INSTITUTO TECNOLÓGICO DE COSTA RICA
ESCUELA DE INGENIERÍA ELECTRÓNICA
EL3313 TALLER DE DISEÑO DIGITAL
PROF. M.SC. KALEB ALFARO BADILLA
GRUPO 20
I SEMESTRE 2024

Laboratorio 2: Lógica Secuencial

CUESTIONARIO PREVIO

Realizado por:

Anthony Artavia Salazar
Yailyn Priscilla Campos Zamora
Samuel Montenegro Gómez

04/03/2024

Índice

1. PREGUNTA 1

Investigue sobre el funcionamiento de máquinas de estados finitos. Explique la diferencia entre una máquina de Moore y una de Mealy, y muestre la diferencia por medio de diagramas de estados y señales.

Las máquinas de estados (*finite state machines*) corresponden a circuitos secuenciales en los que k registros pueden estar en uno de un número finito (2^k) de estados únicos. Un FSM tiene M entradas, N salidas y k bits de estado. También recibe un reloj y, opcionalmente, una señal de *reset*. Un FSM consta de dos bloques de lógica combinacional, lógica de siguiente estado y lógica de salida, y un registro que almacena el estado. En cada flanco del reloj, el FSM avanza al siguiente estado, que se calculó en función del estado actual y las entradas [1]. Hay dos clases generales de máquinas de estados finitos, caracterizadas por sus especificaciones funcionales. En las máquinas de Moore, las salidas dependen únicamente del estado actual de la máquina. En las máquinas Mealy, las salidas dependen tanto del estado actual como de las entradas actuales. Estas máquinas de estado se pueden visualizar a partir de la siguiente figura [1]:

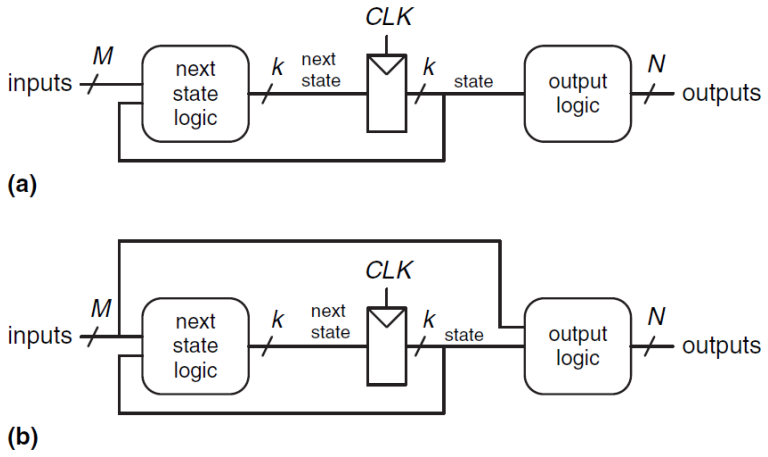

Figura 1.1: Máquinas de estados finitos: (a) Máquina de Moore, (b) Máquina de Mealy [1].

Para denotar mejor la diferencia entre las máquinas de Moore y Mealy se mostrará la solución de un ejercicio planteado en [1]. El enunciado plantea lo siguiente: "Alyssa P. Hacker tiene un caracol robótico como mascota con cerebro FSM. El caracol se arrastra de izquierda a derecha a lo largo de una cinta de papel que contiene una secuencia de unos y ceros. En cada ciclo de reloj, el caracol avanza hasta el siguiente bit. El caracol sonríe cuando los dos últimos bits sobre los que se ha arrastrado son 01. Diseñe el FSM para calcular cuándo debería sonreír el caracol. La entrada A es la punta debajo de las antenas del caracol. La salida Y es VERDADERA cuando el caracol sonríe. Compare los diseños de máquinas de estados de Moore y Mealy. Dibuja un diagrama de tiempo para cada máquina que muestre la entrada, los estados y la salida mientras el caracol de Alyssa se arrastra a lo largo de la secuencia 0100110111". Como solución, en [1] se encuentran los diagramas de a continuación:

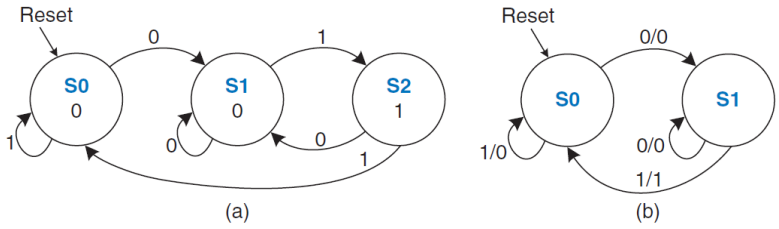

Figura 1.2: Diagramas de transición de estados: (a) Máquina de Moore, (b) Máquina de Mealy [1].

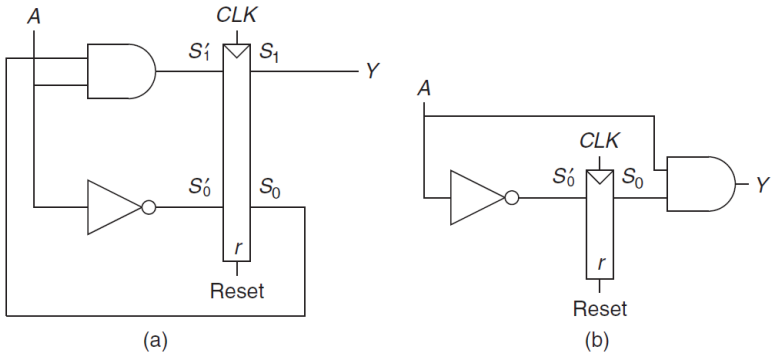

Figura 1.3: Esquemáticos: (a) Máquina de Moore, (b) Máquina de Mealy [1].

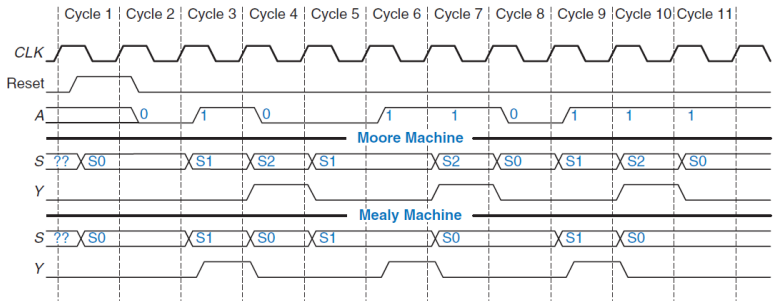

Figura 1.4: Diagramas de tiempo para máquinas de Moore y Mealy [1].

2. Pregunta 2

Explique los conceptos de *setup time* y *hold time*. ¿Qué importancia tienen en el diseño de sistemas digitales?

El tiempo de establecimiento o *setup time* (t_s), es el intervalo de tiempo mínimo que los niveles lógicos deben mantener constantes en las entradas antes de que llegue el flanco de disparo del impulso de reloj, de modo que dichos niveles se sincronicen correctamente. Por ejemplo, en el caso de un flip-flop D (figura 2.1), el nivel lógico debe estar en la entrada D durante un tiempo mayor o igual a t_s antes que el flanco de disparo de clk, para que la entrada de datos sea correcta [2].

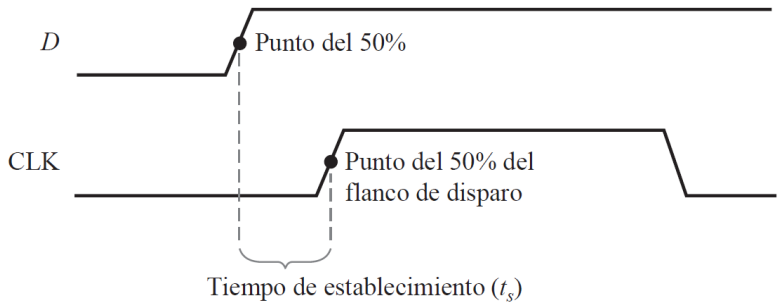

Figura 2.1: Tiempo de establecimiento t_s de un flip-flop D para la entrada D [2].

El tiempo de mantenimiento o *hold time* (t_h) es el intervalo de tiempo mínimo que los niveles lógicos deben mantenerse constantes en las entradas después de que haya pasado el flanco de disparo del impulso de reloj, de modo que dichos niveles se sincronicen correctamente. Tomando nuevamente como ejemplo el caso del flip-flop D (figura 2.2), el nivel lógico se debe mantener en la entrada D durante un tiempo mayor o igual a t_h después que el flanco de disparo de clk para tener una entrada de datos correcta [2].

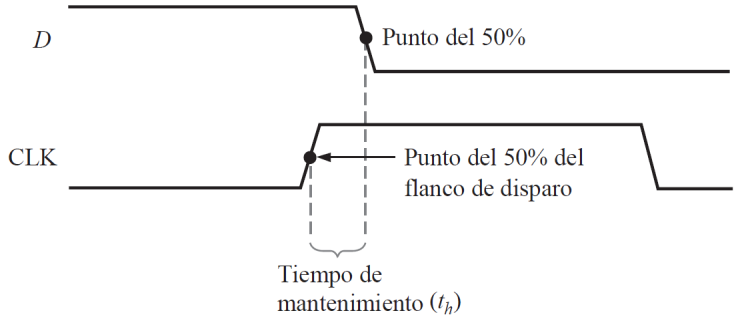

Figura 2.2: Tiempo de mantenimiento t_h de un flip-flop D para la entrada D [2].

La importancia de estos dos tiempos, t_s y t_h, en el diseño de sistemas digitales radica en que, estos determinan los tiempos de apertura de un elemento secuencial (el flip-flop en este caso), donde un tiempo de apertura indica el tiempo que una señal debe mantenerse estable para producir una salida bien definida. Esta limitación viene de la disciplina dinámica en el diseño de sistemas digitales que permite utilizar únicamente señales que cambien de valor fuera del tiempo de apertura [1].

La disciplina dinámica no se puede cumplir siempre, en el caso de que una señal cambie justo en el flanco del reloj y el elemento secuencial capture un valor intermedio entre 0 y 1, se presenta el fenómeno de metaestabilidad. Esta situación puede tomar un tiempo indefinido para resolverse en un buen valor lógico y se dice que la entrada es asincrónica [1].

3. PREGUNTA 3

Explique los conceptos de tiempos de propagación y tiempos de contaminación en circuitos combinacionales. Investigue sobre la ruta crítica y cómo esta afecta en el diseño de sistemas digitales complejos; por

ejemplo, un procesador con pipeline. Investigue su relación con la frecuencia máxima de operación de un circuito.

Los tiempos de propagación y de contaminación caracterizan la lógica combinacional. El tiempo de propagación t_p es el tiempo máximo desde que alguna entrada cambia hasta que las salidas alcanzan su valor final. El tiempo de contaminación t_c es el tiempo mínimo desde que alguna entrada cambia hasta que alguna salida empieza a cambiar su valor [1]. Estos tiempo se ilustran en la figura 3.1 para una compuerta lógica.

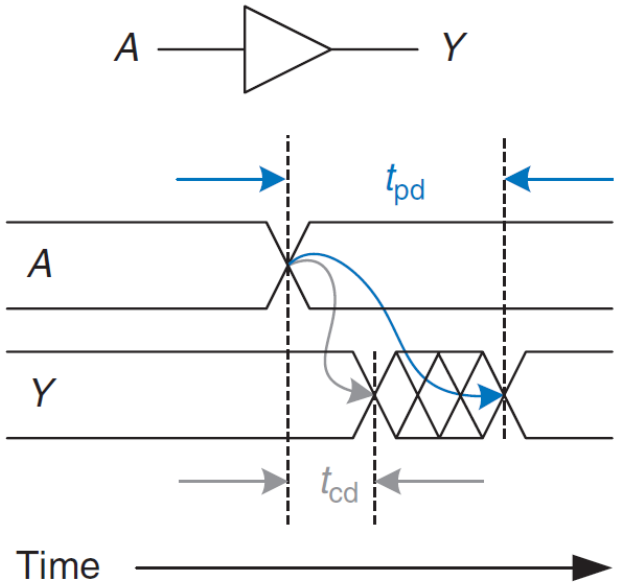

Figura 3.1: Tiempos de propagación t_p y contaminación t_c para un buffer [1].

La ruta crítica o *critical path*, es el camino más largo (y el más lento) que tiene que atravesar una señal desde una entrada hasta una salida. Este camino o ruta se denomina crítica porque limita la velocidad a la que opera el circuito. La ruta corta o *short path*, es el camino más corto (y el más rápido) que tiene que atravesar una señal desde una entrada hasta una salida [1].

El tiempo de propagación de un circuito es la suma de los tiempos de propagación de cada elemento en la ruta crítica. El tiempo de contaminación de un circuito es la suma de los tiempos de contaminación de cada elemento en la ruta corta [1]. En la figura 3.2 se ilustran estos tiempos para un circuito de ejemplo.

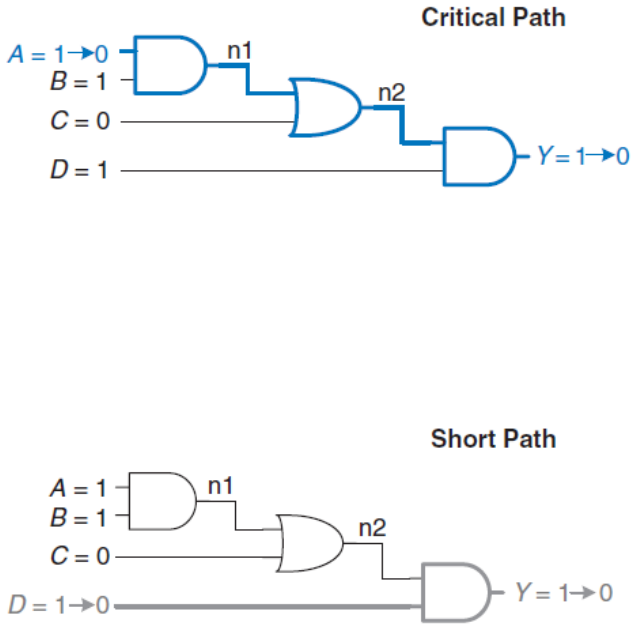

Figura 3.2: Tiempos de propagación t_p y contaminación t_c para un circuito [1].

Pipelinig o canalización es una técnica de implementación en la cual múltiples instrucciones se ejecutan superpuestas para aumentar la velocidad de ejecución [3].

Esta técnica se utiliza en procesadores para aumentar el rendimiento de sistemas electrónicos digitales, usando la sincronización para que la ruta crítica se reduzca [4].

La frecuencia máxima de operación de un circuito es el recíproco de la ruta crítica. A mayor ruta crítica, menor frecuencia máxima de operación [4].

4. Pregunta 4

Investigue sobre las mejores prácticas para la asignación de relojes y división de frecuencia en FPGAs. En este apartado haga énfasis en el uso de las entradas habilitadoras de reloj (*clock enables*) presentes en las celdas de la FPGA, para lograr tener tiempos de ejecución diferentes a lo largo del sistema mientras se utiliza un solo reloj..

Acorde a [5], existen ciertas reglas al llevar a cabo diseño síncrono, entre las cuales, para el presente documento, destaca la siguiente: "El circuito debe disponer de una señal de reloj única y común para todos los flip-flops del circuito". Esta es la regla fundamental del diseño síncrono, y la que condiciona el modo de funcionamiento característico de los circuitos síncronos. Cumplirla implica que la ocurrencia de un flanco activo de reloj es percibida simultáneamente por todos los flip-flops del circuito y, puesto que sus entradas asíncronas no se usan, los instantes de captura de datos y conmutación de las salidas son exactamente los mismos para todos los flip-flops y se suceden periódica e indefinidamente, al ritmo marcado por la frecuencia de reloj del circuito, mientras el sistema opera. De este modo de funcionamiento se deriva la simplicidad del funcionamiento de los

circuitos síncronos y, en última instancia, su inmunidad a los glitches generados por los circuitos combinacionales.
Lo anterior hace que el diseño de la red de interconexión entre los flip-flops resulte delicado. Al igual que en el caso de la señal global de reset, las FPGAs y otros dispositivos lógicos programables disponen de entradas especiales –entradas dedicadas para relojes globales- con capacidad para atacar a todos los flip-flops del chip con una dispersión de retardos mínima.
En el caso de necesitar hacer uso de señales con frecuencias diferentes a las del reloj del sistema es posible hacer uso de divisores de frecuencia, sin embargo, como se indica en [5], estos pueden provocar sesgos (*skew*) lo que puede provocar violaciones de temporizado y por ende un impacto en la sincronización de señales. Para corregir este problema, se recomienda hacer uso de las entradas habilitadoras de reloj (*clock enables*), ya que con ellos se puede obtener mayor control sobre el temporizado y así reducir el riesgo de introducir problemas de muestreo.

5. PREGUNTA 5

Investigue sobre el fenómeno de rebotes y ruido en pulsadores e interruptores. Defina qué técnicas digitales (circuitos) se utilizan para cancelar este fenómeno. Además, investigue sobre los problemas de metastabilidad cuando se tienen entradas asíncronas en circuitos digitales. Finalmente, presente circuitos que permitan la sincronización de entradas como pulsadores e interruptores.

Hay muchos tipos de interruptores conmutadores, como los de palanca, los basculantes, los pulsadores, los microinterruptores y los interruptores de límite, entre otros. La gran mayoría tienen una cosa en común: rebotan [6].

Si el circuito digital es lo suficientemente rápido como para detectar y responder a múltiples rebotes, puede haber graves consecuencias. La tarea de un ingeniero es evitar o mitigar los efectos de este rebote. Los rebotes son las falsas pulsaciones (ruido) que se producen al hacer falsos contactos en el interruptor. El proceso de eliminarlos se llama "Debounce". Si no se mitiga, el rebote de los interruptores puede hacer que los microprocesadores y otros circuitos electrónicos consideren que la activación de un solo interruptor comprende varios eventos [6].

Por ejemplo, si se considera un interruptor conmutador de montaje en panel de un solo polo y un solo tiro (SPST) normalmente abierto (NO) como un M2011SS1W01 de NKK.

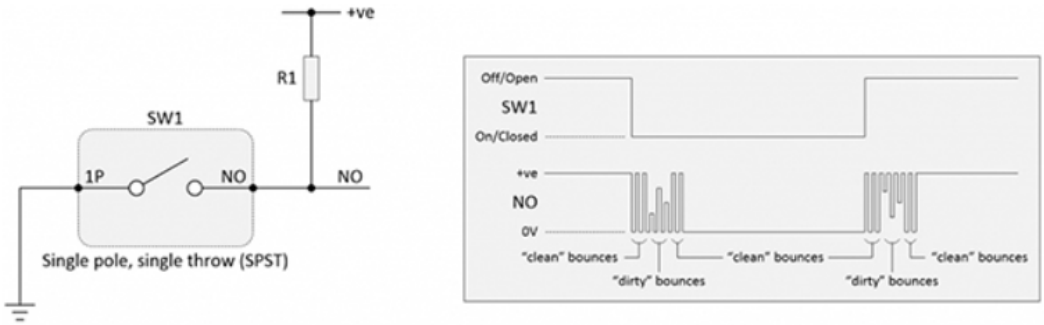

Figura 5.1: En el caso de un interruptor conmutador SPST-NO, pueden producirse rebotes tanto al activar como al desactivar el interruptor.[6]

El rebote del interruptor puede producirse tanto cuando el interruptor está activado (cerrado) como cuando está desactivado (abierto). A veces, los rebotes pueden transitar entre los carriles de alimentación, aquí considerados estados lógicos 0 y 1. En este caso, se trata de rebotes "limpios". En comparación, si la señal solo alcanza una tensión intermedia, se denominan rebotes "sucios". Los problemas surgen cuando un interruptor se conecta a un equipo electrónico lo suficientemente rápido como para detectar y responder a múltiples rebotes. Lo que se necesita es una forma de desbaratar la señal procedente del interruptor antes de que actúe el equipo electrónico [6].

Las principales y más usados maneras de eliminar los rebotes son tres. Eliminación por software, Eliminación mediante filtro R-C y Eliminación mediante biestables tipo R-S. En las décadas de 1960 y 1970, el rebote de los interruptores se implementó utilizando una variedad de técnicas de hardware, desde simples circuitos de retardo de resistencia-condensador (RC) utilizados con los interruptores SPST hasta funciones más sofisticadas de enclavamiento de ajuste/reinicio (SR). Más recientemente, y debido a que muchos sistemas cuentan con una unidad de microprocesador (MPU) o una unidad de microcontrolador (MCU), se ha convertido en algo habitual el uso de técnicas de software para desbaratar la señal procedente de cualquier interruptor [6].

En el caso de un interruptor SPDT, una solución de rebote de hardware común es emplear un latch SR. Este enfoque ha sido considerado como de las soluciones de rebote de hardware más simples. El latch SR puede formarse utilizando dos puertas NAND de dos entradas, una detrás de la otra. Otra, una solución sencilla de rebote de interruptor basado en hardware emplea una resistencia-capacitor (RC). Una de las implementaciones más versátiles implica dos resistencias y un diodo.

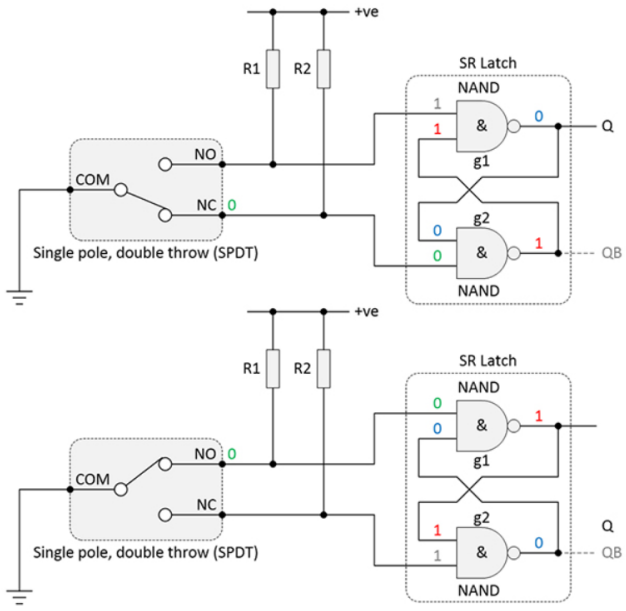

Figura 5.2: Un latch SR basado en NAND para eliminar el rebote de un interruptor SPDT.[6]

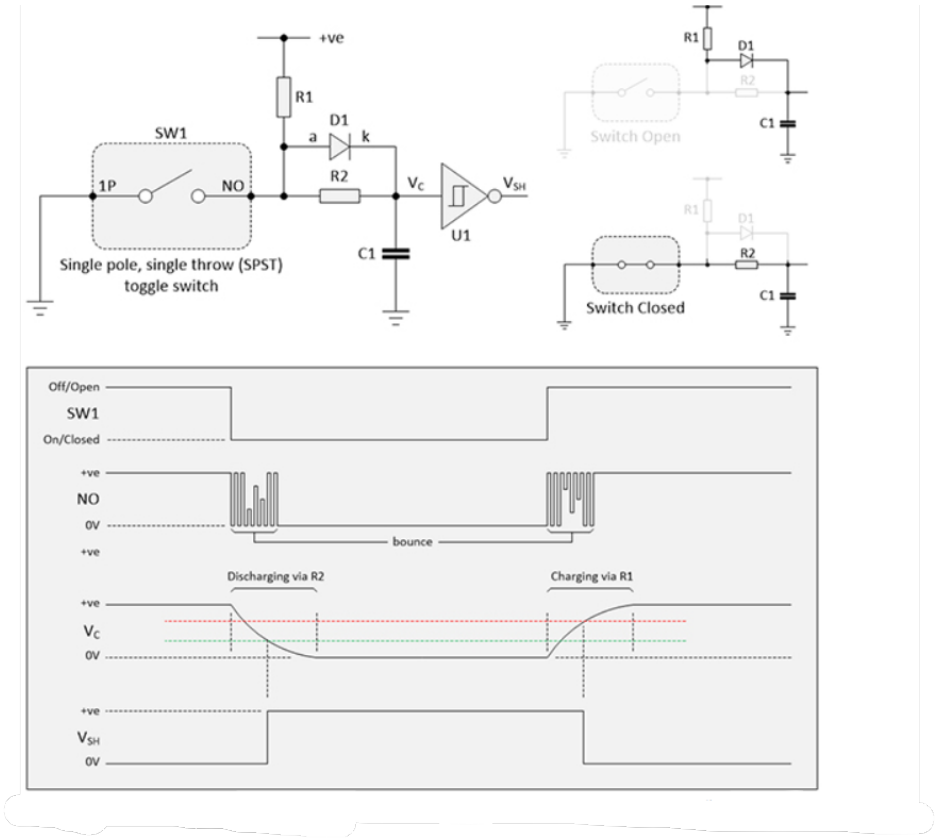

Figura 5.3: Red RC para un interruptor SPST, el diodo obliga al condensador a cargarse a través de R1 y a descargarse a través de R2.[6]

Como se ha señalado anteriormente, no siempre es posible garantizar que la entrada a un circuito secuencial sea estable durante el tiempo de apertura, especialmente cuando la entrada procede del mundo exterior. Considere un botón conectado a la entrada de un flip-flop.

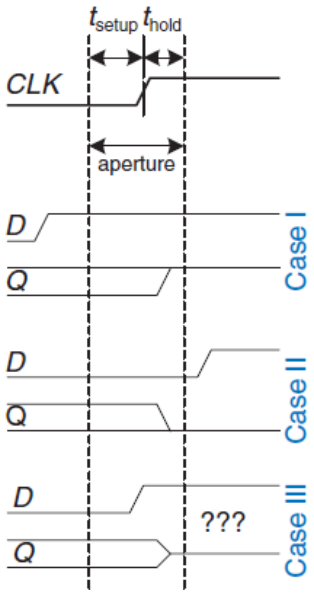

Figura 5.4: Cambio de entrada antes, después o durante la apertura[1]

Cuando el botón no está pulsado, D=0. Cuando el botón está pulsado, D=1. Si un mono pulsa el botón en algún momento aleatorio relativo al flanco ascendente de CLK. Si se quiere conocer la salida Q después del flanco ascendente de CLK. En el caso I, cuando el botón se pulsa mucho antes de CLK, Q= 1. En el Caso II, cuando el botón no se pulsa hasta mucho después de CLK, Q= 0. Pero en el Caso III, cuando el botón se pulsa en algún momento entre t_setup antes de CLK y t_hold después de CLK, la entrada viola la disciplina dinámica y la salida es indefinida. Cuando un flip-flop muestrea una entrada que está cambiando durante su apertura, la salida Q puede tomar momentáneamente un voltaje entre 0 y VDD que está en la zona prohibida. Esto se denomina estado metaestable. Finalmente, el flip-flop resolverá la salida a un estado estable de 0 ó 1. Sin embargo, el tiempo de resolución necesario para alcanzar el estado estable es ilimitado.

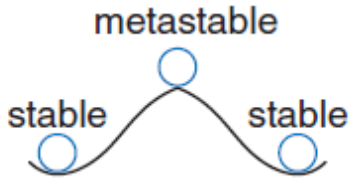

Figura 5.5: Cambio de entrada antes, después o durante la apertura[1]

6. PREGUNTA 6

Investigue sobre el concepto de *IP-Core*. Revise la documentación relativa al uso de las herramientas de *IP-Core* en Vivado, en particular sobre el *Clocking-wizard*, y los IPS de verificación física: ILA (Integrated Logic Analizer) y VIO (Virtual Input/Output). Sobre estos IPs resuma para qué se utilizan, cómo configurarlos, y cómo utilizarlos en su proyecto.

Un módulo de propiedad intelectual (IP Core) es un bloque lógico o datos utilizado en la fabricación de FPGAs o un circuito integrado de aplicación específica para un producto.Comúnmente utilizado en semiconductores,

un PI Core es una unidad reutilizable de lógica o diseño de circuito integrado (CI).Como elementos esenciales de la reutilización de diseños, los módulos IP son parte de la creciente automatización de diseño electrónico(EDA).

Un IP Core suele ser un módulo independiente que forma parte de un dispositivo o sistema más grande, como un procesador u otro CI complejo. Incorpora circuitos electrónicos licenciados el diseñador original a otras empresas. La mayoría de los IP Core se desarrollan utilizando lenguajes de descripción de hardware (HDL), como VHSIC HDL, Verilog o SystemVerilog.

LogiCORE™ IP Clocking Wizard genera código fuente HDL para configurar un circuito de reloj según los requisitos del usuario. El asistente puede seleccionar automáticamente una primitiva de reloj adecuada y configurar los parámetros de búfer, realimentación y temporización de una red de reloj, o ayudar al usuario a configurar los atributos de una primitiva seleccionada manualmente. Si lo desea, el usuario también puede anular cualquier parámetro calculado por el asistente. Además de generar HDL fuente para el circuito de sincronización, el asistente también invoca las herramientas de análisis de temporización de Xilinx para generar un informe de parámetros de temporización.[7]

Dentro de sus carácteristicas cuenta con:

- Selección primitiva de mixed-mode clock manager (MMCM) y phase-locked loop (PLL).
- La función Safe Clock Startup permite un reloj estable y válido en la salida. La activación de la función de secuenciación proporciona relojes de salida secuenciados.
- Acepta hasta dos relojes de entrada y hasta siete relojes de salida por red de reloj.
- Configura automáticamente una primitiva de sincronización en función de las características de sincronización seleccionadas.

Clocking Wizard ayuda a crear el circuito de sincronización para la frecuencia de reloj de salida, la fase y el ciclo de trabajo requeridos utilizando un mixed-mode clock manager (MMCM)(E2/E3/E4) o una primitiva Phase Locked Loop (PLL)(E2/E3/E4). También ayuda a verificar la frecuencia de reloj generada de salida en simulación, proporcionando un diseño de ejemplo sintetizable que puede probarse en el hardware. La siguiente figura muestra un diagrama de bloques del Clocking Wizard.[7]

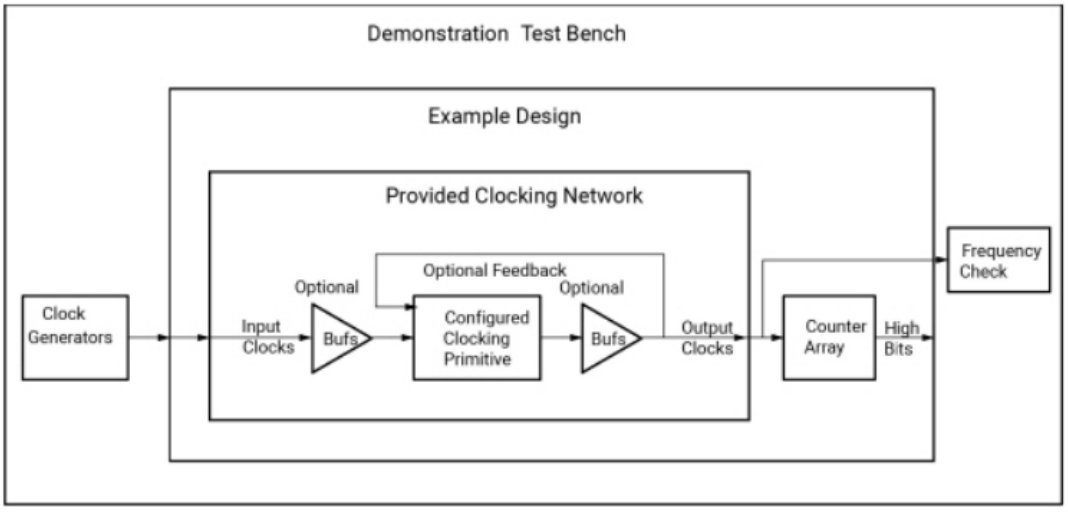

Figura 6.1: Clocking Wizard Block Diagram [7]

Un reloj de entrada es el comportamiento por defecto, pero se pueden elegir dos relojes de entrada seleccionando una fuente de reloj secundaria. Sólo se requieren los parámetros de temporización de los relojes de entrada en sus unidades especificadas; el Wizard utiliza estos parámetros según sea necesario para configurar los relojes de salida. Se puede configurar el número de relojes de salida. El número máximo permitido depende del dispositivo o primitiva seleccionado y de la interacción de las principales características de reloj que especifique. Para MMCM(E2/E3) se puede configurar un máximo de siete, para PLLE2 un máximo de seis y para PLLE3 un máximo de dos relojes de salida. Si la primitiva seleccionada es Auto, entonces se pueden configurar los siete relojes de salida como máximo. Además de configurar la primitiva de sincronización dentro del dispositivo, el Wizard también ayuda a construir la red de sincronización.[7]

Se ofrecen opciones de búfer para los relojes de entrada y salida. El usuario puede controlar la realimentación de la primitiva o dejar que el Wizard la conecte automáticamente. Si se selecciona la realimentación automática, la ruta de realimentación se ajusta a la temporización de clk-out1 [7].

ILA(Integrated Logic Analyzer) y VIO(Virtual Input/Output) son IPs personalizables gratuitas de Xilinx. ILA ayuda a sondear fácilmente las señales internas de su FPGA y llevarlas a un entorno de simulación para monitorizarlas y verificar su comportamiento. la IP VIO permite controlar virtualmente las señales internas de su FPGA para estimular o controlar su diseño, como controlar la señal RESET [8].
ILA Core incluye muchas funciones avanzadas de los analizadores lógicos modernos, como ecuaciones de disparo booleanas y disparos de transición de flancos. Dado que ILA Core es síncrono al diseño que se está supervisando, todas las restricciones de reloj de diseño que se aplican a su diseño también se aplican a los componentes dentro del ILA Core. Está diseñado para ser utilizado en cualquier aplicación que requiera verificación o debugging mediante el Vivado logic analyzer [9].

Para VIO, el número y la anchura de los puertos de entrada y salida son personalizables en tamaño para interactuar con el diseño FPGA. Dado que el VIO Core es síncrono al diseño que está siendo monitorizado y/o controlado, todas las restricciones de reloj de diseño que se aplican a su diseño también se aplican a los componentes dentro del VIO Core.

Este proporciona LED virtuales y otros indicadores de estado a través de puertos de entrada síncronos. También incluye detectores de actividad opcionales en los puertos de entrada para detectar transiciones ascendentes y descendentes entre muestras, junto con una proporción de botones virtuales y otros controles a través de puertos de salida síncronos, incluye una inicialización de salida personalizada que permite especificar el valor de las salidas del núcleo VIO inmediatamente después de la configuración y puesta en marcha del dispositivo y finalemente permite el restablecimiento en tiempo de ejecución del VIO Core a sus valores iniciales [10].

REFERENCIAS

[1] S. Harris and D. Harris, *Digital design and computer architecture: RISC-V Edition.* Oxford, England: Morgan Kaufmann, 2021.

[2] Floyd, Fundamentos de Sistemas Digitales 9 Edicion. Pearson Educacion, 2007.

[3] D. A. Patterson and J. L. Hennessy, *Computer Organization and Design MIPS Edition: The Hardware/Software Interface*, 5th ed. Oxford, England: Morgan Kaufmann, 2013.

[4] I. E. Díaz, *"Técnicas Para Mejorar La Performance De Un Microprocesador. Techniques to improve the performance of a microprocessor* Cloudfront.net. [Online]. Disponible: Técnicas para mejorar la Performance.

[5] M. A Freire Rubio, *Diseño Síncrono de Circuitos Digitales*, Universidad Politécnica de Madrid, 2008.

[6] C. Maxfield, *“Cómo implementar el rebote de hardware para interruptores y relés cuando el rebote de software no es apropiado,* Digikey. [Online]. Disponible: Rebote de hardware para interruptores.

[7] AMD Xilinx ,*Clocking Wizard LogiCORE IP Product Guide (PG065)*, AMD Xilinx. [Online]. Disponible: Clocking Wizard LogiCORE IP Product Guide

[8] A. Zaklouta,*Using Integrated Logic Analyzer (ILA) and Virtual Input/Output (VIO).* [Online]. Disponible: ILA and VIO

[9] AMD Xilinx, *Integrated Logic Analyzer (ILA)*, AMD Xilinx. [Online]. Disponible: ILA Xilinx

[10] AMD Xilinx, *Virtual Input/Output (VIO)*, AMD Xilinx. [Online]. Disponible: VIO Xilinx