https://ieeexplore.ieee.org/document/6418217>.
- [33] Xilinx, «DS181 - Artix-7 FPGAs Data Sheet: DC and AC Switching Characteristics,» Xilinx, Hoja de Datos, 2022. dirección: https://docs.xilinx.com/v/u/en-US/ds181_Artix_7_Data_Sheet.
- [34] Xilinx, «XAPP523 - LVDS 4x Asynchronous Oversampling Using 7 Series FPGAs and Zynq-7000 AP SoCs,» Xilinx, inf. téc., 2017. dirección: <https://docs.xilinx.com/v/u/en-US/xapp523-lvds-4x-asynchronous-oversampling>.
- [35] Xilinx, «UG473 - 7 Series FPGAs Memory Resources User Guide,» Xilinx, inf. téc., 2019. dirección: https://docs.xilinx.com/v/u/en-US/ug473_7Series_Memory_Resources.
- [36] Xilinx, «7 Series Product Selection Guide,» Xilinx, inf. téc., 2021. dirección: <https://www.xilinx.com/content/dam/xilinx/support/documents/selection-guides/7-series-product-selection-guide.pdf>.
- [37] R. Quattlebaum and J. Woodyatt, *Spinel Host-Controller Protocol*, <https://datatracker.ietf.org/doc/html/draft-rquattle-spinel-unified>, Último acceso: 2023-26-03, 2017.

- [38] T. Instruments, «LMH7220 High Speed Comparator with LVDS Output,» Texas Instruments, Hoja de Datos, 2013. dirección: <https://www.ti.com/lit/ds/symlink/lmh7220.pdf>.
- [39] E. Bogatin, *Signal and Power Integrity Simplified*. Boston, Ma 02116: Pearson Education, Inc, 2010.
- [40] Xilinx, «UG470 - 7 Series FPGAs Configuration Guide,» Xilinx, inf. téc., 2023. dirección: https://docs.xilinx.com/r/en-US/ug470_7Series_Config.
- [41] Xilinx, «JTAG-HS3™ Programming Cable for Xilinx® FPGAs,» Xilinx, inf. téc., feb. de 2022. dirección: https://digilent.com/reference/_media/reference/programmers/jtag-hs3/jtag-hs3_rm.pdf.
- [42] Xilinx, *Zynq UltraScale+ MPSoC ZCU102 Evaluation Kit*, <https://www.xilinx.com/products/boards-and-kits/ek-u1-zcu102-g.html>, Último acceso: 2023-26-02, 2023.
- [43] Xilinx, «UG908 - Vivado Design Suite User Guide Programming and Debugging,» Xilinx, inf. téc., 2022. dirección: https://www.xilinx.com/content/dam/xilinx/support/documents/sw_manuals/xilinx2022.1/ug908-vivado-programming-debugging.pdf.
- [44] Infineon, «S25FL064L - 64 Mb (8 MB) FL-L flash,» Infineon, Hoja de Datos, 2022. dirección: [https://www.infineon.com/dgdl/Infineon-S25FL064L_64-Mbit_\(8-Mbyte\)_3.0_V_FL-L_SPI_Flash_Memory-DataSheet-v08.00-EN.pdf?fileId=8ac78c8c7d0d8da4017d0ee2d2846996](https://www.infineon.com/dgdl/Infineon-S25FL064L_64-Mbit_(8-Mbyte)_3.0_V_FL-L_SPI_Flash_Memory-DataSheet-v08.00-EN.pdf?fileId=8ac78c8c7d0d8da4017d0ee2d2846996).
- [45] Digilent, *Schematic - Arty A7*, https://digilent.com/reference/_media/programmable-logic/arty-a7/arty-a7-e2-sch.pdf, Último acceso: 2023-13-04, 2022.
- [46] Digilent, *Schematic - Basys 3*, https://digilent.com/reference/_media/reference/programmable-logic/basys-3/basys-3_sch.pdf, Último acceso: 2023-13-04, 2014.
- [47] F. T. D. I. (FTDI), «FT231X - USB to FULL HANDSHAKE UART IC,» Future Technology Devices International (FTDI), Hoja de Datos, 2013. dirección: http://www.ftdichip.com/Support/Documents/DataSheets/ICs/DS_FT231X.pdf.
- [48] F. T. D. I. (FTDI), «AN146 - USB Hardware Design Guidelines for FTDI ICs,» Future Technology Devices International (FTDI), Nota de aplicación, 2013. dirección: https://www.ftdichip.com/Documents/AppNotes/AN_146_USB_Hardware_Design_Guidelines_for_FTDI_ICs.pdf.
- [49] Jim Lindblom, *FT231x-breakout-v11*, <http://cdn.sparkfun.com/datasheets/BreakoutBoards/ft231x-breakout-v11.pdf>, Último acceso: 2023-26-02, 2016.
- [50] F. T. D. I. (FTDI), «LC231X - Development Module,» Future Technology Devices International (FTDI), Hoja de Datos, 2017. dirección: https://ftdichip.com/wp-content/uploads/2020/07/DS_LC231X.pdf.
- [51] E. Cole, «Performance of LVDS With Different Cables,» Texas Instruments, Nota de aplicación, 2002. dirección: <https://www.ti.com/lit/an/s11a053b/s11a053b.pdf>.
- [52] *American National Standard for FPGA Mezzanine Card (FMC) Standard*, TJA1043, Approved by American National Standards Institute, Inc, VMEbus International Trade Association, jul. de 2008.
- [53] S. Inc, «High Speed Characterization Report: SEAM-XX-03.5-S-XX-2 and SEAF-XX-06.5-S-XX-2,» Samtec Inc, Reporte de ensayo, 2005. dirección: https://suddendocs.samtec.com/testreports/hsc-report-seam-seaf-10mm_web.pdf.
- [54] Xilinx, «7 Series FPGAs Data Sheet: Overview,» Xilinx, Hoja de Datos, 2020. dirección: https://docs.xilinx.com/v/u/en-US/ds180_7Series_Overview.
- [55] Xilinx, «UG475 - 7 Series FPGAs Packaging and Pinout,» Xilinx, inf. téc., 2022. dirección: https://docs.xilinx.com/v/u/en-US/ug475_7Series_Pkg_Pinout.
- [56] Abracon, «ASEM - MEMS Clock Oscillator,» Abracon, Hoja de Datos, 2022. dirección: <https://abracon.com/Oscillators/ASEM.pdf>.
- [57] Sitime, «SiT2001B - Single-Chip, SOT23 Oscillator,» Sitime, Hoja de Datos, 2018. dirección: <https://www.sitime.com/datasheet/SiT2001>.
- [58] P. E. Consortium, «PC/104-Plus Specification,» PC/104 Embedded Consortium, Especificación, 2013. dirección: https://pc104.org/wp-content/uploads/2015/02/PC104_Plus_v2_32.pdf.

- [59] Pumpkin, *CubeSat Kit PCB Specification*, http://www.cubesatkit.com/docs/CSK_PCB_Spec-A5.pdf, Último acceso: 2023-13-04, 2003.
- [60] A. R. Aslan, H. Yağcı, M. Umit et al., «Development of a LEO Communication CubeSat,» *2013 6th International Conference on Recent Advances in Space Technologies (RAST)*, 2013. DOI: 10.1109/RAST.2013.6581288.
- [61] P. Inc, «CubeSat Kit Motherboard (MB) - Single Board Computer Motherboard for Harsh Environments,» Pumpkin Inc, Hoja de Datos, 2009. dirección: http://www.cubesatkit.com/docs/datasheet/DS_CSK_MB_710-00484-D.pdf.
- [62] Xilinx, «DS187 - Zynq-7000 SoC (Z-7007S, Z-7012S, Z-7014S, Z-7010, Z-7015, and Z-7020): DC and AC Switching Characteristics,» Xilinx, Hoja de Datos, 2020. dirección: <https://docs.xilinx.com/v/u/en-US/ds187-XC7Z010-XC7Z020-Data-Sheet>.
- [63] D. D. W. G. Promoters, «Digital Visual Interface DVI,» 1999. dirección: http://www.cs.unc.edu/Research/stc/FAQs/Video/dvi_spec-V1_0.pdf.
- [64] Xilinx, «UG474 - 7 Series FPGAs Configurable Logic Block,» Xilinx, inf. téc., 2016. dirección: https://docs.xilinx.com/v/u/en-US/ug474_7Series_CLB.
- [65] S. Sirhan y S. Gupta, «Power-supply sequencing for FPGAs,» Texas Instruments, Application Report, 2014. dirección: <http://www.ti.com/lit/lyt598>.
- [66] T. Instruments, «Tips for successful power-up of today's high-performance FPGAs,» Texas Instruments, Article, 2005. dirección: https://www.ti.com/lit/an/slyt079/slyt079.pdf?ts=1663198255431&ref_url=https%253A%252F%252Fwww.google.com%252F.
- [67] A. Devices, «Dual Channel 3A, 20V Monolithic Synchronous Step-Down Regulator,» Analog Devices, Application Report, 2012. dirección: <https://www.analog.com/media/en/technical-documentation/data-sheets/3633a1fb.pdf>.
- [68] H. X. E. Guo, «How the Switching Frequency Affects the Performance of a Buck Converter,» Application Report, 2011. dirección: <https://www.ti.com/lit/an/slvaed3/slvaed3.pdf>.
- [69] S. P. Singh, «Output Ripple Voltage for Buck Switching Regulator,» Texas Instruments, Application Report, 2014. dirección: <https://www.ti.com/lit/an/slva630a/slva630a.pdf>.
- [70] A. S. Limjoco, «Measuring Output Ripple and Switching Transients in Switching Regulators,» Analog Devices, Application Report, 2022. dirección: <https://www.analog.com/media/en/technical-documentation/application-notes/an-1144.pdf>.
- [71] T. Instruments, «Understanding Dead-time Based On TPS51225/275/285,» Analog Devices, Application Report, 2018. dirección: <https://www.ti.com/lit/pdf/slua919>.
- [72] M. Xie, «How to select input capacitors for a buck converter,» Texas Instruments, Nota de aplicación, 2016. dirección: <https://www.ti.com/lit/an/slyt670/slyt670.pdf>.
- [73] A. Asinovski, «Equivalent Capacitance and ESR of Paraleled Capacitors,» Murata Power Solutions, Artículo de revista, 2014. dirección: https://www.power-mag.com/pdf/feature_pdf/1387888355_Murata_Feature_Layout_1.pdf.
- [74] Xilinx, «ARTIX 7 FPGA Development Board AX7 A User Manual,» Alinx, Guia de usuario, 2019. dirección: https://www.xilinx.com/content/dam/xilinx/support/documents/user_guides/ug483_7Series_PCB.pdf.
- [75] S. Roberts, *DC/DC Book Of Knowledge - Practical tips for the User*. Gmunden: RECOM, 2014.
- [76] R. B. Ridley, *An Accurate and Practical Small Signal Model for Current-Mode Control*. Virginia: Ridley Engineering, Inc, 1999.
- [77] V. Yang, «Peak Current Mode and Continuous Current Mode DC-to-DC Converter Modeling and Loop Compensation Design Considerations,» Analog Devices, Technical Article, 2016. dirección: <https://www.analog.com/media/en/technical-documentation/tech-articles/peak-current-mode-and-continuous-current-mode-dc-dc-converters.pdf>.
- [78] H. J. Zhang, «Modeling and Loop Compensation Design of Switching Mode Power Supplies,» Linear Technologies, Application Note, 2015. dirección: <https://www.analog.com/media/en/technical-documentation/application-notes/an149fa.pdf>.
- [79] T. Instruments, «Switch-mode power converter compensation made easy,» Texas Instruments, Application Report, 2016. dirección: <https://www.ti.com/seclit/ml/slup340/slup340.pdf>.