Cuestionario Previo 2
Tecnológico de Costa Rica

Bryan Esquivel Flores
2020035806
zayus@estudiantec.cr
Alajuela, Costa Rica

Freddy Mora Bolaños
2021040443
freddy.mora@estudiantec.cr
Alajuela, Costa Rica

Andrés Vargas Arce
2018151379
anbo80@estudiantec.cr
Alajuela, Costa Rica

September 7, 2024

1 Preguntas

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

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

3. Explique los conceptos de tiempos de propagación y tiempos de contaminación en circuitos combinacionales. Investigue sobre la ruta crítica y cómo esta afecta el diseño de sistemas digitales complejos; por ejemplo, un procesador con *pipeline*. Investigue su relación con la frecuencia máxima de operación de un circuito.

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.

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.

6. Investigue sobre la especificación de la interfaz SPI. Preste atención a los aspectos necesarios para poder diseñar un controlador maestro de SPI, además de los diferentes modos de SPI.

7. Investigue sobre la comunicación serie UART. Preste atención a las diferentes características de configuración necesarias para la comunicación serie mediante UART (por ejemplo, *baud rate*, paridad, etc.). Además, investigue cómo puede utilizar puertos seriales en su computadora, considerando el sistema operativo que utilice.

8. Investigue el funcionamiento básico del controlador ST7789V de la pantalla LCD RGB de la Tangnano 9k. La hoja de datos será entregada por el profesor del curso.

2 Respuestas

2.1 Pregunta 1

Las máquinas de Moore y Mealy son tipos de máquinas de estado finito que se diferencian en cómo generan sus salidas, esto según lo visto en [1].

Máquinas de Moore: Las salidas dependen únicamente del estado actual. Esto hace que las salidas sean estables y cambien solo cuando el sistema cambia de estado, lo que simplifica el diseño pero puede hacer que la respuesta a las entradas sea más lenta.

Máquinas de Mealy: Las salidas dependen tanto del estado actual como de las entradas. Esto permite una respuesta más rápida a los cambios en las entradas, pero hace que el diseño sea más complejo.

Ejemplo, una secuencia de 0's y 1's que sea verdadera cuando los últimos dígitos que ha leído sean 01.

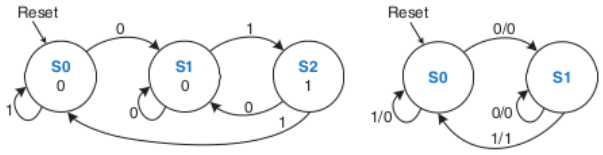

Figura 1: Máquina Moore y máquina Mealy, obtenido de [1]

2.2 Pregunta 2

De [1] podemos concluir que el *setup time* es el tiempo en el que la entrada debe estar estable antes del cambio de reloj, y el *hold time* es el tiempo establecido para que el sistema identifique la entrada como válida.

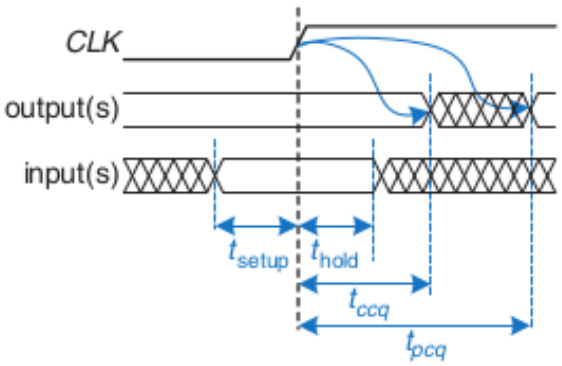

Figura 2: Setup y Hold time. Obtenido de [1]

2.3 Pregunta 3

Con base en [1] la lógica combinacional se caracteriza por sus tiempos o retardos de propagación y contaminación. El retardo de propagación (t_{pd}) es el tiempo máximo que transcurre desde que cambia una entrada hasta que la salida o salidas alcanzan su valor final, mientras que el retardo de contaminación (t_{cd}) es el tiempo mínimo desde que cambia una entrada hasta que cualquier salida comienza a cambiar su valor.

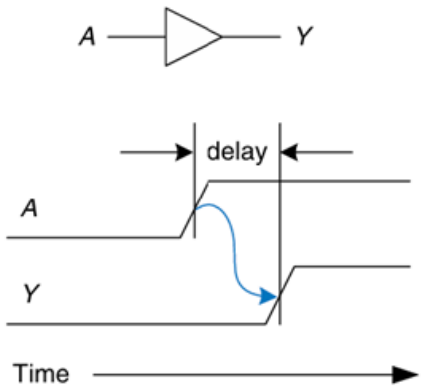

Figura 3: Retardo del circuito.[1]

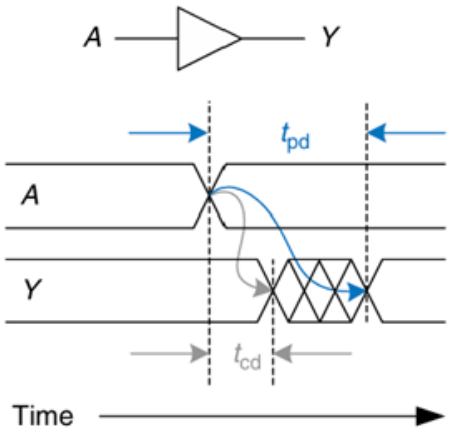

Figura 4: Retardo de propagación y contaminación.[1]

Además, indica que la ruta critica es el camino más largo que toma una señal desde una entrada hasta una salida a través de varios elementos lógicos, determina el tiempo de propagación máximo de un circuito al sumar los retardos de propagación de cada elemento a lo largo de esa ruta y es fundamental porque impone el límite superior de la frecuencia de reloj en un sistema digital, como un procesador. La frecuencia máxima de operación de un sistema digital está limitada por el tiempo de propagación máximo en la ruta crítica, por lo tanto, la frecuencia de reloj se calcula como el inverso del tiempo de propagación máximo:

$$f_{\text{máx}} = \frac{1}{t_{pd(\text{máx})}} \tag{1}$$

Esto significa que, para operar a frecuencias más altas, se debe reducir la longitud de la ruta crítica o mejorar el rendimiento de los componentes del circuito para disminuir su tiempo de propagación. En casos como los procesadores con pipeline, cada etapa del pipeline debe completarse en un ciclo de reloj. Si una etapa tiene una ruta crítica más larga, el ciclo de reloj debe alargarse para permitir que la señal se propague completamente a través de esa etapa. Por lo tanto, una ruta crítica más corta permite que el procesador funcione a una frecuencia de reloj más alta, mejorando el rendimiento general.

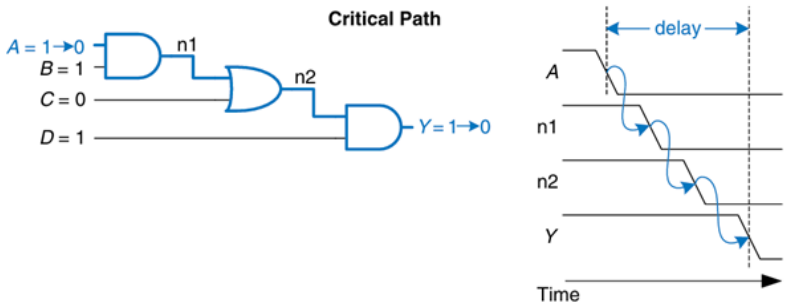

Figura 5: Señal de la ruta crítica.[1]

2.4 Pregunta 4

Uso de un Solo Reloj Principal

Una práctica recomendada es utilizar un solo reloj global y derivar otras señales de reloj de este. Esto simplifica la sincronización en el diseño y evita problemas de gestión de múltiples dominios de reloj, como el cruce de dominios, que puede introducir errores difíciles de detectar, como la metastabilidad.

División de Frecuencia con Clock Enables

En lugar de utilizar múltiples relojes con diferentes frecuencias, es preferible usar un único reloj rápido y controlar las operaciones de las diferentes partes del sistema utilizando las entradas habilitadoras de reloj (clock enables). Las entradas habilitadoras permiten que una parte del circuito se active solo en ciertos ciclos del reloj principal. Esto se logra mediante la creación de señales de habilitación que se activan a intervalos específicos, simulando así una división de frecuencia.

Ventajas del Uso de Clock Enables

- **Reducción del Consumo de Energía:** Al activar partes del circuito solo cuando es necesario, se puede reducir significativamente el consumo de energía, lo que es crucial en aplicaciones sensibles al consumo energético.
- **Menor Complejidad de Diseño:** Usar un solo reloj y clock enables reduce la complejidad relacionada con el manejo de múltiples dominios de reloj, como la necesidad de cruzar dominios de reloj con técnicas de sincronización adicionales.
- **Sincronización Mejorada:** Mantener un solo dominio de reloj mejora la sincronización en todo el sistema, ya que todas las operaciones están referenciadas al mismo reloj, eliminando la posibilidad de desincronización entre diferentes partes del sistema.

Planificación del Reloj y Colocación de Recursos

Es esencial planificar cuidadosamente la distribución del reloj dentro del FPGA. La mayoría de los FPGAs modernos tienen redes de distribución de

reloj dedicadas que deben ser utilizadas para minimizar la latencia y el skew (desfase) del reloj en todo el dispositivo. Además, colocar los recursos que comparten un clock enable cerca unos de otros dentro del FPGA puede mejorar la eficiencia del diseño y reducir las rutas de señal críticas.

Uso de PLLs y DCMs para Ajustar la Frecuencia del Reloj

Aunque el uso de clock enables es preferido para la división de frecuencia, en algunos casos es necesario generar relojes con diferentes frecuencias o fases específicas. Para estos casos, las FPGAs incluyen elementos como Phase-Locked Loops (PLLs) y Digital Clock Managers (DCMs) que permiten ajustar la frecuencia y la fase del reloj, derivando otros relojes sincronizados con el reloj principal.

2.5 Pregunta 5

En [2] podemos entender el fenómeno de los rebotes en botones o interruptores al considerar su construcción y el contacto interno. Un rebote ocurre cuando pequeñas oscilaciones en el contacto duran unos pocos milisegundos antes de que el dispositivo envíe una señal estable. Este efecto puede ser problemático en circuitos de respuesta rápida, donde esos breves milisegundos de inestabilidad pueden causar problemas significativos.

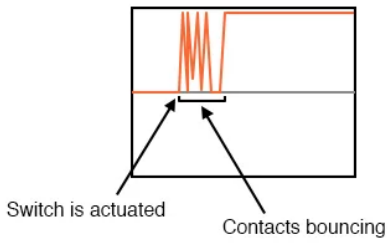

Figura 6: Representación de un rebote causado por un switch. Obtenido de [2].

Una solución que se le puede dar a los circuitos es la recomendada por [2], este nos dice que se puede utilizar un filtro pasa-bajas para filtrar el rebote:

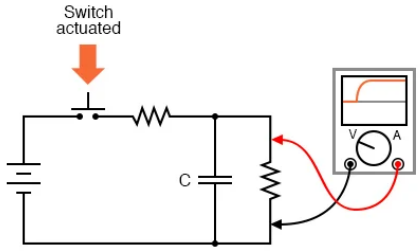

Figura 7: Filtro paso-bajo para eliminar el rebote de un switch o botón. Obtenido de [2].

Otro de los problemas que se pueden encontrar al utilizar switches es la metaestabilidad, en [1] se define un estado meta-estable como el momento en el que, por ejemplo, un flip-flop experimenta una entrada que cambia en su tiempo de apertura o su *setup time*, en ese momento la salida puede tomar un valor entre 0 y VDD, lo cual es la zona prohibida. Al final el flip-flop se estabilizará luego de un *tiempo de resolución*.

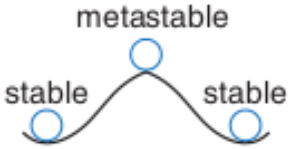

Figura 8: Representación de un estado meta-estable. Obtenido de [1].

Para evitar los meta-estados se utilizan los sincronizadores mencionados en [1], estos reciben una entrada asincrona, además de un reloj y generan una salida en un tiempo establecido, dicha salida tiene una gran probabilidad de tener un nivel lógico aceptable. A continuación se presentará una manera de construir un sincronizador a partir de dos flip-flops.

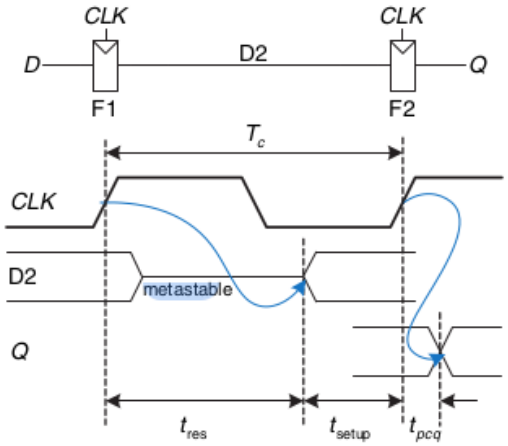

Figura 9: Sincronizador a partir de dos flip-flops. Obtendio de [1].

2.6 Pregunta 6

Según la información obtenida de [3] el protocolo SPI (Serial Peripheral Interface) es un bus serie síncrono que permite la transmisión simultánea de datos entre un maestro y uno o más esclavos. Está diseñado para la comunicación a corta distancia entre un procesador y varios periféricos o sensores. Se basa en una estructura Maestro-Esclavo, donde el maestro inicia la transmisión de datos, genera la señal de reloj y sincroniza la transferencia, mientras que el esclavo solo puede responder, estos no pueden iniciar una comunicación ni comunicarse entre sí, su única función es recibir y enviar datos al maestro. Para llevar a cabo esta comunicación se emplean 4 líneas de conexión:

1. **SC (Serial Clock):** Señal de reloj generada por el maestro que sincroniza las transferencias.
2. **MOSI (Master Out Slave In):** Línea utilizada para enviar datos del maestro al esclavo.
3. **MISO (Master In Slave Out):** Línea utilizada para enviar datos del esclavo al maestro.
4. **CS (Chip Select):** Línea de selección del esclavo, controlada por el maestro para elegir con qué esclavo comunicarse.

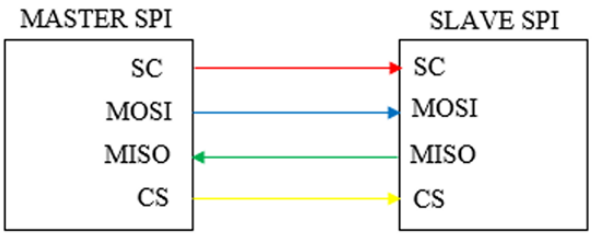

Figura 10: Conexión entre un maestro y un esclavo.[3]

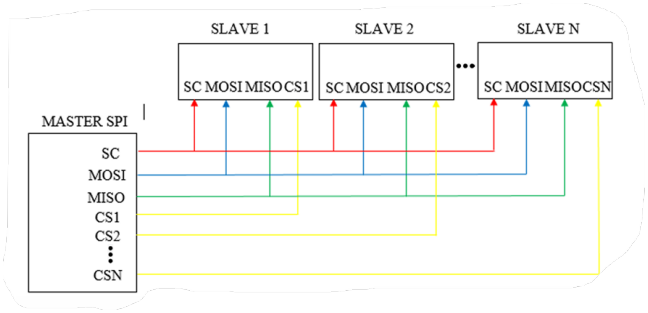

Figura 11: Conexión entre un maestro y "n" esclavos.[3]

2.6.1 Funcionamiento

En la comunicación SPI, el maestro controla el inicio de la transferencia activando la línea de "Chip Select" del esclavo con el que quiere comunicarse y generando la señal de reloj para sincronizar la transferencia de datos con una frecuencia adecuada. A través de la línea MOSI, el maestro envía los datos hacia el esclavo, mientras que el esclavo responde enviando datos por la línea MISO. La escritura y lectura de estos datos ocurre en los flancos del reloj, dependiendo de la configuración de polaridad y fase, asegurando que el maestro y el esclavo lean y escriban en momentos opuestos del ciclo de reloj. Cuando la transmisión de ambos lados finaliza, las líneas MOSI y MISO se colocan en alta impedancia para evitar interferencias con otros dispositivos que puedan estar conectados al bus. Por último, el maestro detiene la señal de reloj y desactiva la línea "Chip Select", indicando que la comunicación ha concluido. Este proceso garantiza una sincronización precisa y eficiente entre los dispositivos que utilizan el protocolo SPI.

2.6.2 Parámetros de configuración

Por otro lado, dentro del protocolo SPI, existen algunos parámetros que determinan el funcionamiento del maestro y el esclavo:

- **Velocidad de transmisión:** Se mide en Baudios (bits/s) y se configura a través de la frecuencia de reloj SC mediante código y es necesario tener en cuenta las restricciones temporales de cada esclavo.

- **Polaridad y Fase:** En la interfaz SPI de los dispositivos esclavos existen 2 bits de configuración que definen los momentos en los que se escriben y se leen los datos en las líneas de MOSI y MISO en una transferencia. Un bit corresponde a la polaridad, CPOL (Clock Polarity) y el otro a la fase, CPHA (Clock Phase). La polaridad establece el estado de reposo de la línea de reloj del bus, es decir, el nivel que tendrá el reloj SC cuando no se está haciendo ninguna transferencia, mientras que la fase determina el flanco de reloj en el que los datos son muestreados y desplazados. Existen combinaciones de estos valores que permiten cuatro modos de comunicación:

  1. **Modo 0 (CPOL = 0, CPHA = 0):** El reloj está en nivel bajo cuando está inactivo, y los datos se muestrean en el flanco ascendente.

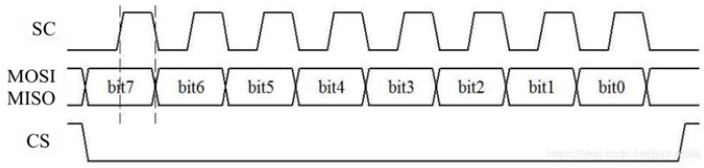

Figura 12: Diagrama de tiempos con polaridad cero y fase cero.[3]

  2. **Modo 1 (CPOL = 0, CPHA = 1):** Similar al modo 0, pero con un retraso de un ciclo de reloj antes de iniciar la transferencia de datos.

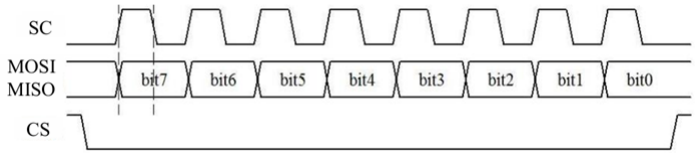

Figura 13: Diagrama de tiempos con polaridad cero y fase uno.[3]

  3. **Modo 2 (CPOL = 1, CPHA = 0):** El reloj está en nivel alto cuando está inactivo, y los datos se muestrean en el flanco descendente.

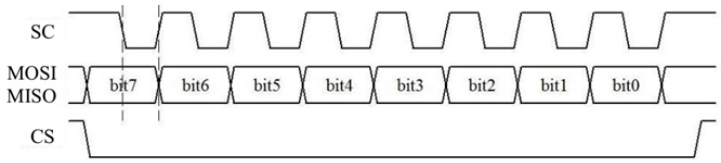

Figura 14: Diagrama de tiempos con polaridad uno y fase cero.[3]

4. **Modo 3 (CPOL = 1, CPHA = 1):** El reloj está en nivel alto cuando está inactivo, y los datos se muestrean en el flanco ascendente tras un ciclo de retraso.

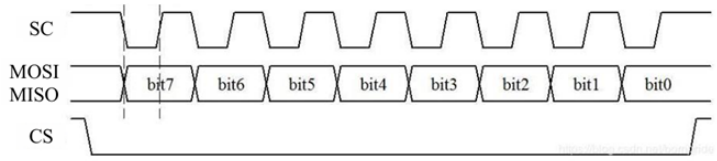

Figura 15: Diagrama de tiempos con polaridad uno y fase uno.[3]

- **Configuraciones de la línea de Chip Select:** Esta línea controlada por el maestro selecciona el dispositivo esclavo con el que va a realizar una transmisión de datos. Puede ser activa en nivel bajo o a nivel alto para la transferencia de datos.

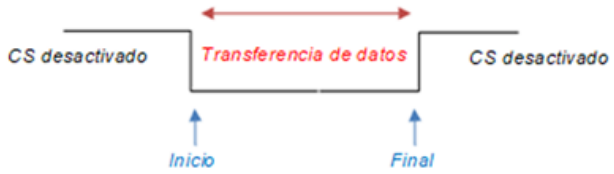

Figura 16: Activación de CS a nivel bajo.[3]

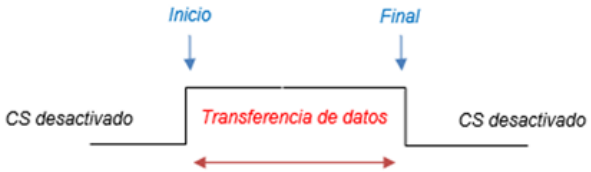

Figura 17: Activación de CS a nivel alto.[3]

- **Longitud de datos a transmitir:** No especifica y no tiene restricción con el numero de bits que tenga el dato, por lo tanto, el dato a enviar puede ser de cualquier tamaño, pero para configurar este parámetro es necesario tener en cuenta las restricciones y especificaciones que tiene cada dispositivo.
- **Orden de transmisión de los bits:** Se configura el orden en el que se deben enviar y/o leer los datos al esclavo y se debe tener en cuenta porque los dispositivos pueden realizar la transmisión de datos de 2 formas: envían primero los bits MSB o los bits LSB.

2.7 Pregunta 7

Bándonos en [4], UART son las iniciales para **universal asynchronous receiver / transmitter**, es un protocolo utilizado para intercambiar información entre dos dispositivos.

La comunicación en UART puede ser simplex (los datos se envían en una sola dirección), half-duplex (cada lado puede hablar, pero solo uno a la vez), o full-duplex (ambos lados pueden transmitir simultáneamente). Los datos en UART se transmiten en forma de tramas.

UART Frame Format

Figura 18: UART Frames utilizan un bit de incio y de final, así como los bit de información y otro bit de paridad. Obtenido de [4]

Bits de Inicio y final

El bit de incio pasa del estado de reposo(HIGH) a un estado lOW que es seguido por la información envíada, el bit del final pasa a un estado(HIGH) que continua por el estado de descanso.

Data Bits

Por lo general se utilizan de 7 a 8 bis para la información, en UART el bit menos signigicativo es el primero en ser leído.

Parity bit

El bit de paridad es un bit opcional que se utiliza para detectar errores. El bit de paridad puede ser 0 o 1 dependiendo de la cantidad de 1's en la señal de información. Si existen una cantidad par de 1's entonces el bit de paridad es 0, de lo cantrario será 1.

El bit de paridad se ubica al final de la señal con los datos, antes del bit final.

Utilización de puertos seriales en Linux

La siguiente página explica detalladamente cómo se pueden utilizar los puertos seriales en Linux,Puertos seriales.

2.8 Pregunta 8

El funcionamiento del controlador ST7789V se puede encontrar en mayor extensión en el siguiente documento ST7789V. Por otro lado en este documento se abordarán de manera muy general su funcionamiento.

Para comenzar se debe seleccionar el tipo de interfaz MPU(Main Processing Unit), esta puede ser alguna de la siguiente lista, los pines dedicados para seleccionar esta interfaz son los IM[3:0].

La interfaz MCU puede leer y escribir datos en la memoria interna del controlador, se utiliza para enviar señales RGB [5]. El controlador ST7789V utiliza las interfases MCU 8080-I y 8080-II. De igual manera los pines IM[3:0] se utilizan para seleccionar la interfaz y su ancho en bits.

La interfaz serial es bidireccional y su función es hacer de intermedario entre el microcontrolador y el driver LCD.

Modo de transferencia de datos, el módulo tiene tres modos de color para transferir información a la RAM display. Estos son color de 12-bit, 16-bit, 18-bit por pixel.

Codificación de color, esta varía según la interfaz que se utilice.

Interfaz RGB, esta se selecciona por medio del RIM y el comando 3Ah, utilzando los pines DB[6:4].

2.8.1 Secuencia de apagado y encendido

VDDI y VDD se pueden aplicar en cualquier orden.

VDD y VDDI se pueden apagar en cualquier orden.

Durante el apagado, si el LCD está en el modo Sleep Out, VDD y VDDI deben apagarse un mínimo de 120 mseg después de que RESX haya sido liberado.

Durante el apagado, si el LCD está en el modo Sleep In, VDDI o VDD se pueden apagar un mínimo de 0 mseg después de que RESX haya sido liberado.

CSX se puede aplicar en cualquier momento o se puede conectar a tierra permanentemente. RESX tiene prioridad sobre CSX.

2.8.2 Niveles de Potencia

Directamente del pdf se definen 6 niveles de potencia, ordenados de mayor a menor consumo de energía:

1. **Modo Normal Activado (pantalla completa), Modo Inactivo Desactivado, Sleep Out.** En este modo, la pantalla puede mostrar un máximo de 262,144 colores.

2. **Modo Parcial Activado, Modo Inactivo Desactivado, Sleep Out.** En este modo, se utiliza una parte de la pantalla con un máximo de 262,144 colores.

3. **Modo Normal Activado (pantalla completa), Modo Inactivo Activado, Sleep Out.** En este modo, se utiliza toda el área de la pantalla, pero con 8 colores.

4. **Modo Parcial Activado, Modo Inactivo Activado, Sleep Out.** En este modo, se utiliza una parte de la pantalla, pero con 8 colores.

5. **Modo Sleep In.** En este modo, el convertidor DC:DC, el oscilador interno y el circuito controlador del panel están detenidos. Solo la interfaz MCU y la memoria funcionan con la fuente de alimentación VDDI. El contenido de la memoria está seguro.

References

[1] D. Harris and S. Harris, *Digital Design and Computer Architecture: ARM Edition*, en. Oxford, Inglaterra: Morgan Kaufmann, 2016.

[2] `https://www.allaboutcircuits.com/textbook/digital/chpt-4/contact-bounce/`, Accessed: 2024-9-4.

[3] T. D. de Una Interfaz Slave Spi Configurable, *PROYECTO FIN DE GRADO*, `https://oa.upm.es/70157/1/TFG_JOSE_BORRALLO_BLANCO.pdf`, Accessed: 2024-9-7.

[4] Rohde, Schwarz GmbH, and K. G. Co, *Understanding UART*, en, `https://www.rohde-schwarz.com/us/products/test-and-measurement/essentials-test-equipment/digital-oscilloscopes/understanding-uart_254524.html`, Accessed: 2024-9-6.

[5] *LCD resources*, en, `https://focuslcds.com/lcd-resources/parallel-interfaces-mcu-vs-rgb/`, Accessed: 2024-9-4, Oct. 2023.