Laboratorio 2:
Lógica secuencial

Profesor

Kaled Alfaro Bonilla

Estudiantes

Fiorela Chavarría Castrillo

William Sánchez Ramírez

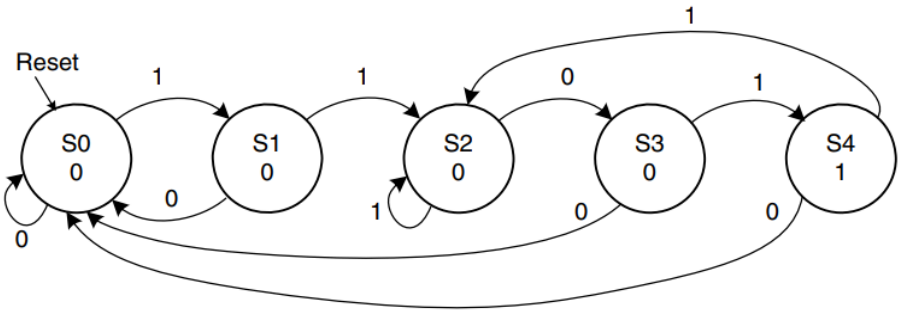
Rolando Vega Marino

I Semestre 2024

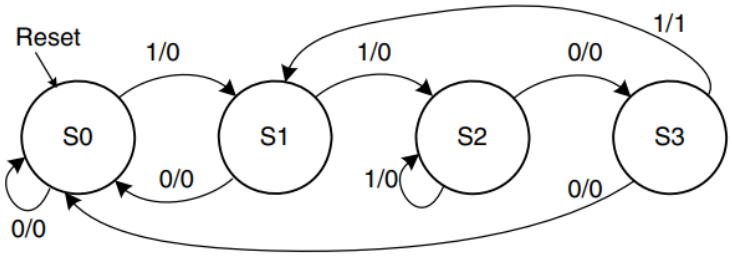
En el ámbito del diseño de sistemas digitales, la comprensión de conceptos y técnicas clave es fundamental para garantizar el correcto funcionamiento y la fiabilidad de los circuitos. Entre estos conceptos se encuentran las máquinas de estado finito, fundamentales en el diseño de sistemas secuenciales, junto con aspectos como el setup time, hold time y los tiempos de propagación y contaminación en circuitos combinacionales, que juegan un papel crucial en la estabilidad y el rendimiento del sistema. La asignación adecuada de relojes y la división de frecuencia en FPGAs también son aspectos importantes a considerar para optimizar el diseño y la operación del sistema. Además, los desafíos asociados con los rebotes y el ruido en pulsadores e interruptores, así como los problemas de metaestabilidad en circuitos digitales, deben abordarse con técnicas específicas para garantizar la fiabilidad del sistema. Por último, la integración de IP-Cores en el diseño de sistemas digitales ofrece una solución eficiente para la implementación de funciones específicas, proporcionando flexibilidad y reduciendo los costos de desarrollo.

Máquinas de estado finitos

Las máquinas de estado finitas como lo dice el nombre tienen un funcionamiento basado en un número de estados específico y un número de transiciones entre los estados. Las transiciones de un estado a otro se generan cuando tenemos algún cambio en las entradas o en las señales de reloj, esto genera cambios en las salidas según lo que se busca con el diseño de la máquina de estado.
La máquina de Moore se basa en un funcionamiento donde las salidas dependen solamente del estado actual, mientras que la máquina de Mealy las salidas dependen tanto de las entradas como del estado actual.
Ejemplo de un detector de secuencia de números binarios para la combinación 1101:



Máquina de Moore



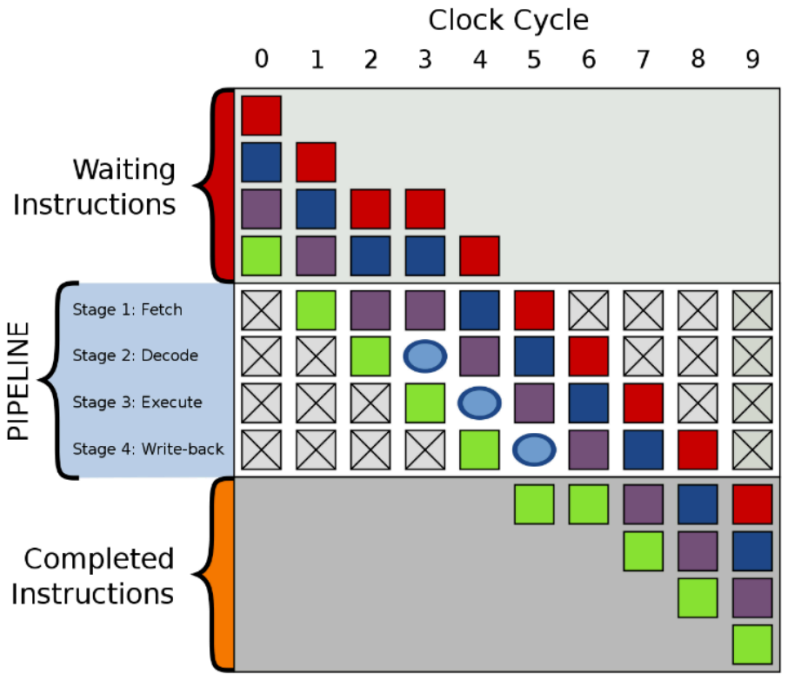
Máquina de Mealy

Setup time y hold time

En el diseño de sistemas digitales existen dos parámetros que son relevantes en cuanto al tiempo de una señal de entrada, estos son: el *setup time* y el *hold time* que básicamente son tiempos donde la señal debe permanecer estable al ocurrir un evento de clock para que así la operación del circuito sea confiable. El *setup time* es el tiempo mínimo requerido para que los datos de entrada permanezcan estables antes de que ocurra cualquiera de los flancos de disparo (triggering-edge) del reloj. Por otra parte, el *hold time* es ese tiempo en el cual la señal de entrada debe mantenerse estable después de que los flancos de disparo sean activados. Estos dos parámetros son importantes porque de ser el caso contrario, donde la señal tenga un cambio mientras el período esté transcurriendo podría darse un problema de metaestabilidad, dejando como resultado una posible pérdida de los datos almacenados o que estos sean inestables.

Tiempo de propagación y contaminación

El tiempo de propagación se refiere al tiempo que tarda una señal en propagarse a través de un circuito, desde la entrada hasta la salida. Por otro lado, el tiempo de contaminación se refiere al tiempo que tarda una señal en contaminar una salida después de que la entrada ha cambiado. La ruta crítica en un circuito es la ruta que tiene el mayor tiempo de propagación entre la entrada y la salida. Este tiempo de propagación en la ruta crítica es crucial ya que limita la velocidad máxima de operación del circuito. Si la frecuencia de operación del circuito es demasiado alta, la señal no tendrá suficiente tiempo para propagarse por la ruta crítica antes de la llegada del siguiente pulso de reloj, lo que puede resultar en errores en la salida. En un procesador con pipeline, la ruta crítica es la ruta más larga entre las etapas de la tubería. El tiempo de propagación en la ruta crítica determina la velocidad máxima de operación del procesador. Si la frecuencia de operación del procesador es demasiado alta, las etapas de la tubería no tendrán suficiente tiempo para procesar la información antes de que llegue el siguiente pulso de reloj, lo que puede provocar errores en la salida.



Por otra parte, la implementación de registros de retardo y buffers de interconexión puede ser beneficiosa para reducir el tiempo de propagación en la ruta crítica y aumentar la velocidad de operación del circuito.

Relojes y frecuencia en FPGA

A la hora de trabajar con FPGA es importante saber asignar bien los tiempo de reloj, para realizar esta asignación existen diferentes formas, tenemos el caso del incremento, el caso donde se niega la entrada, otro caso con if/else, otro caso utilizando el comando case, otro caso de sumar y restar, existen otros casos más específicos. Para la división de frecuencia tenemos la práctica es con un circuito biestable con el que podemos dividir por 2 la frecuencia, se puede usar también un *toggle switch*, se puede implementar con compuertas lógicas. La importancia de utilizar un *clock enable* es que esto nos ayuda a no utilizar tantos recursos de reloj y nos ayuda a mejorar las tiempos dentro del programa, con esto se obtienen mejores datos para los análisis de los proyectos. En la FPGA se puede utilizar unas celdas de reloj que vienen integradas y para los casos que se tenga que utilizar valores diferentes en la frecuencia del reloj se pueden añadir *clock enables* para poder controlar la frecuencia a la que se va a trabajar.

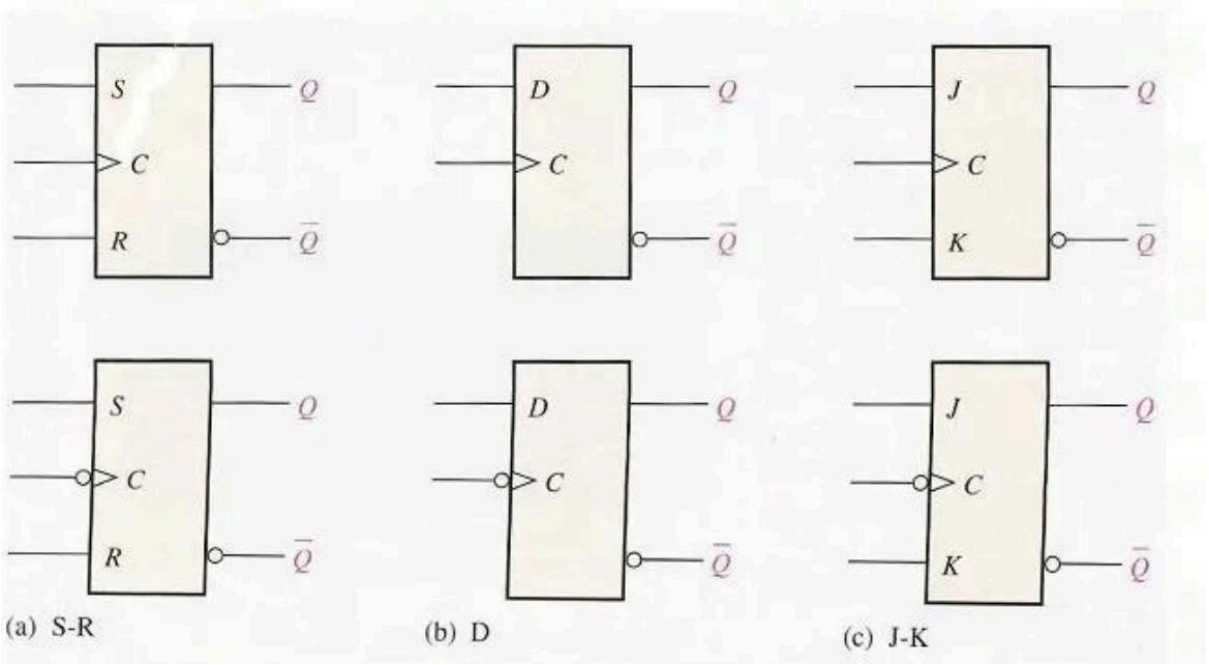
Rebote y ruido

El rebote en botones e interruptores se manifiesta cuando, al presionar o liberar el botón, los contactos metálicos dentro del interruptor entran en contacto y se separan repetidamente en un breve periodo de tiempo, en lugar de establecerse de manera instantánea y firme. Este fenómeno genera señales eléctricas no deseadas, denominadas "ruido", las cuales pueden ocasionar que el sistema interprete múltiples pulsaciones o cambios de estado incorrectos. En esencia, el rebote representa fluctuaciones temporales en la **conexión eléctrica** del

interruptor, mientras que el ruido corresponde a las **señales eléctricas** no deseadas que acompañan estos cambios.

Las técnicas de hardware para mitigar el efecto de rebote en interruptores abarcan diversas estrategias de implementación de circuitos a nivel físico. Entre estas técnicas se encuentran el uso de latches S-R (Flip-Flop), donde un circuito latch se coloca en la salida del interruptor para retener el nivel de voltaje de la entrada y latching al cambio de estado introducido. Aunque eficaz, esta técnica puede aumentar la complejidad y el tamaño del circuito. Otra técnica común es el empleo de circuitos R-C (filtro RC), que combina un resistor y un capacitor para actuar como un filtro que suaviza la salida del interruptor, eliminando las fluctuaciones causadas por el rebote.

Los problemas de metaestabilidad en circuitos digitales, especialmente cuando se enfrentan a entradas asíncronas, pueden generar una serie de consecuencias no deseadas en el sistema. Esto incluye la posibilidad de que los flip-flops o latches entren en un estado metaestable, donde su salida oscila entre estados alto y bajo de manera impredecible, lo que resulta en una salida indefinida que puede causar comportamientos erráticos en el sistema. Además, la metaestabilidad puede llevar a la corrupción de datos, especialmente en sistemas donde la estabilidad de los datos es crítica, lo que puede resultar en errores en la operación del sistema. El tiempo de recuperación extendido necesario para que los flip-flops o latches se estabilicen en un estado lógico definido puede causar retrasos significativos en la operación del sistema y una disminución en su rendimiento. Además, si la metaestabilidad no se maneja adecuadamente, existe un riesgo de que el sistema experimente fallas intermitentes o permanentes, lo que puede dificultar el diagnóstico y la solución de problemas, resultando en una operación no confiable del sistema.



Ip-Core

Un IP-Core es un bloque de propiedad intelectual que se integra en un diseño para proporcionar una función específica, ahorrando tiempo y costos de desarrollo al utilizar diseños previos. En Vivado, se incluyen varios IPs predefinidos, incluyendo el Clocking-wizard, ILA (Integrated Logic Analyzer) y VIO (Virtual Input/Output). *El Clocking-wizard:* es un IP que se utiliza para producir señales de reloj precisas y estables para los componentes del sistema. Se puede ajustar para generar diferentes tipos de señales de reloj, como señales de reloj diferencial, y modificar la frecuencia y la fase de la señal de salida. En Vivado, para configurar el Clocking-wizard es necesario seleccionar la frecuencia de reloj de entrada, la frecuencia de reloj de salida y la configuración de la señal de reloj. *El ILA:* es un IP que se utiliza para analizar y depurar el comportamiento del sistema en tiempo real. Se puede utilizar para capturar y almacenar datos de la señal en el interior del sistema para su posterior análisis. Para configurar el ILA en Vivado, se debe seleccionar el ancho de banda de la señal y el tamaño del búfer de datos. *El VIO:* es un IP que se utiliza para simular entradas y salidas virtuales en el diseño del sistema. Se puede utilizar para probar diferentes escenarios y verificar el comportamiento del sistema en diferentes condiciones. Para configurar el VIO en Vivado, se debe seleccionar el número de pines de entrada y salida y la configuración de la señal. Por último, para utilizar estos IPs en un proyecto en particular, es necesario agregarlos a la biblioteca de IPs de Vivado, y después insertarlos en el diseño del sistema. Posteriormente, se deben configurar de acuerdo con las necesidades del proyecto y así, proporcionar la funcionalidad requerida.

Bibliografía

Harris, D. R., & Harris, S. (2007b). Digital Design and Computer Architecture.

IC Design Tips. "Setup and Hold Time Explained." IC Design Tips. Disponible: https://www.icdesigntips.com/2020/10/setup-and-hold-time-explained.html

Murky Robot. "Debounce - Murky Robot Guides," Murky Robot, [Online]. Disponible: https://www.murkyrobot.com/guias/arduino/debounce.

S. Hageman, "Switch Bounce: How to Deal with It," All About Circuits, [Online]. Disponible: https://www.allaboutcircuits.com/technical-articles/switch-bounce-how-to-deal-with-it/.

P. Brossette, "How to Implement Hardware Debounce for Switches and Relays," Digi-Key Electronics, [Online]. Available: https://www.digikey.com/en/articles/how-to-implement-hardware-debounce-for-switches-and-relays.

"Switch Debounce in Digital Circuits," GeeksforGeeks, [Online]. Disponible: https://www.geeksforgeeks.org/switch-debounce-in-digital-circuits/.

López, Leonardo. (2017). Modelado de epidemias utilizando sistemas auto-organizados. 10.13140/RG.2.2.29087.87203. Obtenido de: https://www.researchgate.net/figure/Figura-3-1-Diagrama-de-estados-de-a-una-maquina-de-Moore-b-una-maquina-de-Mealy_fig13_320623527

Martín M, J.C.(2018) implementación de Aplicaciones en FPGA[Trabajo Fin de Grado]. Escuela Politécnica Superior de Jaén, Universidad de Jaén. Access on : https://tauja.ujaen.es/bitstream/10953.1/14345/1/TFG%20Juan%20Carlos%20Martin%20Migorance.pdf

Kilts, S. (2007). Advanced FPGA design: architecture, implementation, and optimization. John Wiley Sons.

Ronal J Tocci, Neal S Widmer, Gregory L Mos.2007. Sistemas digitales Principios y aplicaciones.10ed.Pearson. México, 425-436