Universidad Nacional del Sur

Proyecto Final de Carrera

Implementación de un TDC en FPGA basado en TDLs para la calibración de una línea digital de retardos.

Filsinger Miqueas

Contents

Introducción

Un TDC (Time to Digital Converter) es un circuito electrónico capaz de medir un intervalo de tiempo y producir un valor digital. Si buscamos medir el intervalo temporal entre una señal y el clock entonces lo llamaremos *síncrono*, si el intervalo puede comenzar y terminar en cualquier momento entonces lo llamaremos *asíncrono*. La resolución temporal es un factor importante, si se busca más allá de la alcanzada por el clock, es decir queremos medir intervalos mucho más pequeños, entonces surge el problema de implementar alguna estrategia que logren interpolar la medición con suficiente precisión y linealidad. Esto fue resuelto primeramente en ASICs, pero con el último desarrollo en FPGAs se pudo estrechar la brecha [1], manteniendo las ventajas de flexibilidad que nos permite realizar una implementación en FPGA.

La arquitectura TDL (*tapped delay line*) es la más típicamente utilizada, tal vez gracias a su simplicidad de análisis. Se ingresa la señal *hit* en una cadena de retardos, la salida de cada retardo está también conectada a un registro y el resultado es registrado al disparar la señal *stop*, de forma que los 1s y 0s registrados indican la profundidad temporal alcanzada por la señal dentro de la línea. De esta forma el resultado se presenta en los registros como de código unario (*thermometer*) y debe ser decodificado a binario, lo que traera algunos problemas que expondremos luego.

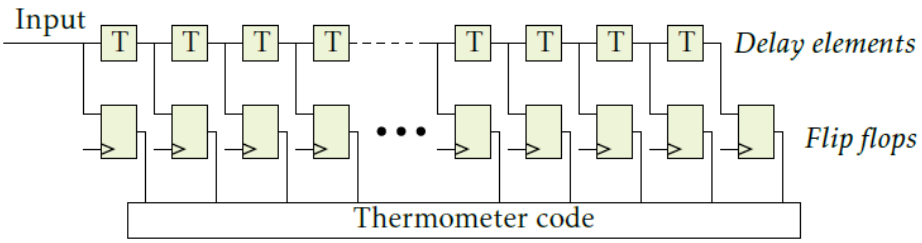


Figure 2.1: *Principles of a TDL*

Figure 1: Example of Tapped delay line

Los elementos de retardo pueden ser implementados de distintas formas, e.g. utilizando compuertas inversoras ([2]), no obstante como la implementación será hecha en una FPGA se utilizarán *Carry4s*, unidades básicas de compuertas lógicas configurables contenidas en *slices* ([3], [4]) originalmente destinadas a construir comparadores, decodificadores, y otros, cuyos routeos están especialmente hechos para tener el menor retardo de propagación. El rango de medición está dado por la cantidad de elementos de retardo utilizados, mientras que la precisión no tiene una relación de mejora con este factor sino que se cree que puede incluso empeorar al incrementar las probabilidades de error. Estas imprecisiones están dadas por cambios en las condiciones PVT (Process, Voltage, and Temperature), cambios en la distribución del clock o clock jitter, y no linealidades. Las arquitecturas utilizadas principalmente dependen de estos factores, al intentar atenuar los problemas mencionados es necesario el uso de cada vez más recursos.

Análogo a un ADC la característica entrada-salida se mide en LSB (leas significant bit) versus una cantidad continua que en este caso será el intervalo temporal a medir. Los intervalos son transformados a cantidades cuantizadas de salida que idealmente tienen una cantidad uniforme de error, sin embargo este puede no ser el caso ([5]).

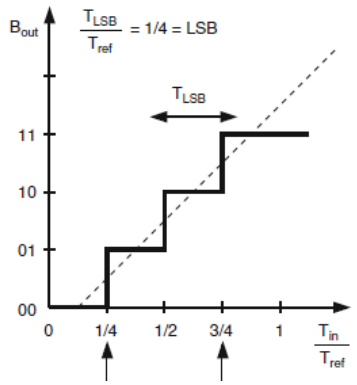


Figure 2: Característica entrada-salida.

En principio la utilización de un *coarse counter* (contador grueso) es necesario para extender el rango de mediciones sin necesidad de aumentar la cantidad de taps, mientras se cubra el largo de un clock el contador grueso nos permite seguir teniendo la misma precisión.

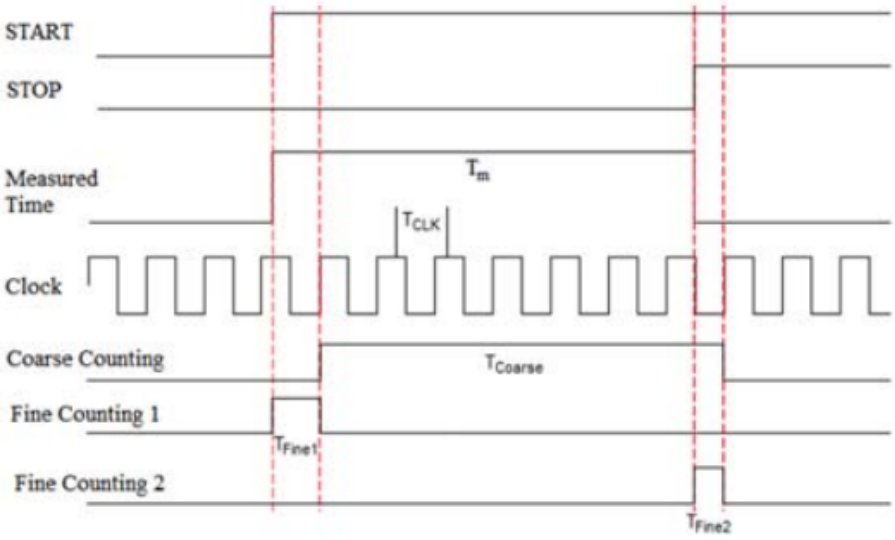


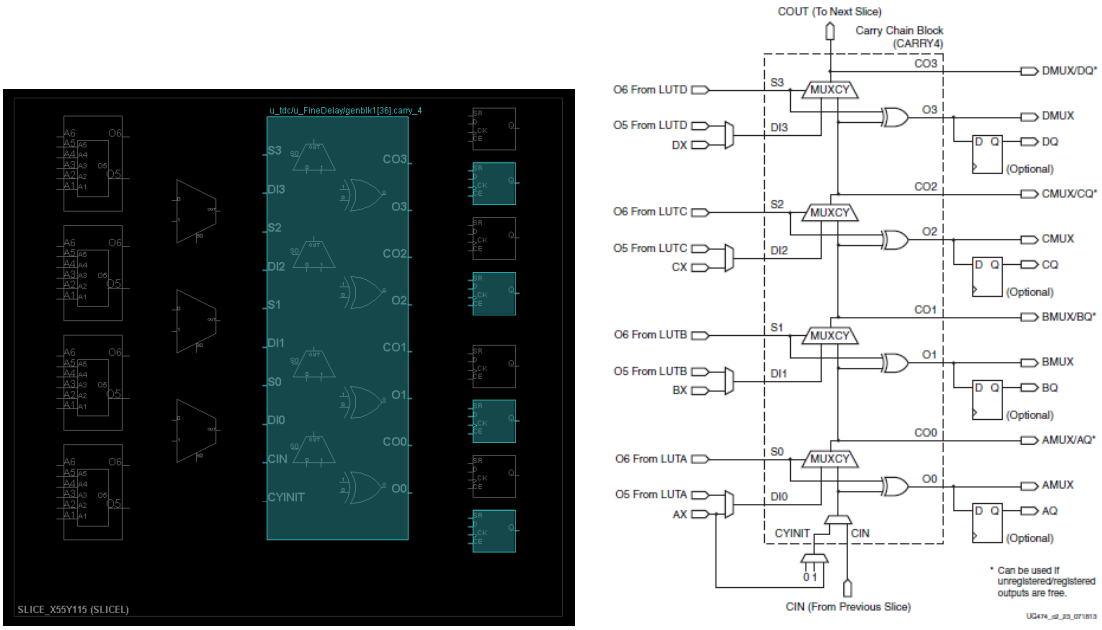
Fig. 1. Functioning principle of a generic TDC.

Figure 3: Caption de prueba

Arquitectura

A este tipo de estructuras digitales nos gusta llamarlas *ad-hoc*, ya que desde su concepción hasta su implementación podremos ver que no se puede enmarcar dentro de una estructura típica, sino que su diseño viene estrechamente ligado a su aplicación en particular. A simple vista puede verse que los elementos que se utilizarán, las etapas de procesamiento, y las técnicas implementadas son poco ortodoxas para lo que es el mundo del diseño digital. Se da a continuación una explicación descriptiva de cada bloque fundamental que conforma la arquitectura del *TDC*.

El bloque principal de este diseño, encargado de realizar la *medición fina*, es una cadena de retardos registrados. Para implementar estos retardos existen distintas pero acotadas opciones en una implementación en FPGA. En sus inicios la solución predilecta era utilizar multiplexores en cadena [6], pero para aprovechar las ventajas propias de la estructura de una FPGA se evolucionó a utilizar *Carry4s*, posiblemente utilizado por primera vez en [7]. Estos elementos forman parte de la unidad mínima de trabajo en una FPGA, y su combinación con LUTs, Flip-Flops, y Multiplexores forman lo que se conoce como un *SLICE*[1]. Los *Carry4s* en su concepción fueron destinados para implementar sumadores, y tienen la ventaja de estar optimizados para tener el mínimo retardo entre compuerta y compuerta. Además cada salida cuenta con un Flip-Flop disponible inmediatamente luego, como se muestra en la Figura 4.



(a) SLICE L de una Artix-7 (b) Carry 4

Figure 4: Elementos disponibles dentro de una SLICE.

Cada Carry4 cuenta con 4 salidas que pasan por una XOR, (O0-O3) y 4 salidas que pasan por cada MUXCY (CO0-CO3), además cuenta con un puerto *CIN* con el propósito de colocarlas en cadena y una entrada auxiliar *CYINIT* desde donde ingresará el pulso externo.
Como la implementación aquí presentada es asíncrona precisamos capturar la señal en su flanco de entrada y de salida de la cadena de retardos, por lo que luego de la colocación de la primer columna de Flip Flops (*First* Flip Flops), como se ve en 4, se utilizan otras dos columnas de Flip Flops llamados *Start* Flip Flops y *Stop* Flip Flops. Estos se encargan de registrar la primer columna solamente cuando un flanco de subida o de bajada es detectado, a traves de su *Clock Enable*. Así se genera la estructura que se ve a continuación en la Figura 5.

[1]La Artix-7 cuenta con un total de 33650 slices.

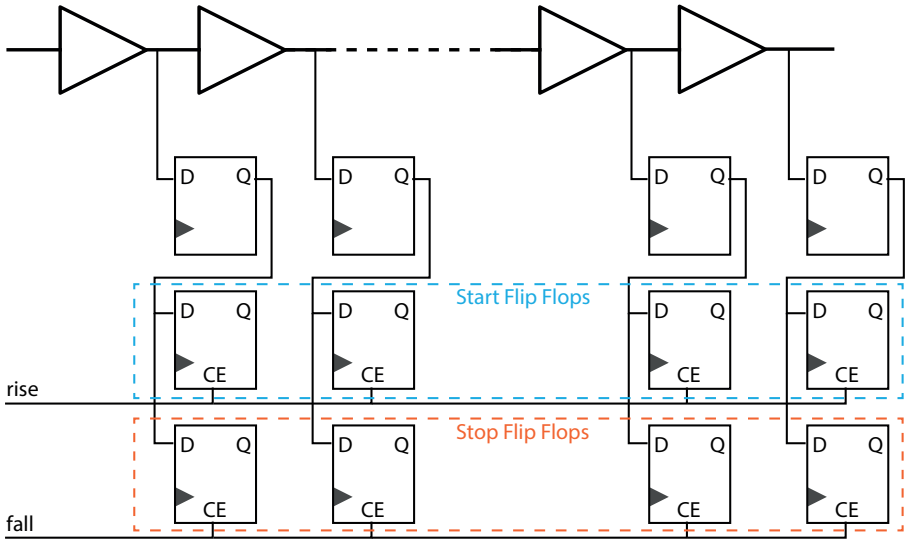


Figure 5: Esquema de la cadena del módulo de interpolación fina.

De esta estructura surgen dos aspectos importantes:

1) Es sabido que la transferencia de información analoga a digital puede producir metastabilidades si no se realiza correctamente. La configuración de doble flip flop para registrar los elementos de retardo soluciona este problema introduciendo lo que se conoce como "sincronizador".
Esto es, si el tiempo de establecimiento de la señal externa es menor al tiempo de establecimiento requerido por un sólo flip flop, este puede obtener un valor desconocido a la salida. Sin embargo introduciendo un segundo flip flop en serie, este no sufre de este problema, ya que para el próximo clock la señal es estable, y esta indeterminación se resuelve decantando hacia un estado. Además la técnica de doble registro reduce los problemas de burbujas en el resultado [8].

2) Una propiedad intrínseca del diseño tipo Nutt es que el tiempo del pulso a medir atraviese la cadena en un tiempo no menor al de un clock del sistema. Esta es una condición de diseño y se aprovecha en múltiples bloques, pero en particular aquí se puede ver que el registro de *Start* y *Stop* dependen de un flanco de subida o bajada para capturar la cadena retardos, el cuál tiene una duración de un clock.

Por otro lado, es importante prestar especial atención al routeo y colocación de cada elemento en el bitstream final, pues es preponderante en la calidad de los resultados del TDC. Si los retardos introducidos por el routeo son iguales y uniformes para cada elemento de retardo entonces podemos asegurar que la calibración es capaz de corregir en gran parte estos errores. Según [8] es importante colocar la cadena de retardos en las Slices que quedan a continuación de los Clock Buffers, y que los flip flops sean colocados también cerca de la cadena. Esto fue probado empíricamente y se obtuvieron mejores resultados cuando los extremos de la cadena de retardos quedaban simétricos respecto de los clock buffers.
Experimentalmente se observó que cuando el extremo inferior de la cadena es lejana a los buffers, se pueden observar huecos en el histograma resultante, mientras que al colocarlo simétrico, este espacio se reparte al inicio y al final de la cadena, como se ilustra en la Figura **??**.

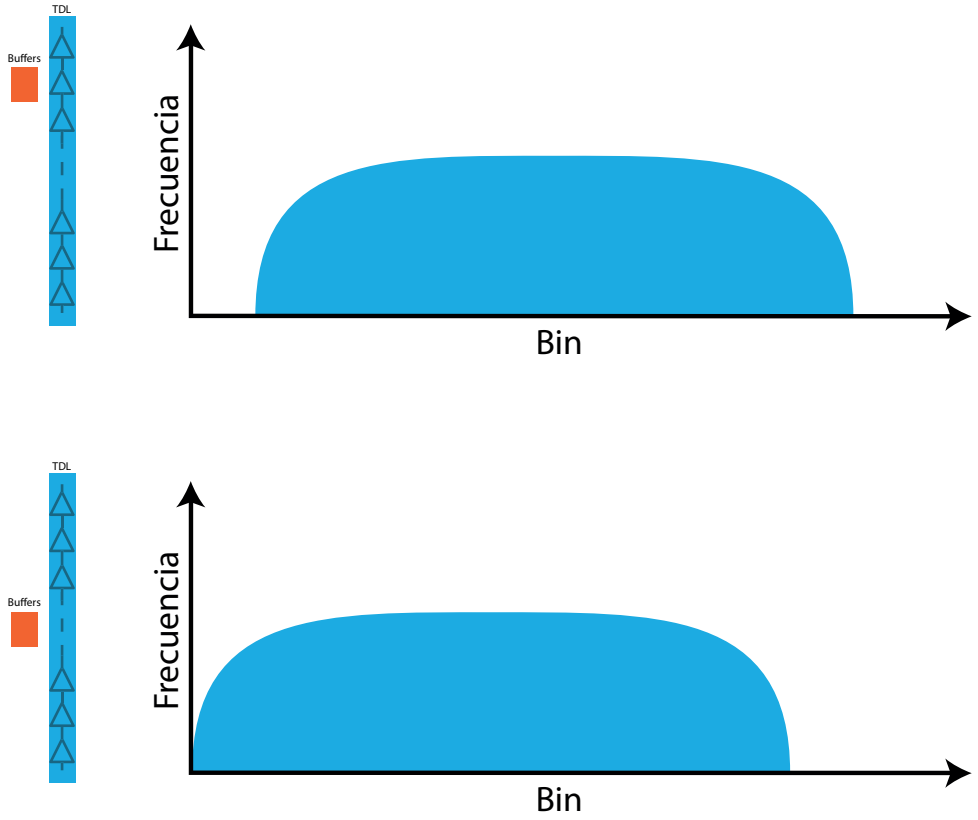


Figure 6: Ejemplo de histograma conseguidos al excitar el TDC con una frecuencia pseudo-aleatoria modificando la posición relativa de la cadena de retardos respecto a los buffers de clock.

b Este fenómeno tiene una explicación, y es que si la cadena de buffers mide temporalmente un clock, entonces la posición de los buffers de clock posiblemente cambie la sincronía de los Flip-Flops si no se realiza con cuidado. Los Flip-Flops extremos son los más susceptibles a tener un mayor desfase que el resto, debido al delay de routeo. Por lo tanto al poner los buffers en una posición equidistante logramos compensar ese delay a ambos extremos, mientras que en el ejemplo de la Figura 6 existe un retardo mayor para el *Carry4* de entrada.

2.2 Decoder

La propagación de un pulso por la línea de retardos naturalmente genera un código unario sobre los registros. Esto es, a medida que el flanco descendente progresa sobre la línea de retardos deja a su paso unos en cada registro de la cadena, mientra que cuando egresa un flanco ascendente deja ceros a su paso, tal como muestra la Figura 7.

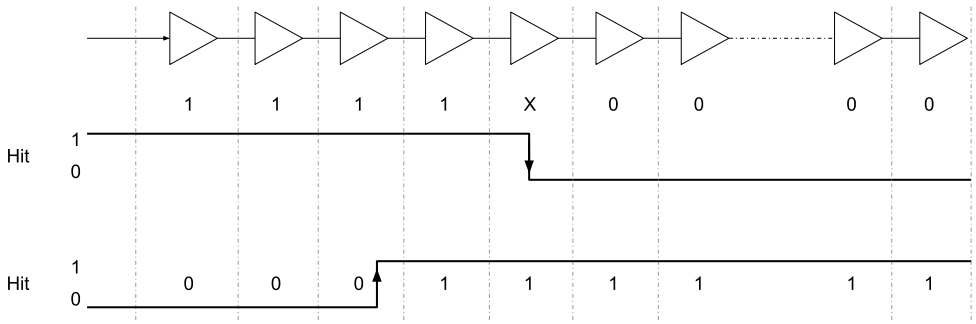


Figure 7: Registro de un pulso digital en el momento que ingresa a la línea, y en el momento que egresa de ella.

La misión principal del decoder es traducir estos valores leídos de los registros *Start* y *Stop* a datos procesables. Aquí se presenta un gran inconveniente presente en todas las soluciones propuestas por bibliografía, pues debido a las variabilidades propias de la arquitectura de una FPGA y de los órdenes de precisión que se pretenden es muy frecuente encontrar burbujas en los registros resultantes, imposibilitando la utilización de decodificadores unarios convencionales. Esto significa que en los registros *Start* o *Stop* se pueden encontrar registros con errores del tipo "11..111001000...", es decir, no es posible distinguir una transición clara de estado. Según la bibliografía ([1]) esto se debe principalmente al efecto de *clock skew* cada vez más presente en los últimas tecnologías, gracias al decrecimiento abrupto de tamaño de fabricación. Es altamente posible que la diferencia temporal del flanco de clock (el clock skew) entre dos Flip Flops que registran elementos de retardo adyacentes sea mayor al tiempo de retardo de la celda a registrar, y por lo tanto es altamente posible que se produza una burbuja en el registro. Esto se ejemplifica en la Figura 8. Si el Flip-Flop número uno captura la cadena en el momento t_1 y el Flip-Flop número dos lo hace un momento después en t_2, es posible entonces que el primero decante su estado en un valor 0, mientras que el segundo decante su estado en un valor 1. La burbuja ocurre entonces si $\Delta = t_2 - t_1 \geq \tau$ siendo τ el tiempo de una celda de retardo, y Δ el clock skew.

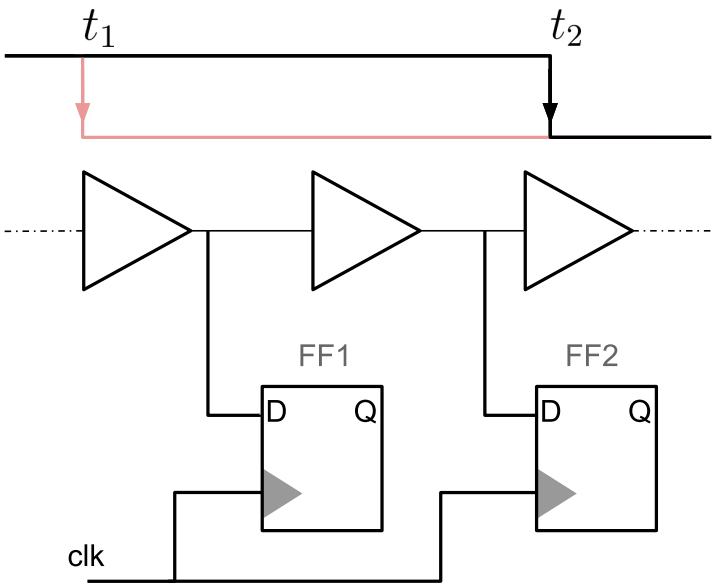


Figure 8: Retardo virtual de propagación que se puede observar cuando existe *clock skew* entre dos flip-flops aledaños.

Se han propuesto distintas soluciones, en este sección resumiremos las principales ideas:

1) **Contador de unos (ones counter):** se propuso como una solución en el diseño de ADCs, y luego en [9] se utiliza por primera vez para el diseño de un TDC. La implementación consiste básicamente en contar la cantidad de estados altos en el registro para definir la posición aproximada del flanco. Si se puede asegurar que la cantidad de burbujas es despreciable entonces la respuesta del decodificador se acerca a la real. Esto es beneficioso porque se prescinde del ordenamiento de un registro, es decir la entrada "111100" se decodificaría en la misma respuesta que la entrada "111010", por lo tanto tiene la habilidad natural de corregir errores de burbuja.

2) **Buscador de flancos:** Consiste en realizar un circuito combinacional que busque secuencialmente (look-for) transiciones de estados del tipo "0-1" o "1-0" en el registro de entrada y devuelva la posición de este dentro la cadena, e.g para un registro "0001111111" el decoder debe devolver tres, ya que es la posición donde se detecta la transición. Para eliminar el problema de burbujas se recurre a aumentar el tamaño de la secuencia buscada, es decir si se tiene un registro con burbujas del tipo "0001001111" podemos buscar la secuencia "0001" de forma que únicamente la primer transición es detectada y la burbuja se ignora ([10]). Rápidamente se hace evidente que a medida que el tamaño del filtro incrementa es capaz de corregir burbujas de mayor tamaño, a costa de descartar una mayor cantidad de muestras al principio y final de la cadena.

3) **Realineamiento de taps:** Una vez detectados los lugares del registro donde se producen burbujas, se intercambia la posición de salida de estos flip-flops con el de uno cercano ([11]), luego se utiliza un decoder del tipo "buscador de flancos". Para su implementación primero se debe hacer un "code-density test" que nos permita conocer cuáles son los bines de tamaño cero (burbujas), para luego realizar el reordenamiento de taps. Esta implementación depende fuertemente de las condiciones PVT[1] en las que se utiliza la FPGA.

2.3 Medición gruesa y Árbitro

Una de las ventajas de la implementación tipo Nutt es que nos permite mantener un buen rango de medición sin perder precisión. Esto se logra separando la implementación en dos tareas, la medición gruesa y la medición fina, para esto se utiliza un contador grueso (coarse counter) con el que se mide la cantidad de periodos de clocks que tarda la señal en atravesar la cadena. Aunque esta propuesta parece sencilla precisa de una sincronización previo al procesamiento del resultado, ya que si la señal externa entra en sincronía con el clock del sistema es posible que el contador cuente o no un periodo de reloj menos, perdiendo una precisión de un periodo completo. En [8] se propone la utilización de más de un contador grueso como árbitros para decidir el resultado final del cálculo, el desarrollo es el siguiente:

Se implementaron tres contadores gruesos, cada uno alimentado por un clock derivado del clock del sistema y ligeramente desfasado, como se muestra en la Figura 9.

[1] Proceso, voltaje y temperatura.

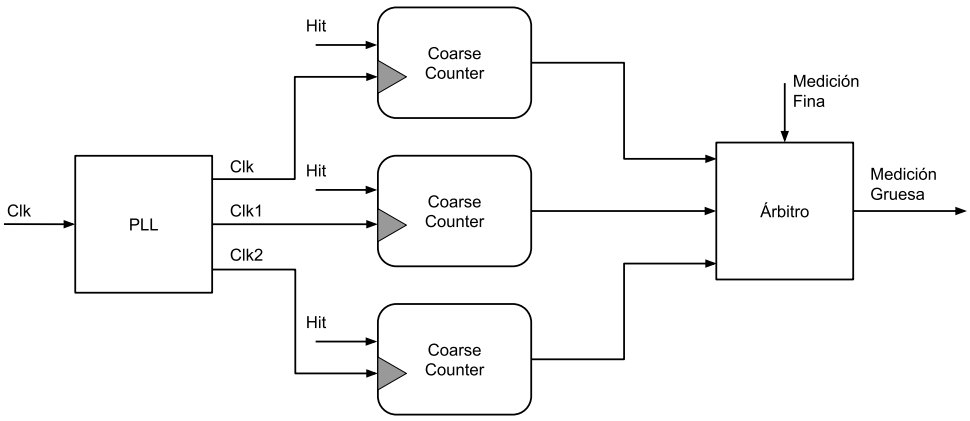


Figure 9: Arbitro implementado.

Estos tres clocks inducen cuatro casos a diferenciar, estos se plantearan para el flanco de entrada pero puede hacerse de la misma forma para el flanco de salida. La señal puede llegar antes del flanco de subida de todos los clocks, como muestra el Caso I de la Figura 10, o también llegar entre cada par de flancos como muestran los Casos II, III y IV. Para el primer caso, la línea de retardos captura la señal al inicio de la cadena pues el clock ocurre cerca de la llegada del pulso, sin embargo para el resto de casos donde el flanco de clock ya ocurrió es necesario esperar un periodo al próximo flanco de clock, en consecuencia se captura el flanco de entrada al borde de saturar la cadena de retardos, como muestra la Figura 11.

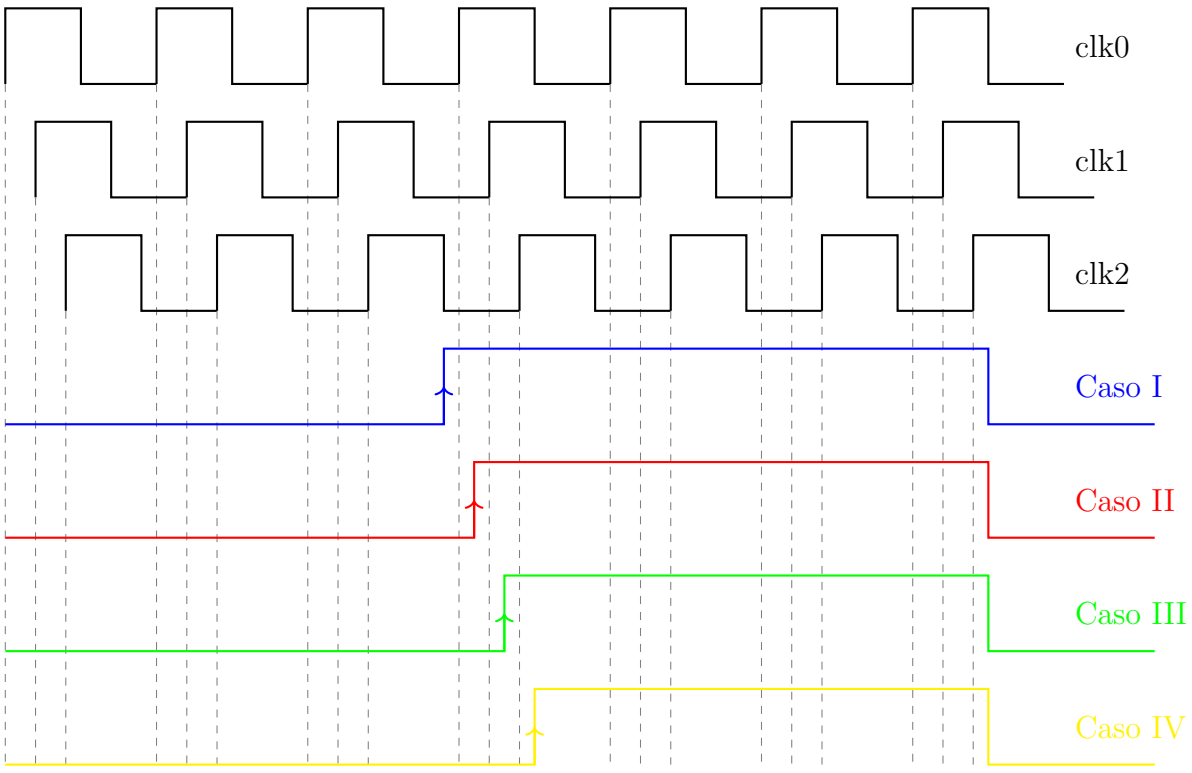


Figure 10: Todos los casos posibles de arbitraje para el flanco Start

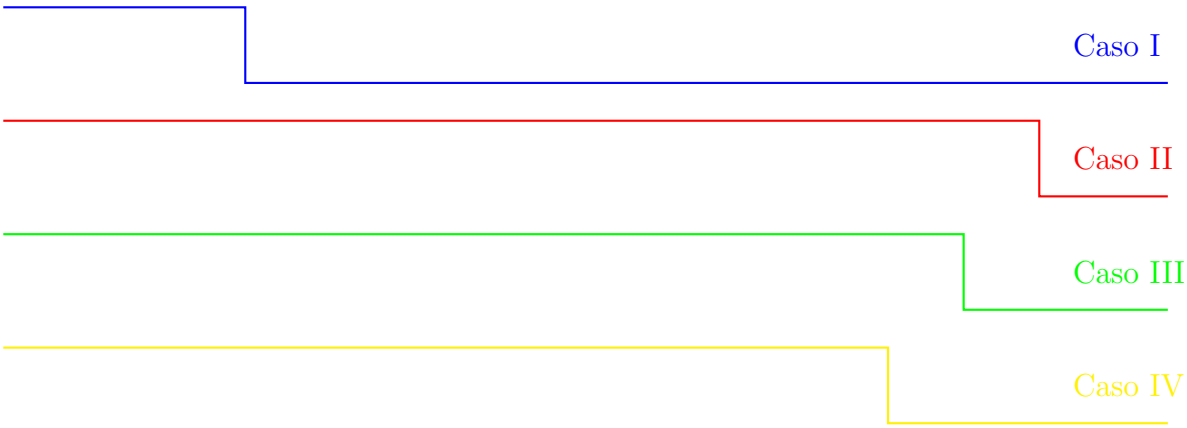


Figure 11: Registro *Start* de la cadena de retardos para cada caso de arbitraje.

Si analizamos cada caso con su coarse counter entonces obtendríamos los resultados que se presentando en la Tabla **??**

Coarse counter según:	clk0	clk1	clk2
Caso I	4	4	4
Caso II	3	4	4
Caso III	3	3	4
Caso IV	3	3	3

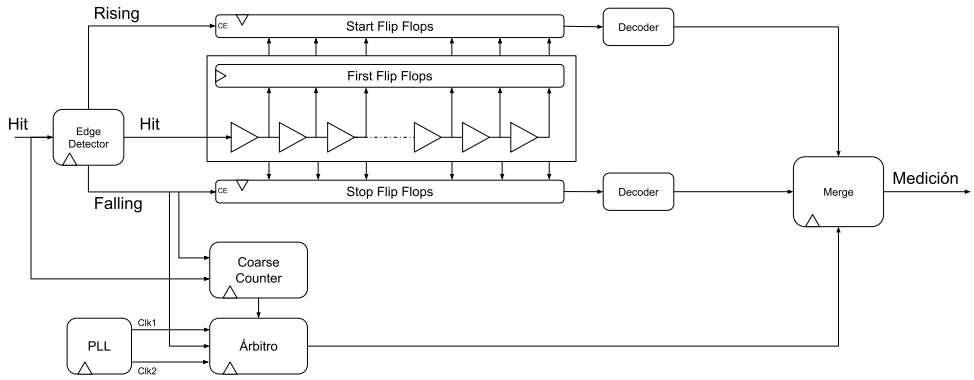


Figure 12: Arquítectura del TDC.

Referencias

[1] Rui Machado, Jorge Cabral, and Filipe Serra Alves. "Recent Developments and Challenges in FPGA-Based Time-to-Digital Converters". In: *IEEE Transactions on Instrumentation and Measurement* 68.11 (2019), pp. 4205–4221. DOI: 10.1109/TIM.2019.2938436.

[2] Juan Ignacio Morales. "Diseño de sistemas microelectrónicos basados en alta resolución temporal". PhD thesis. Bahía Blanca: Universidad del Sur, 2020.

[3] Xilinx. *7 Series FPGAs Configurable Logic Block*. AMD. 2016.

[4] AMD. *Carry4*. https://docs.amd.com/r/en-US/ug953-vivado-7series-libraries/CARRY4. Accedido: 25-03-2024. 2024.

[5] Stephan Henzler. *Time-to-Digital Converters*. Springer, 2010.

[6] J. Kalisz, R. Szplet, J. Pasierbinski, and A. Poniecki. "Field-programmable-gate-array-based time-to-digital converter with 200-ps resolution". In: *IEEE Transactions on Instrumentation and Measurement* 46.1 (Feb. 1997). Conference Name: IEEE Transactions on Instrumentation and Measurement, pp. 51–55. ISSN: 1557-9662. DOI: 10.1109/19.552156. URL: https://ieeexplore.ieee.org/document/552156/citations#citations (visited on 07/16/2024).

[7] Claudio Favi and Edoardo Charbon. "A 17ps time-to-digital converter implemented in 65nm FPGA technology". In: *Proceedings of the ACM/SIGDA international symposium on Field programmable gate arrays*. FPGA '09. New York, NY, USA: Association for Computing Machinery, Feb. 2009, pp. 113–120. ISBN: 978-1-60558-410-2. DOI: 10.1145/1508128.1508145. URL: https://doi.org/10.1145/1508128.1508145 (visited on 07/15/2024).

[8] Rui Machado, Luis A. Rocha, and Jorge Cabral. "A novel synchronizer for a 17.9ps Nutt Time-to-Digital Converter implemented on FPGA". In: *2018 AEIT International Annual Conference*. Oct. 2018, pp. 1–6. DOI: 10.23919/AEIT.2018.8577365. URL: https://ieeexplore.ieee.org/document/8577365 (visited on 07/16/2024).

[9] Yonggang Wang, Jie Kuang, Chong Liu, and Qiang Cao. "A 3.9-ps RMS Precision Time-to-Digital Converter Using Ones-Counter Encoding Scheme in a Kintex-7 FPGA". In: *IEEE Transactions on Nuclear Science* 64.10 (Oct. 2017). Conference Name: IEEE Transactions on Nuclear Science, pp. 2713–2718. ISSN: 1558-1578. DOI: 10.1109/TNS.2017.2746626. URL: https://ieeexplore.ieee.org/document/8022888 (visited on 07/25/2024).

[10] Jinyuan Wu. "Several Key Issues on Implementing Delay Line Based TDCs Using FPGAs". In: *IEEE Transactions on Nuclear Science* 57.3 (June 2010), pp. 1543–1548. ISSN: 1558-1578. DOI: 10.1109/tns.2010.2045901. URL: http://dx.doi.org/10.1109/TNS.2010.2045901.

[11] Chong Liu and Yonggang Wang. "A 128-Channel, 710 M Samples/Second, and Less Than 10 ps RMS Resolution Time-to-Digital Converter Implemented in a Kintex-7 FPGA". In: *IEEE Transactions on Nuclear Science* 62.3 (June 2015), pp. 773–783. ISSN: 1558-1578. DOI: 10.1109/tns.2015.2421319. URL: http://dx.doi.org/10.1109/TNS.2015.2421319.