

Figura 1: Circuito afilador de flanco con SRD

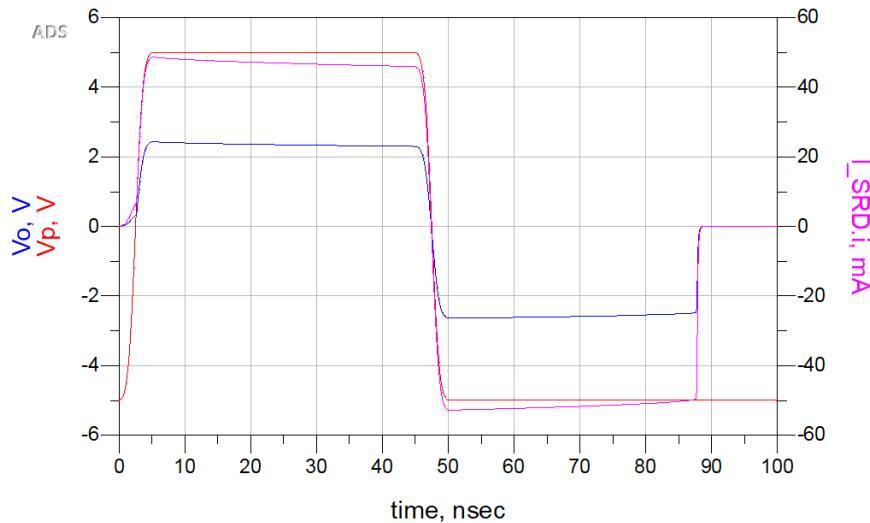


Figura 2: Resultado de simulación.

1. Diseño

1.1. Diseño del *pulser*

1.1.1. SRD como afilador de flanco

En la sección AGREGAR REF?? fue explicado como un diodo *SRD* funciona como afilador de flanco.

En la figura 1 puede observarse un circuito que demuestra el funcionamiento.

El circuito está compuesto por un generador de cuadrada lento, con tiempos de crecimiento y decrecimiento de 5 ns, en serie una resistencia de fuente R_s de valor 50 Ω. La carga del circuito es la resistencia R_L de 50 Ω.

En la figura 2 se observa el resultado de la simulación. Vemos que, hasta aproximadamente 85 ns, la señal de salida V_o es igual a la señal de entrada, afectada por el divisor entre R_L y R_s , $\frac{R_L}{R_L + R_s}$. Durante este tiempo, el *SRD* presenta una baja impedancia. En la porción positiva de la señal de entrada V_p , esto es coincidente con un diodo usual, ya que el mismo se encuentra polarizado en directa. En lo que destaca el *SRD* de un diodo usual, es que luego de que la tensión de entrada se invierta, este sigue presentando una baja impedancia. Esto se debe al gran tiempo de vida de sus portadores minoritarios, lo que requiere un tiempo apreciable para descargárselos y pasar al estado de alta impedancia.

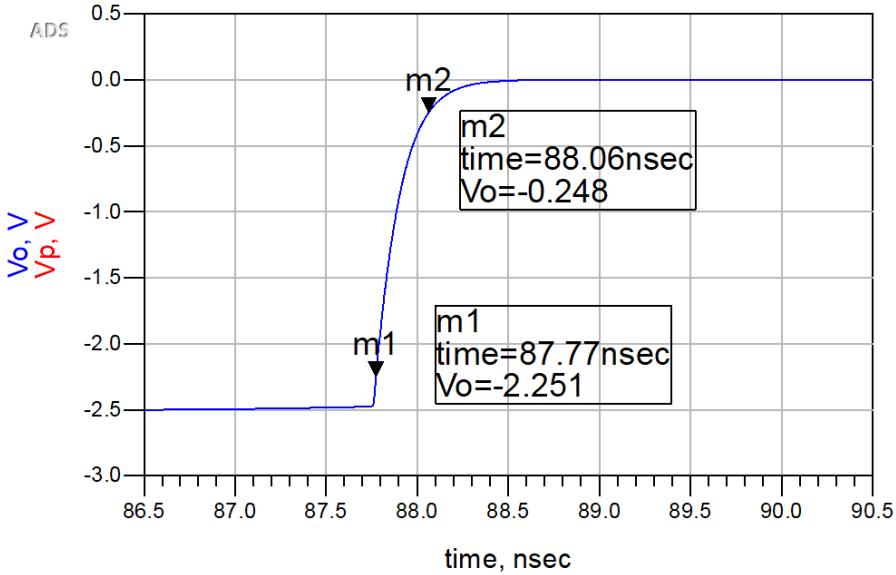


Figura 3: Tiempo de crecimiento.

Se observa en la forma de onda de V_o que esta transición se da alrededor de 85 ns, donde la tensión de salida cae abruptamente a 0. En la figura 2 puede observarse el comportamiento de la corriente. Se observa la misma caída abrupta en los 85 ns, y una inversión en el signo de la corriente con la inversión en el signo de la cuadrada de entrada.

Llamaremos corriente de inyección de carga I_F a la corriente que circula por el *SRD* con sentido positivo. Esta corriente determina la carga almacenada en el mismo, y ambas se relacionan mediante [1], [2]

$$Q_F = I_F \cdot \tau \cdot \left(1 - e^{-t_F/\tau}\right) \quad (1)$$

Para el circuito presentado, la corriente I_F estará dada por

$$I_F = \frac{V_h}{R_s + R_L} \quad (2)$$

con V_h el valor de la tensión positiva de la señal cuadrada de entrada.

Para la corriente de extracción de carga I_R , que es la corriente que extrae carga durante el tiempo en el que el *SRD* se encuentra en un estado de baja impedancia y la corriente que circula es negativa, tenemos la siguiente expresión

$$I_R = \frac{V_l}{R_s + R_L} \quad (3)$$

con V_l el valor de la tensión negativa de la señal cuadrada de entrada.

En la figura 3, puede observarse el tiempo de crecimiento del escalón generado con el apagado del *SRD*. Se toma el tiempo de crecimiento 10 %-90 %. Siendo que el escalón de tensión se da entre -2,5 V y 0 V, tenemos que el punto de 10 % es $V_{10\%} = -2,5 V \cdot 0,9 = 2,25 V$, y el de 90 % es $V_{90\%} = -2,5 V \cdot 0,1 = -0,25 V$.

En cuanto a la magnitud del salto de tensión ΔV , estará dado por el valor de la tensión en el cátodo del *SRD* antes de que pase al estado de alta impedancia. Esta tensión estará dada por el divisor entre R_L y R_s ,

$$\Delta V = V_l \cdot \frac{R_L}{R_L + R_s} \quad (4)$$



Figura 4: Reflexiones en un *stub* cortocircuitado.

Vemos por los marcadores de la figura, que estos tiempos son 87,77 ns y 88,06 ns respectivamente, por lo que tenemos un tiempo de crecimiento

$$t_r = 87,77 \text{ ns} - 88,06 \text{ ns} = 290 \text{ ps} \quad (5)$$

Como fuese explicado en la sección AGREGAR REF??, este tiempo de crecimiento estará dado por el tiempo de transición del diodo y por el tiempo del RC formado entre la capacidad de reversa del diodo y la resistencia vista desde los nodos del capacitor.

1.1.2. Generador de pulsos con *stub*

1.1.2.1 Principios del *stub*

Un *stub* consiste de una línea de transmisión conectada en paralelo al camino de la señal. Su efecto sobre la señal dependerá de su impedancia característica, largo e impedancia de terminación [3].

Cuando el *stub* se encuentra abierto, es decir, terminado por una impedancia infinita, la señal propagada se verá reflejada con signo positivo, y en el caso de una línea de transmisión sin pérdidas, con un factor de ganancia unitario. En el caso de una línea de transmisión real, las pérdidas resultaran en un factor de atenuación. *chequear esto de la atenuación, y lo q sigue de las aplicaciones de un stub abierto.* Este efecto permite generar resonancias en ciertas frecuencias, útiles para filtrado de señales o adaptación de impedancias.

En el caso de un *stub* cortocircuitado, es decir, con una impedancia de terminación igual a 0, el efecto será una reflexión de la señal con fase opuesta, y un factor de atenuación dado por las pérdidas de la línea.

El caso de interés para el circuito generador de pulsos, es el del *stub* cortocircuitado, ya que la reflexión de señal con fase opuesta, permite generar un pulso en base a una forma de onda creciente o decreciente.

En la figura ?? se observa el principio de funcionamiento. Se observa un pulso de entrada, formado por una forma de onda de tipo escalón, en nuestro caso este escalón será el mismo que en la figura ???. Este escalón se ve reflejado con polaridad opuesta, y se suma al escalón de entrada.

Dado el tiempo de propagación de la línea de transmisión T , vemos que el tiempo que tarda el pulso de entrada en reflejarse es $2 * T$, el tiempo de un camino de ida y vuelta. Dado que el pulso se forma cuando vuelve la componente reflejada, vemos que el ancho de pulso estará dado por $2 * T$. De esta forma, la duración temporal del pulso y, por lo tanto, el ancho de banda del sistema, estará dado por la longitud L del *stub*.

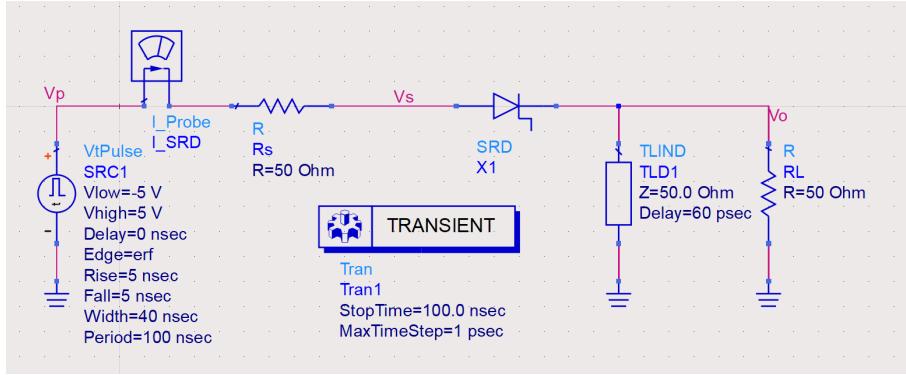


Figura 5: Generador de pulsos basado en *stub*

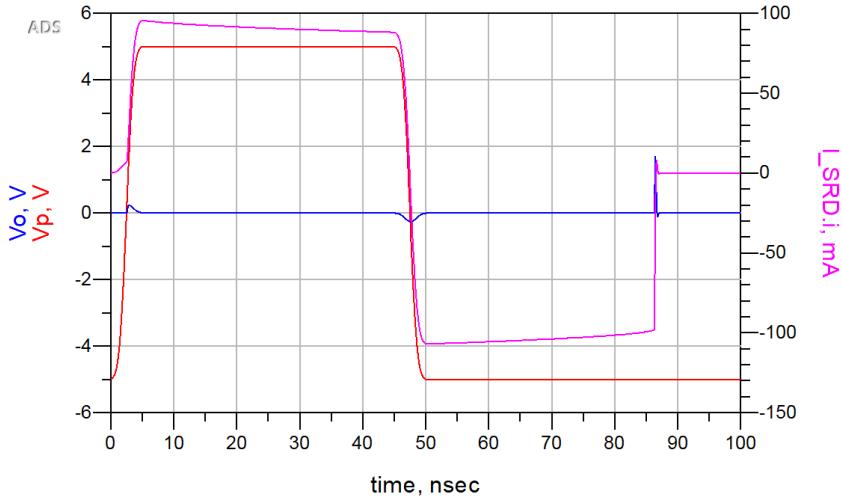


Figura 6: Resultado de simulación de generador con *stub*

Desarrollar la expresión de T .

1.1.2.2 Generador de pulsos propuesto

Un *stub* es una línea de transmisión puesta a tierra en paralelo con la señal. Una línea de transmisión puesta a tierra presenta una reflexión total de la señal transmitida, con polaridad opuesta [3].

Agregando un *stub* al afilador de pulsos descripto anteriormente, es posible formar un pulso con el flanco rápido generado por el SRD.

El *stub* actúa como una puesta a tierra para señales cuya variación temporal es mucho más lenta que el tiempo de propagación en el mismo. Para el flanco generado por el SRD, la reflexión del mismo genera un pulso. El ancho de este pulso es el doble del retardo temporal del *stub*

$$T_{pulso} = 2 \cdot T_{stub} \quad (6)$$

Vemos que el pulso tiene un ancho igual al doble del stub. vemos que se generan pulsos espurios con los flancos de la señal de entrada. Vemos que se el pulso final tiene un undershoot. vemos que la corriente baja abruptamente a 0. Vemos que ahora la corriente es V_p/R_s , R_L ya no entra en juego porque esta cortocircuitada por el stub.

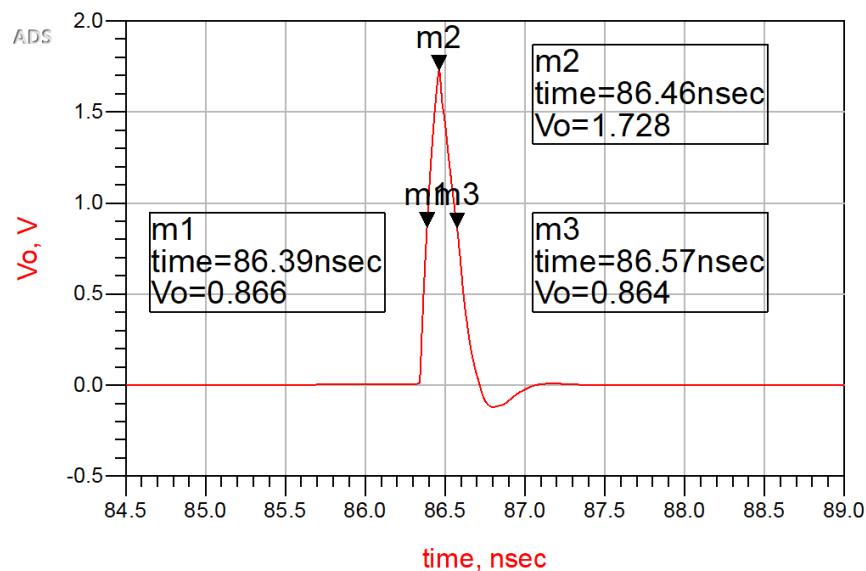


Figura 7: Pulso simulado con generador con *stub*

Hablar sobre la falta de reflexiones en el stub por ser de 50 ohm, q es la impedancia q ve una vez q se abre el SRD? ver q pasa si movemos esa Z_0 ?

1.1.3. Calculo de stub

En algún lado explicar cómo obtuvimos el largo en mm del stub.

1.1.4. Diseño final del *pulser*

Incluir diodo *Schottky*

1.2. Diseño del *driver*

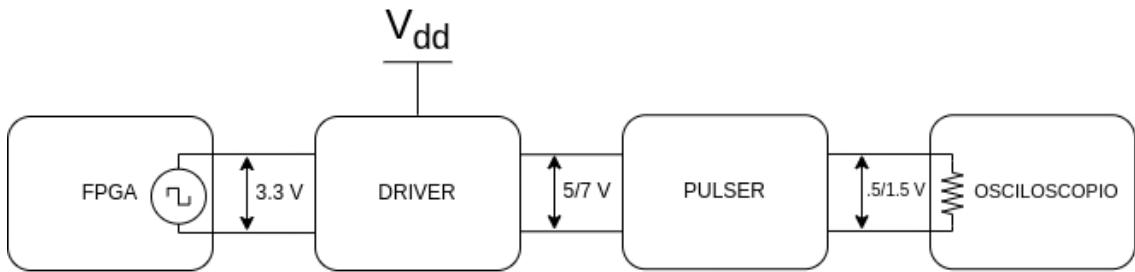


Figura 8: Banco de medición

2. Mediciones

2.1. Banco de medición

En la figura 8 puede observarse un diagrama del banco de medición. Consiste de los siguientes bloques:

- Placa FPGA: genera un pulso cuadrado unipolar, de frecuencia y ciclo de trabajo configurables. Con la frecuencia se controla la *PRF* de los pulsos de salida, y con el ciclo de trabajo los valores de tensión del pulso de salida del *driver*.
- Fuente de alimentación: provee la alimentación V_{dd} para el driver. El valor de esta tensión determina la amplitud pico a pico del pulso de salida del *driver*.
- *Driver*: cumplía la función de *buffer* para la *FPGA*, presentando una alta impedancia a la salida de la misma. Convierte el pulso unipolar de 3,3 V en uno bipolar, con amplitud pico a pico igual a V_{dd} (5 V o 7 V).
- *Pulser*: el *DUT*, genera pulsos ultra cortos en base a la salida del driver.
- Osciloscopio: instrumento de medición del experimento. Actúa como carga con su impedancia de entrada de $50\ \Omega$.

2.1.1. Fuente de alimentación

Para la fuente de alimentación se utilizó una *Marconi Instruments TF2154*, en la figura 9 puede observarse la misma.

Presentaba limitación de corriente regulable e indicadores para la amplitud y la corriente suministrada, lo que permitía trabajar de manera segura, dentro de los límites de consumo obtenidos en las simulaciones anteriores **REEMPLAZAR ESTA REF ??**.

Como fuese explicado en la sección **REEMPLAZAR ESTA REF ??**, la corriente máxima esperada en las condiciones de trabajo era menor a 200 mA, por lo que se monitoreó durante todo el experimento que la corriente entregada por la fuente no supere este máximo teórico.

2.1.2. FPGA

La *FPGA* generaba el pulso unipolar cuadrado de entrada, que controla la *PRF* y el ciclo de trabajo del pulso del driver. La placa utilizada fue *Nexys-4 DDR* de Digilent [4], con un chip *Artix-7* de Xilinx. En la figura 10 puede observarse la misma.



Figura 9: Fuente de alimentación *Marconi Instruments TF2154*.

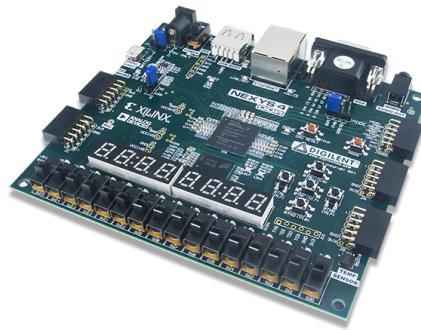


Figura 10: Placa de desarrollo *Nexys-4 DDR*.

Se utilizó una *FPGA* para poder validar la utilidad del prototipo en el contexto de un sistema UWB como el descripto en ??, en el que se dispone de señales de control digitales. Este componente del sistema es fácilmente reemplazable por otra *FPGA* o sistema embebido.

Las variables de ajuste del pulso unipolar de la *FPGA* eran las siguientes

- Frecuencia: la frecuencia de la señal cuadrada de entrada es igual a la frecuencia de repetición de pulsos (*PRF*) del sistema, ya que controla la frecuencia con la que el *SRD* se prende y se apaga y, por lo tanto, la frecuencia de generación de pulsos.
- Ciclo de trabajo: el ciclo de trabajo de la señal cuadrada unipolar determina los extremos de tensión de la señal cuadrada bipolar de salida del driver. A mayor ciclo de trabajo, valores más negativos. Este control se da a través del control del valor medio de la señal, que luego es restado por el capacitor serie del *driver*.

2.1.2.1 Diseño implementado

El diseño implementado en la *FPGA* consistía en un generador de cuadrada con ciclo de trabajo y frecuencia variables. La interfaz del sistema consistió en

- Los botones *BTNL* y *BTNR* controlaban el ciclo de trabajo en pasos de a 1 % en incrementos y decrementos respectivamente.

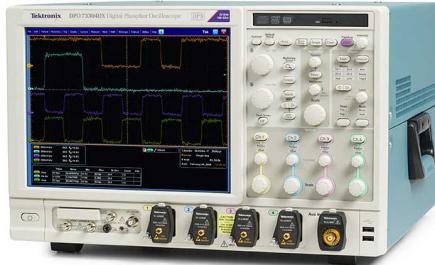


Figura 11: Osciloscopio *Tektronix MSO 70404C*

- Los botones *BTND* y *BTNU* controlaban el ciclo de trabajo en pasos de a 10% en incrementos y decrementos respectivamente.
- Con los *switches SW0* a *SW1* se controlaba la frecuencia del pulso unipolar.
 - Con *SW0* seleccionado, la frecuencia era de 1 MHz.
 - Con *SW1* seleccionado, la frecuencia era de 5 MHz.
 - Con *SW2* seleccionado, la frecuencia era de 10 MHz.

En el anexo A se encuentra el *HDL* del diseño implementado.

2.1.3. Osciloscopio

El osciloscopio fue utilizado para realizar la medición en el dominio del tiempo del pulso generado. Para una medición exitosa, era indispensable que este instrumento cuente con los requerimientos de ancho de banda del pulso. Como fuese explicado en REEMPLAZAR ESTA REF ??, el ancho de banda esperado para el pulso era de 3,6 GHz.

El osciloscopio utilizado fue *Tektronix MSO 70404C*, en la figura 11 puede observarse el mismo. El instrumento posee 4 GHz de ancho de banda analógico, y una tasa de muestreo de 25 GS/s, con la posibilidad de realizar muestreo en tiempo equivalente [5]. Estas prestaciones eran suficientes para medir el pulso de salida.

El instrumento posee configuraciones de impedancia de entrada seleccionables entre 50Ω y $500\text{ M}\Omega$ [5]. Para la medición del prototipo, se seleccionó la entrada de 50Ω , actuando esta impedancia como carga del generador de pulsos.

2.1.3.1 Seguridad del instrumento

Debido a las prestaciones del osciloscopio, era fundamental garantizar la integridad del mismo en la medición del prototipo. Dado que actuaba como carga del *DUT*, se debía garantizar que bajo todas las condiciones de trabajo, la potencia entregada por el generador de pulsos se encuentre dentro de los límites determinados por el fabricante del equipo para evitar posibles daños.

La máxima tensión de entrada se especifica en 5 V_{RMS} para una resolución $\geq 100\text{ mV/div}$ y 1 V_{RMS} para una resolución $< 100\text{ mV/div}$. Para garantizar la seguridad del equipo en cualquier caso, se toma como límite el valor de peor caso, 1 V_{RMS} (correspondiente a una resolución $\geq 100\text{ mV/div}$, para resoluciones menores a esta el límite es mayor).

En condiciones normales de funcionamiento, la potencia disipada por la carga es mínima, ya que es la potencia que disipa el tren de pulsos en un carga de 50Ω . Como fuese desarrollado en REEMPLAZAR ESTA REF ??, esta potencia está acotada por 9999 mW , que en 50Ω resultan en 9999 V_{rms} , que se encuentran muy por debajo de los 1 V_{RMS} especificados por el fabricante.

No solo es necesario analizar la disipación de potencia en condiciones normales de funcionamiento, sino también para el caso de una falla, ya que el principal objetivo es garantizar la integridad del instrumento en cualquier condición.

En caso de ocurrir alguna falla con algún componente del circuito, el *stub* de salida provee una función de protección. Este componente, para señales con una variación temporal mucho mayor al largo del mismo, actúa como una puesta a tierra.

Entonces, la componente de continua a la salida del generador de pulsos tiene un valor esperado de 0 V, tanto para condiciones normales de funcionamiento como en presencia de fallas.

En cuanto a la componente alterna de la salida, su valor esperado es extremadamente bajo, ya que únicamente señales de gran ancho de banda pueden ser filtradas y permanecer con una amplitud considerable a la salida del *stub*.

2.2. Mediciones realizadas

Las mediciones consistieron en mediciones en el dominio del tiempo del pulso de salida. Utilizando funciones provistas por el osciloscopio, se midieron tiempo de crecimiento, tiempo de decaimiento, amplitud máxima, y ancho a medio máximo (*FWHM* del inglés *Full Width at Half Maximum*).

Se realizaron distintas mediciones para distintas condiciones de trabajo del circuito. Se barrió para el pulso digital de entrada, el ciclo de trabajo, y para la fuente de alimentación distintos valores de tensión.

- Para la amplitud de la fuente, se utilizaron valores de 5 V y 7 V.
 - 5 V por ser un valor fácilmente obtenible en los sistemas *UWB* de referencia.
 - 7 V por ser la máxima amplitud tolerable por el circuito. Tensiones de alimentación mayores a estas resultan en corrientes de polarización mayores a las máximas admisibles dado los dimensionamientos de las pistas de los *PCBs*.
- El ciclo de trabajo se barrió entre 50 % y 70 %.
 - Se tomó 50 % como límite inferior por ser un valor fácilmente obtenible como división de un reloj digital.
 - Se tomó 70 % como límite superior ya que se observó que valores superiores a este resultaban en un pulso bipolar con amplitudes negativas decrecientes, y por lo tanto, amplitudes de pulso decrecientes.
 - La teoría no indicaba un límite superior para el ciclo de trabajo. Sin embargo, este se observó en la práctica debido a no idealidades en el pulso de salida del driver, que no era perfectamente cuadrado.

En la figura 12 puede observarse el *pulser* junto con el *driver* y la *FPGA*.

2.2.1. Mediciones preliminares

Previo a las mediciones principales, se realizó una medición de la salida del driver, con el objetivo de validar el pulso bipolar generado.

El motivo de esta medición previa, fue la limitada disponibilidad del osciloscopio de gran ancho de banda utilizado para la medición final del pulso. Esta pre-medición del pulso bipolar se realizó con un osciloscopio de bajo ancho de banda, ya que el objetivo era validar los niveles de tensión del pulso, y su correcta variación con la variación del ciclo de trabajo del pulso unipolar.

En la figura 13 puede observarse el banco de medición. Los resultados fueron los esperados y, por lo tanto, no se requirió ninguna iteración sobre la implementación del driver.



Figura 12: *FPGA, driver y pulser.*

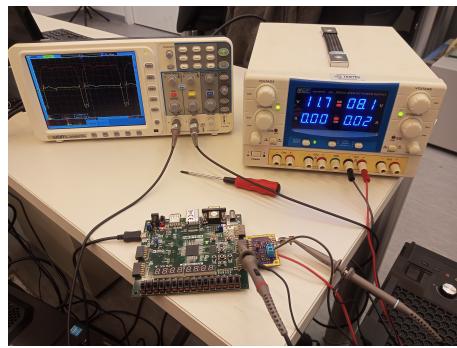


Figura 13: Banco de mediciones previas a la medición final del pulso.

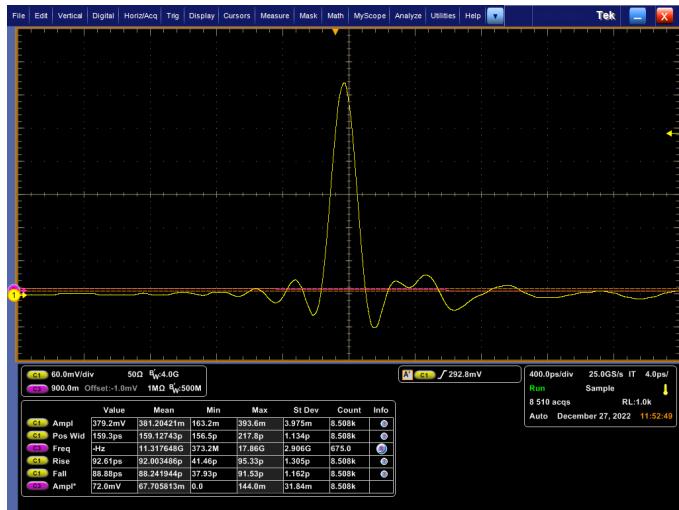


Figura 14: Salida @ V_{cc} 5 V, D 50 %

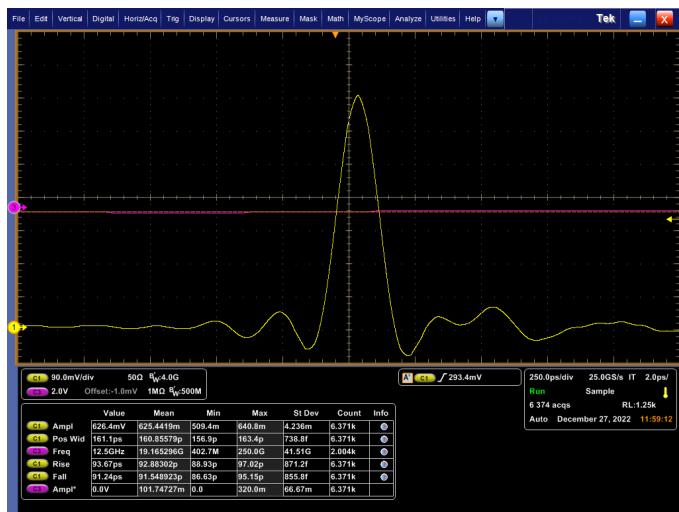


Figura 15: Salida @ V_{cc} 5 V, D 70 %

2.3. Resultados

En las figuras 14, 15, 16, 17, 18 pueden observarse los resultados en diversas capturas de pantalla tomadas del osciloscopio.

Se observó en las mediciones una amplitud de pulso creciente con mayor ciclo de trabajo y mayor amplitud de pulso, como era esperado. La menor amplitud de pulso obtenida fue de 380 mV para un V_{cc} de 5 V y un D de 50 %, y la mayor fue de 1,12 V para un V_{cc} de 7 V y un D de 70 %

En cuanto al ancho de pulso, se mantuvo aproximadamente constante en 160 ps, al igual que los tiempos de crecimiento y decrecimiento, que se mantuvieron constantes en 90 ps. Este resultado es el esperado para un *pulser* basado en un *stub*, ya que el ancho de pulso está determinado por el largo del *stub*.

En la tabla 1 pueden observarse los resultados obtenidos. Para el ancho de banda, se utiliza el obtenido a partir de la *PSD* del pulso medido. En la sección 2.3.1 se detalla cómo fue obtenido este valor.

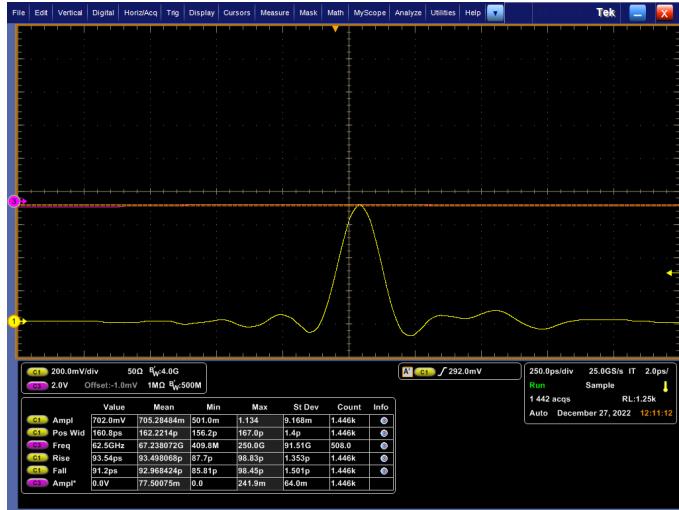


Figura 16: Salida @ V_{cc} 7V, D 50%

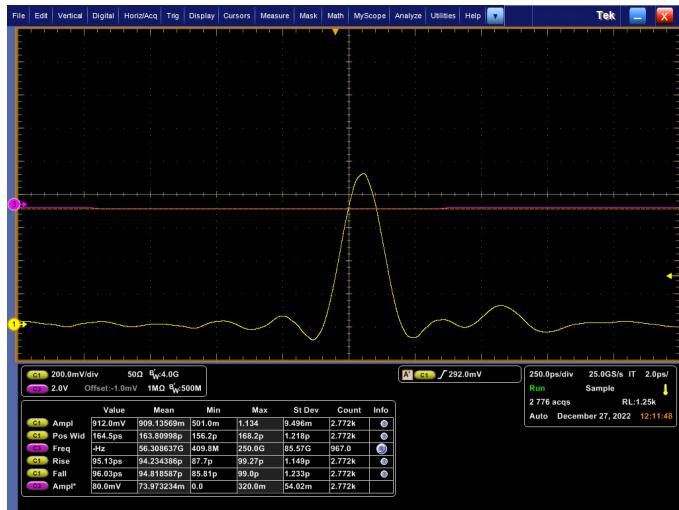


Figura 17: Salida @ V_{cc} 7V, D 60%

V_{cc} [V]	D [%]	A [V]	FWHM [ps]	3 dB B [GHz]	t_r [ps]	t_f [ps]
5	50	0.380	159	7.5	93	88
5	70	0.625	161	3.6	93	91
7	50	0.702	162	4	93	93
7	60	0.909	164	4	94	95
7	70	1.120	165	2.8	95	96

Cuadro 1: Resultados de mediciones.

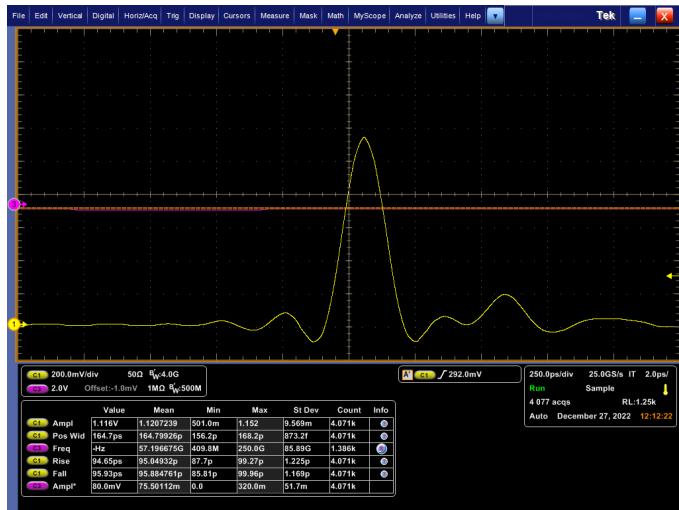


Figura 18: Salida @ V_{cc} 7 V, D 70 %

2.3.1. Comparación con simulación

En las figuras 19 a 28 pueden observarse los resultados de las mediciones obtenidas superpuestos con los resultados de simulación para las mismas condiciones de trabajo (amplitud de alimentación y ciclo de trabajo).

Para las simulaciones, se toman dos resultados:

- Una simulación “ideal”, indicada como “esquemático ideal” en las leyendas, que se corresponde a una simulación sin contemplar parásitos de ningún tipo.
- Una simulación “real”, en las leyendas “Layout”, una simulación en la que se extrajeron previamente los efectos parásitos del *PCB* mediante una simulación electromagnética y se incorporaron en la simulación del pulso.

Se realizan las comparaciones en el dominio del tiempo y de la frecuencia. Las comparaciones en el dominio del tiempo consisten en la superposición del pulso medido con los simulados. Para las comparaciones en el dominio de la frecuencia, se calculó el espectro de cada una de las formas de onda del dominio del tiempo. Para reducir el *leakage* espectral, se utilizó una ventana de *Hanning* [6].

En el dominio del tiempo, se observa una buena coincidencia entre la amplitud de los pulsos y el ancho. Se observa una diferencia en el *ringing* de ambos. Las simulaciones prácticamente no presentan oscilaciones alrededor del pulso, mientras que las mediciones las presentan tanto previa como posteriormente. También se observa un segundo pulso de menor amplitud siguiendo al primero.

Como causa de estas discrepancias, se descarta un efecto del *PCB* no modelado, ya que los parásitos de esta estructura fueron extraídos por una simulación electromagnética, y sus efectos contemplados en las simulaciones del *layout*.

Estas discrepancias sugieren una limitación en el modelado de alguno de los dispositivos, tanto el SRD como el Schottky. Las simulaciones predijeron correctamente la amplitud y el ancho de los pulsos resultantes, pero fallaron en predecir el *ringing* y el pulso secundario.

2.3.2. Comparación con resultados de la literatura

En la tabla 2 se resumen resultados reportados para generadores de pulsos *UWB* en la literatura. En la figura 29 se observan los valores de amplitud y duración reportados en un gráfico de dispersión.

V_{cc} : 5V, duty: 50%

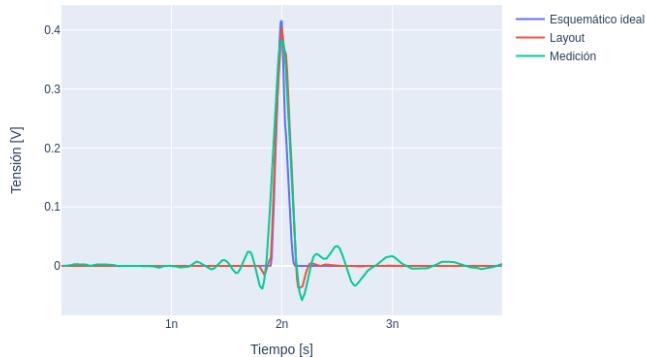


Figura 19: Pulso @ V_{cc} 5 V, D 50 %

PSD, V_{cc} : 5V, duty: 50%

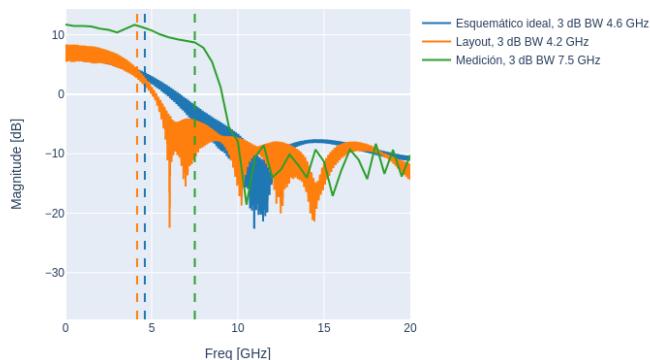


Figura 20: PSD @ V_{cc} 5 V, D 50 %

V_{cc} : 5V, duty: 70%

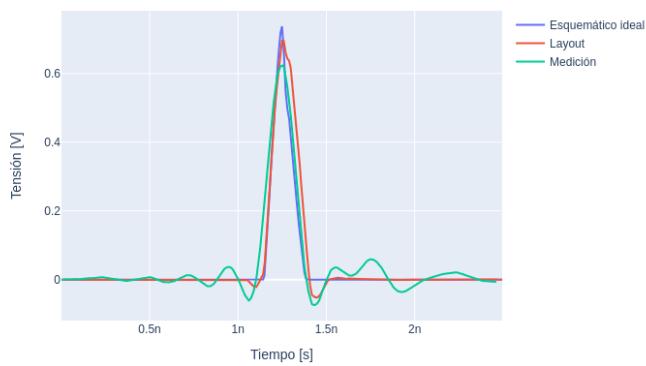


Figura 21: Pulso @ V_{cc} 5 V, D 70 %

PSD, V_{cc} : 5V, duty: 70%

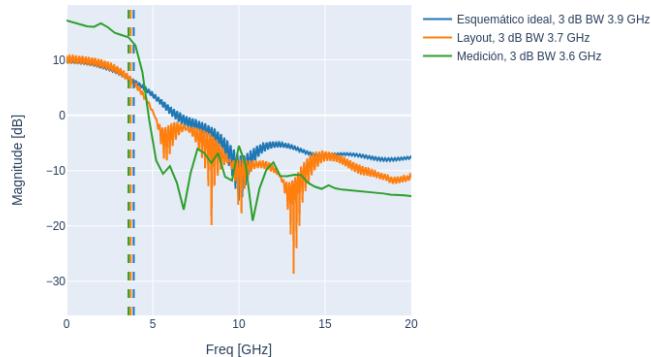


Figura 22: PSD @ V_{cc} 5 V, D 70 %

V_{cc} : 7V, duty: 50%

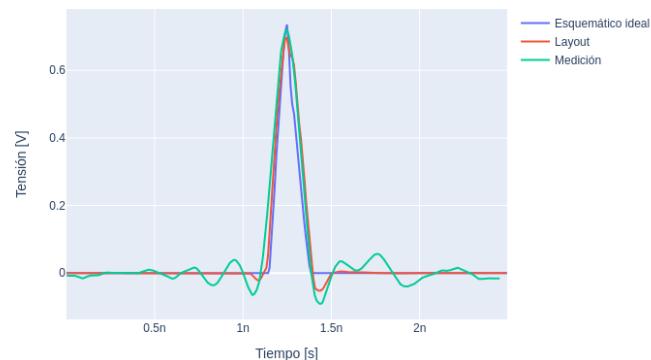


Figura 23: Pulso @ V_{cc} 7 V, D 50 %

PSD, V_{cc} : 7V, duty: 50%

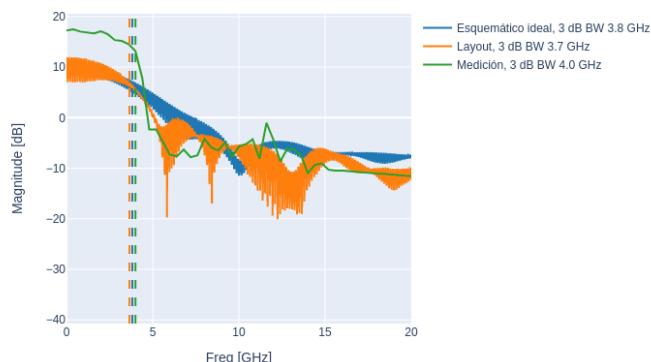


Figura 24: PSD @ V_{cc} 7 V, D 50 %

V_{cc} : 7V, duty: 60%

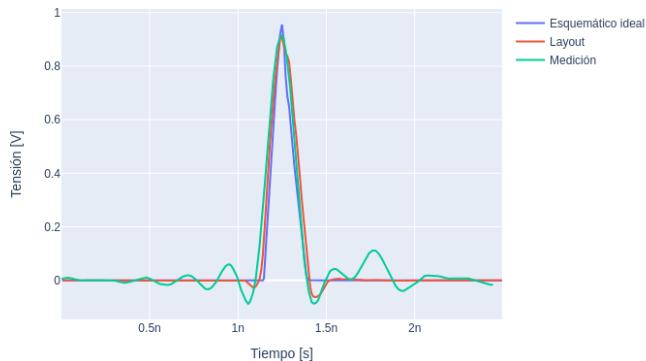


Figura 25: Pulso @ V_{cc} 7 V, D 60 %

PSD, V_{cc} : 7V, duty: 60%

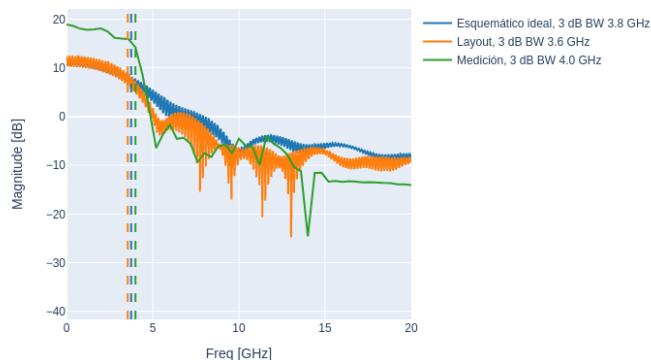


Figura 26: PSD @ V_{cc} 7 V, D 60 %

V_{cc} : 7V, duty: 70%

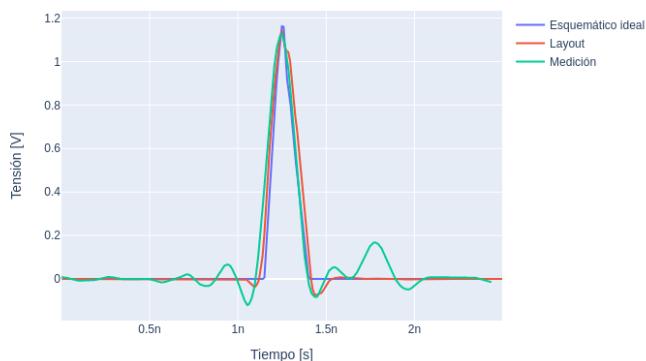


Figura 27: Pulso @ V_{cc} 7 V, D 70 %

PSD, Vcc: 7V, duty: 70%

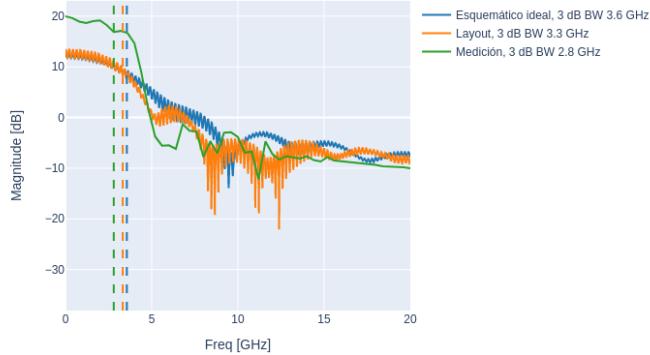


Figura 28: PSD @ V_{cc} 7V, D 70 %

Cuadro 2: Resultados reportados en la literatura

Referencia	A [V]	$FWHM$ [ps]	Bal ^a	Bias	Dispositivos	V_{cc} [V]	V_{in} [V]	PRF [MHz]
[7]	$\pm 0,896, \pm 1,6$ ^b	335, 511	Sí	Int	SRD	5	TTL	50
[8]	-7,5	110	No	Ext	SRD+3TBJ+SD	12	TTL	5
[9]	0,8	170	No	Int	SRD	4	4	10
[10]	0,2	300	No	Ext	SRD+2SD	?	?	10
[11]	-6, -4	150	No	Int	SRD+L	?	5	12
[12]	1,27 ^c	286	No	Int	2SRD+L	10	10 ^d	?
Este trabajo	1,12	165	No	Int	SRD+SD	7	CMOS	10

^a Balanceado.

^b la publicación presenta dos resultados, correspondientes a circuitos con componentes concentrados y distribuidos.

^c la publicación presenta múltiples resultados, se muestran los mejores.

^d la señal de entrada es senoidal.

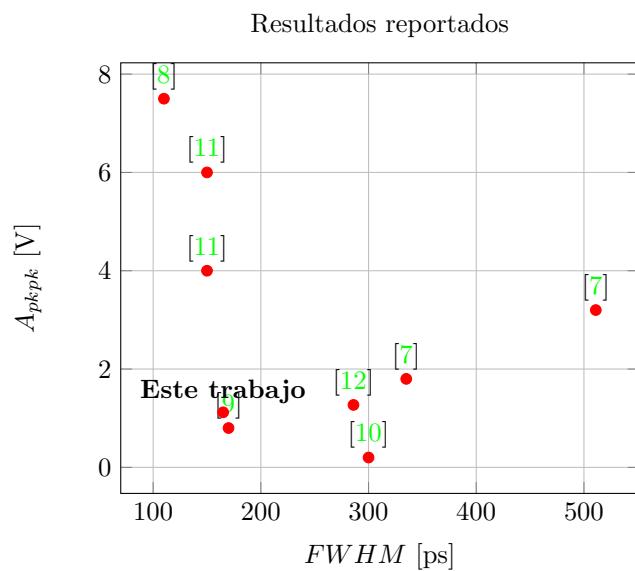


Figura 29: Diagrama de dispersión de resultados reportados.

En cuanto a los resultados reportados en este trabajo, se obtuvo uno de los anchos de pulso más bajos, existiendo otros trabajos que reportan el mismo o menor ancho de pulso con mayor amplitud, pero también mayor complejidad. Otra característica a destacar es la simplicidad del diseño implementado, tanto en cantidad de componentes activos, como en requisitos de fuente de alimentación y pulso de entrada.

En [7] se presenta un diseño compuesto de un solo SRD en el que se desarrolla un pulso balanceado a la salida. Se presentan dos diseños, uno con componentes distribuidos y otro con componentes concentrados. En ambos casos se obtienen amplitudes pico a pico de pulso mayores a las de este trabajo. Es destacable que se obtienen mayores amplitudes usando una fuente de alimentación menor, de 5 V. En cuanto al ancho de pulso, ambos pulsos presentan duraciones mayores que las de nuestro trabajo. En cuanto a la complejidad, el *pulser* está implementado solamente con 1 SRD y componentes distribuidos o concentrados, dependiendo de la versión, por lo que es más simple que nuestro trabajo. El *driver* presenta la misma complejidad en ambos casos, ya que está implementado con un solo circuito integrado. Nuestro trabajo presenta más versatilidad, ya que la amplitud de la fuente de alimentación puede variarse entre 0 V y 30 V, mientras que el integrado utilizado en el driver de [7] trabaja con 5 V fijos.

En [8] se reporta un resultado de gran amplitud y duración de pulso menor a la de este trabajo. La complejidad del diseño implementado es mucho mayor: necesita una alimentación de 12 V y una corriente de *bias* externa, y la etapa *driver* está implementada con 3 TBJs, frente a 1 solo *gate driver* en nuestro trabajo.

En [9] el pulso reportado es de características muy similares a las de nuestro trabajo. La duración del pulso es prácticamente la misma, mientras que en amplitud nuestro trabajo logró una mayor en un 40 %. Nuestro trabajo logró para el *pulser* una complejidad menor a la reportada en [9], ya que en nuestro caso omitimos la red RC y el atenuador. En cuanto a etapa *driver*, [9] no presenta ninguna, en los resultados se reporta haber utilizado como entrada al *pulser* un pulso bipolar.

En [10] el generador presentado desarrolla pulsos monociclo, que son la primera derivada de un pulso gaussiano. Nuestro trabajo logró una amplitud de pulso mayor y también una duración de pulso menor. No se especifica el valor de amplitud de señal de entrada utilizado. La entrada al circuito es bipolar, y no se incluye un *driver* de adaptación de pulso. La complejidad del generador descripto es mayor que la de este trabajo, utilizando bias externo, un diodo SRD, dos Schottky y una red RC. Sin embargo, el generador de [10] implementa un monociclo que es una derivada de un pulso gaussiano como el desarrollado en nuestro trabajo, por lo que es natural que la complejidad sea mayor.

El resultado reportado en [11] consiste en pulsos de mayor amplitud y menor duración a los de nuestro trabajo. En ambos casos, el pulser se encuentra acoplado por *AC*, lo que vuelve más complejo el diseño. Se presentan dos generadores, uno con línea de transmisión, y otro con un inductor, que utiliza al SRD en paralelo. El diseño con inductor es más complejo, ya que este se encuentra en el camino del pulso por lo que debe ser seleccionado con cuidado. En ambos casos se utiliza una red RC paralelo, mientras que en nuestro trabajo no.

El resultado de [12] consiste en un pulso de mayor amplitud y mayor ancho al de nuestro trabajo. La complejidad del diseño es mayor, ya que utiliza dos SRD y un inductor que se encuentra en el camino de alta frecuencia, por lo que es costoso de seleccionar. La señal de entrada es una senoidal, que para el contexto en el que desarrollamos nuestro generador, es más costosa de conseguir, ya que requiere algún DAC, frente a la excitación de nuestro generador que es una señal cuadrada, fácilmente obtenible como salida digital de una FPGA o microcontrolador.

Referencias

- [1] Hewlett-Packard, “Pulse and Waveform Generation with Step Recovery Diodes,” Tech. Rep. 918, Hewlett-Packard, 1984.

- [2] J. Moll and S. Hamilton, “Physical modeling of the step recovery diode for pulse and harmonic generation circuits,” *Proceedings of the IEEE*, vol. 57, no. 7, pp. 1250–1259, 1969.
- [3] D. Pozar, *Microwave Engineering, 4th Edition*. Wiley, 2011.
- [4] Digilent, “Nexys 4 DDR.” Página Oficial, 2023. Disponible online en <https://digilent.com/reference/programmable-logic/nexys-4-ddr/start>.
- [5] Tektronix, Inc., “Osciloscopio MSO 70404C.” Datasheet, 2023. Disponible online en <https://download.tek.com/datasheet/DPO-DSA-MSO70000-DataSheet-EN-11Apr23.pdf>.
- [6] A. V. Oppenheim, R. W. Schafer, and J. R. Buck, *Discrete-Time Signal Processing*. Upper Saddle River, NJ: Prentice Hall, 2nd ed., 1999.
- [7] P. Rulikowski and J. Barrett, “Truly balanced step recovery diode pulse generator with single power supply,” in *Proceedings. 2004 IEEE Radio and Wireless Conference (IEEE Cat. No.04TH8746)*, pp. 347–350, 2004.
- [8] P. Protiva, J. Mrkvica, and J. Macháč, “A compact step recovery diode subnanosecond pulse generator,” *Microwave and Optical Technology Letters*, vol. 52, no. 2, pp. 438–440, 2010.
- [9] A. Kamal, A. Bhattacharya, M. Tamrakar, and C. Roy, “Low-ringing and reduced-cost step recovery diode based uwb pulse generators for gpr applications,” *Microwave and Optical Technology Letters*, vol. 56, no. 10, pp. 2289–2294, 2014.
- [10] J. Han and C. Nguyen, “A new ultra-wideband, ultra-short monocycle pulse generator with reduced ringing,” *IEEE Microwave and Wireless Components Letters*, vol. 12, no. 6, pp. 206–208, 2002.
- [11] J. Han and C. Nguyen, “Coupled-slotline-hybrid sampling mixer integrated with step-recovery-diode pulse generator for uwb applications,” *IEEE Transactions on Microwave Theory and Techniques*, vol. 53, no. 6, pp. 1875–1882, 2005.
- [12] D. Oloumi and E. Fear, “A picosecond pulse generator using srd diodes: Design, analysis, and measurements,” in *2018 USNC-URSI Radio Science Meeting (Joint with AP-S Symposium)*, pp. 159–160, 2018.

A. Código Fuente Verilog

```

'timescale 1ns / 1ps

module top(
    input clk,
    input btnU, btnD,
    input btnL, btnR,
    input [1:0] sw,
    output [1:0] led,
    output [0:0] JA
);

    wire duty_inc_coarse, duty_inc_fine, duty_dec_coarse, duty_dec_fine;

    localparam W = 8;
    reg [W-1:0] div_value = 'd10;
    reg [1:0] sw_old = 'd0;
    reg initialized = 'b0;
    reg [3:0] init_counter = 'b0;

```

```

reg srst = 'b0;

localparam real DUTY_CYCLE_NOMINAL = 0.5;
localparam real DUTY_CYCLE_COARSE = 0.1;
localparam real DUTY_CYCLE_FINE = 0.01;

localparam integer DIV_VALUE_10MHZ = 10;
localparam integer COUNTS_DUTY_NOMINAL_10MHZ = DUTY_CYCLE_NOMINAL *
    DIV_VALUE_10MHZ;
localparam integer COUNTS_DUTY_COARSE_10MHZ = DUTY_CYCLE_COARSE *
    DIV_VALUE_10MHZ;
localparam integer COUNTS_DUTY_FINE_10MHZ = DUTY_CYCLE_FINE *
    DIV_VALUE_10MHZ;

localparam integer DIV_VALUE_5MHZ = 20;
localparam integer COUNTS_DUTY_NOMINAL_5MHZ = DUTY_CYCLE_NOMINAL *
    DIV_VALUE_5MHZ;
localparam integer COUNTS_DUTY_COARSE_5MHZ = DUTY_CYCLE_COARSE *
    DIV_VALUE_5MHZ;
localparam integer COUNTS_DUTY_FINE_5MHZ = 'd1;

localparam integer DIV_VALUE_1MHZ = 100;
localparam integer COUNTS_DUTY_NOMINAL_1MHZ = DUTY_CYCLE_NOMINAL *
    DIV_VALUE_1MHZ;
localparam integer COUNTS_DUTY_COARSE_1MHZ = DUTY_CYCLE_COARSE *
    DIV_VALUE_1MHZ;
localparam integer COUNTS_DUTY_FINE_1MHZ = DUTY_CYCLE_FINE *
    DIV_VALUE_1MHZ;

reg [W-1:0] div_value_curr,
            counts_duty_nominal_curr,
            counts_duty_coarse_curr,
            counts_duty_fine_curr;

always @(posedge clk) begin
    if (sw[1]) begin
        div_value_curr      <= DIV_VALUE_1MHZ;
        counts_duty_nominal_curr <= COUNTS_DUTY_NOMINAL_1MHZ;
        counts_duty_coarse_curr <= COUNTS_DUTY_COARSE_1MHZ;
        counts_duty_fine_curr <= COUNTS_DUTY_FINE_1MHZ;
    end
    else if (sw[0]) begin
        div_value_curr      <= DIV_VALUE_5MHZ;
        counts_duty_nominal_curr <= COUNTS_DUTY_NOMINAL_5MHZ;
        counts_duty_coarse_curr <= COUNTS_DUTY_COARSE_5MHZ;
        counts_duty_fine_curr <= COUNTS_DUTY_FINE_5MHZ;
    end
    else begin
        div_value_curr      <= DIV_VALUE_10MHZ;
        counts_duty_nominal_curr <= COUNTS_DUTY_NOMINAL_10MHZ;
        counts_duty_coarse_curr <= COUNTS_DUTY_COARSE_10MHZ;
        counts_duty_fine_curr <= COUNTS_DUTY_FINE_10MHZ;
    end
    sw_old <= sw;
end

always @(posedge clk) begin
    if (initialized == 'b0) begin

```

```

        if (init_counter < 'd14) begin
            init_counter <= init_counter+1;
        end
        else begin
            initialized <= 'b1;
            srst <= 'b1;
        end
    end
    else begin
        if (sw_old != sw) srst <= 'b1;
    end
    if (srst) srst <= 'b0;
end

// debounce of buttons
debounce u_debounce_inc_coarse(clk,btnU,duty_inc_coarse);
debounce u_debounce_dec_coarse(clk,btnD,duty_dec_coarse);
debounce u_debounce_inc_fine(clk,btnR,duty_inc_fine);
debounce u_debounce_dec_fine(clk,btnL,duty_dec_fine);

// PWM generator
adhoc_generator#.WIDTH(W)u_generator(
    .clk(clk),
    .srst(srst),
    .duty_coarse(counts_duty_coarse_curr),
    .duty_fine(counts_duty_fine_curr),
    .duty_nominal(counts_duty_nominal_curr),
    .div_value(div_value_curr),
    .duty_inc_coarse(duty_inc_coarse),
    .duty_inc_fine(duty_inc_fine),
    .duty_dec_coarse(duty_dec_coarse),
    .duty_dec_fine(duty_dec_fine),
    .PWM(JA[0])
);
assign led = sw;

endmodule

module debounce(
    input clk,
    input data_in,
    output data_out
);
localparam DEBOUNCE_BITS = 23;

reg [DEBOUNCE_BITS-1:0] debounce_counter;
wire debounce_enable;
wire tmp_1, tmp_2;

// debounce enable generation, has period T_clk/2**DEBOUNCE_BITS
always @ (posedge clk) debounce_counter = debounce_counter + 'b1;
assign debounce_enable = debounce_counter == 2**DEBOUNCE_BITS-1 ? 'b1
: 'b0;

// debounce of buttons
DFF u_DFF_inc_coarse(clk,debounce_enable,data_in,tmp_1);

```

```

        DFF u_DFF_dec_coarse(clk,debounce_enable,tmp_1,tmp_2);

        assign data_out = tmp_1 & (~tmp_2) & debounce_enable;

endmodule

module DFF(
    input clk,
    input en,
    input D,
    output reg Q
);
    always @(posedge clk) begin
        if (en) Q <= D;
    end
endmodule

'timescale 1ns / 1ps

module adhoc_generator#(
    parameter WIDTH = 8
) (
    input clk, srst,
    input duty_inc_coarse, duty_inc_fine,
    input duty_dec_coarse, duty_dec_fine,
    input [WIDTH-1:0] duty_coarse, duty_fine, duty_nominal, div_value,
    output reg PWM
);

reg [WIDTH-1:0] counter      = 'd0;
reg [WIDTH-1:0] duty_cycle   = 'd0;
wire n_PWM;

assign n_PWM = counter < duty_cycle ? 'b1 : 'b0;

always @(posedge clk) begin
    if (srst) begin
        duty_cycle  <= duty_nominal;
        counter     <= 'd0;
        PWM         <= 'd1;
    end
    else begin
        counter           <= counter >= div_value-1
            ? 'd0 : counter + 1;
        PWM             <= n_PWM;
        if      (duty_inc_coarse) duty_cycle <= duty_cycle +
            duty_coarse;
        else if (duty_inc_fine)   duty_cycle <= duty_cycle + duty_fine
            ;
        else if (duty_dec_coarse) duty_cycle <= duty_cycle -
            duty_coarse;
        else if (duty_dec_fine)   duty_cycle <= duty_cycle - duty_fine
            ;
    end
end
end

endmodule

```