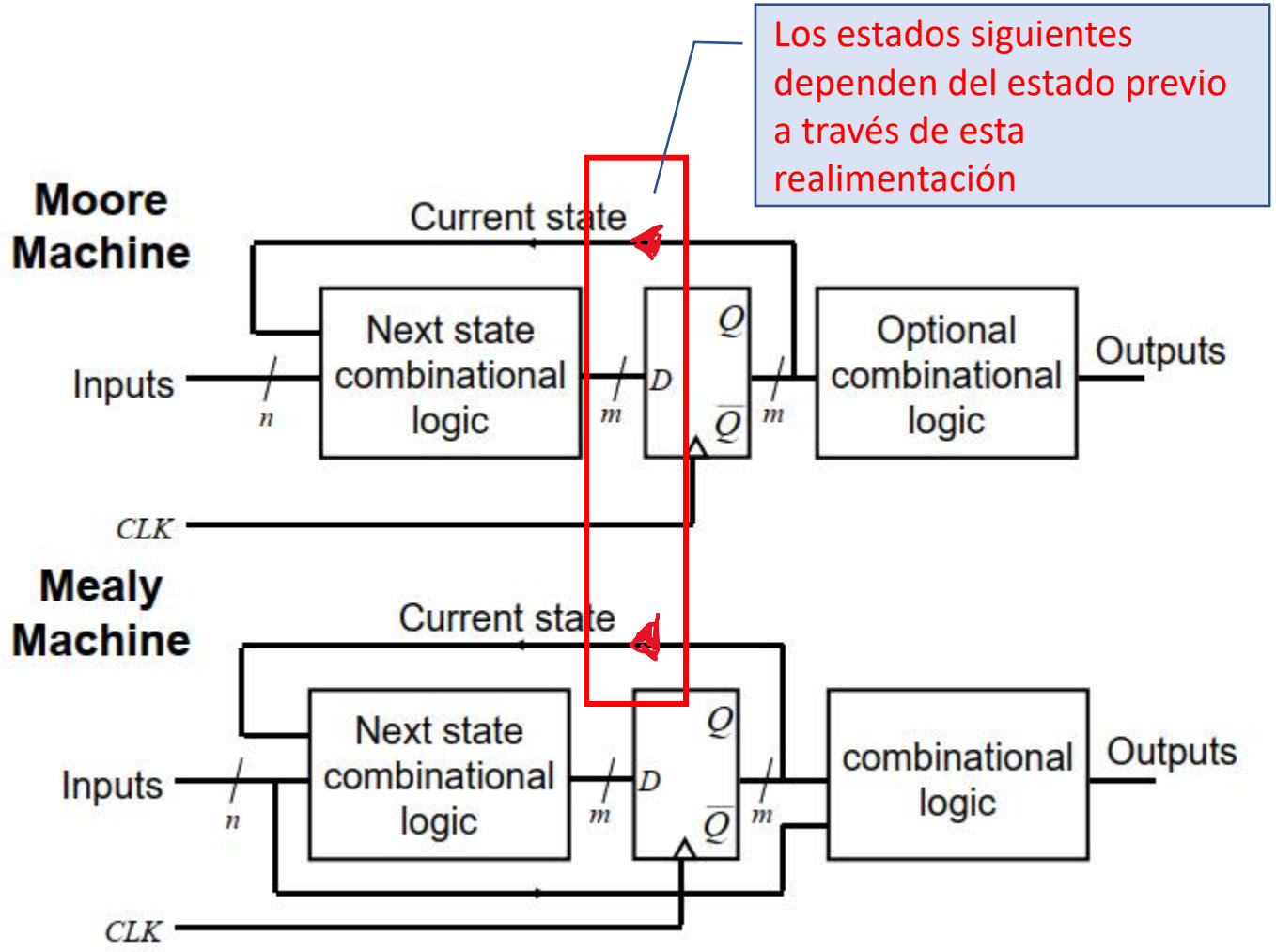
Tema 6. Lógica Secuencial.
Parte 3. Diseño basado en Máquinas de estados

Máquinas de estados síncronas

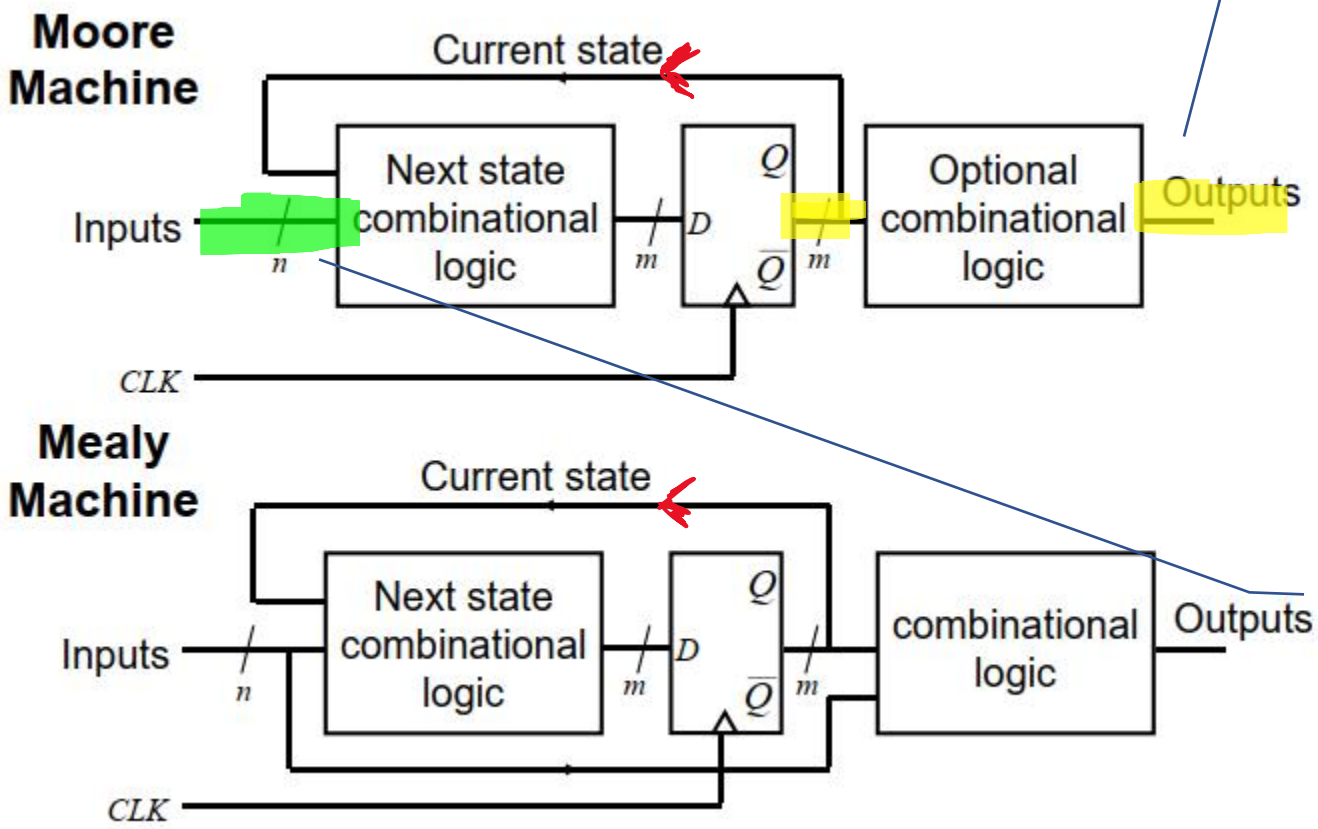
- Se trata de aprovechar lo estudiado hasta ahora sobre el desarrollo de circuitos secuenciales, en particular los contadores, para obtener un **modelo de diseño** más genérico (del que los contadores vistos anteriormente serán un subconjunto)
- Introducimos: Máquina de estados finitos (*Finite State Machine – FSM*)
  - Llamamos **estado** al conjunto de todos los estados internos memorizados.
  - Definición de una **FSM**: cualquier circuito que produce salidas que dependen de las entradas y del estado actual del mismo.
  - Por tanto a mayores de una serie de elementos de memoria y una lógica combinacional una FSM tendrá conectadas:
    - Entradas
    - Salidas
  - Al número de estados de una FSM es **módulo de la FSM**.

Tipos de Máquinas de estados Finitos

- Existen 2 tipos fundamentales:
  - Máquinas de Moore
  - Máquinas de Mealy
- Es una cuestión de modelado del problema:
  - CUALQUIER máquina de Moore tiene una equivalente de Mealy y viceversa.
- Se usan en el desarrollo de circuitos pero también se extiende su uso a las aplicaciones de **automática** en general.



Características de cada tipo de máquina de estados finitos



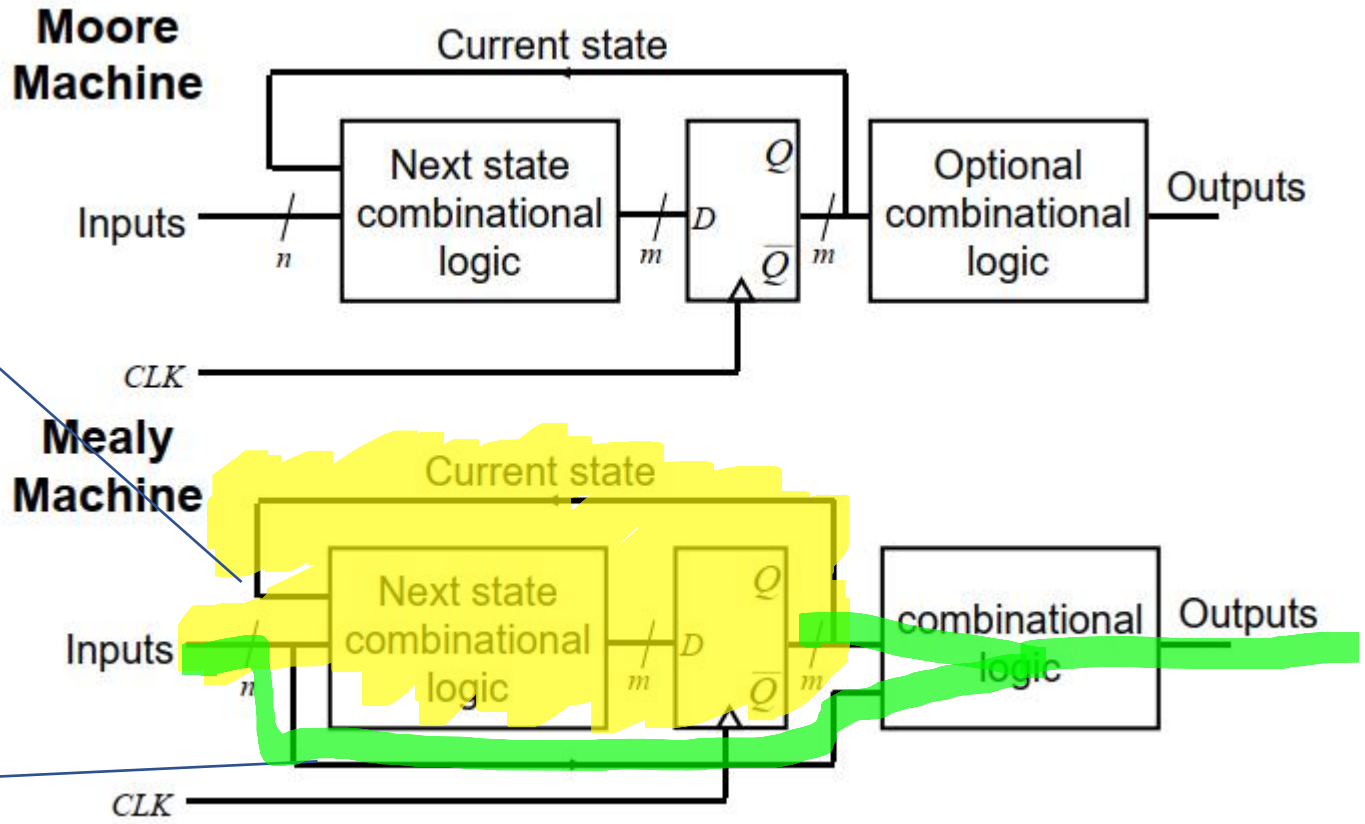
1. Máquina de Moore:
Las salidas se generan con un bloque combinacional (opcional) o directamente desde los biestables. El estado del sistema se refleja directamente en la salida del mismo.

2. Máquina de Moore:
Las entradas solo afectan a la lógica que ocasiona el cambio de estado.

Características de cada tipo de máquina de estados finitos

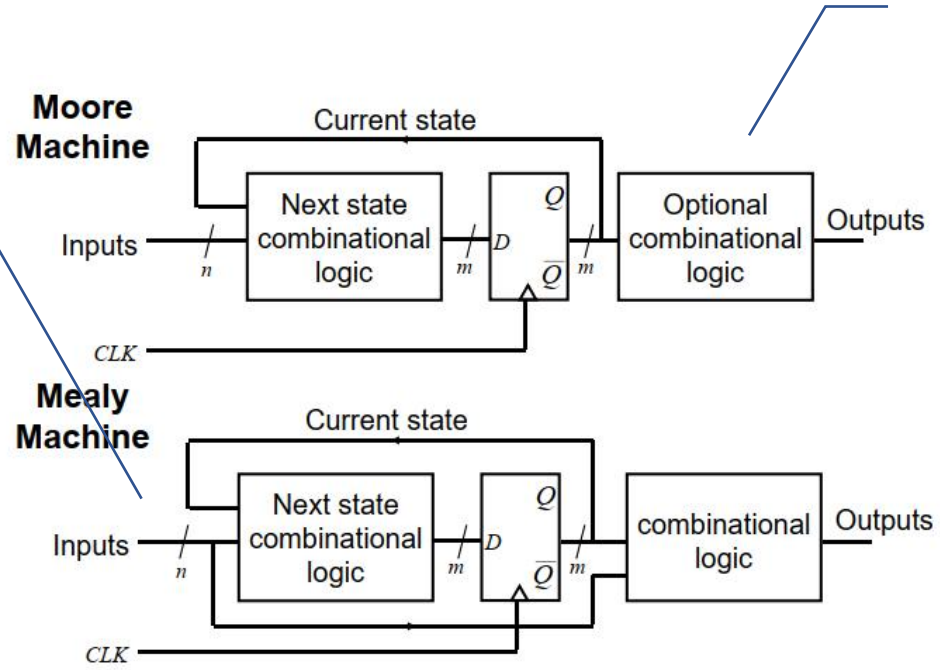
1. Máquina de Mealy:
La zona en amarillo es el bloque que actualiza el estado del sistema, usando las **entradas** y una **realimentación** del estado siguiente.

2. Máquina de Mealy:
El circuito en verde marca los datos que generan realmente la salida. Las entradas se **combinan con los estados** y generan las salidas.



Diferencias entre los distintos tipos de máquinas
Características de comportamiento **temporal** de los circuitos (*timing*)

2. Máquina de Mealy:
Las salidas en las máquinas **combinan entradas al sistema y las salidas de los biestables (estados)** controladas por el CLK.
Las salidas por tanto está influenciadas por "en qué momento" se produce el cambio de las señales de entrada.



1. Máquina de Moore:
Las salidas en las máquinas de Moore están conectadas directamente a una serie de biestables (FF) cuyas salidas están controladas por el reloj del sistema.
Las salidas solo dependen de los estados.
Los estados, por tanto **sólo cambian en el momento que marca el reloj** (CLK)

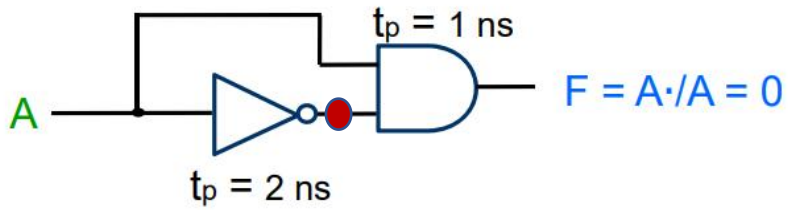
3. Son más fáciles de manejar desde el punto de vista del *timing* las máquinas de Moore
4. Las máquinas de Mealy pueden presentar un problema de timing denominado *glitch:* Se trata de valores espurios en las salidas, por cambios en las entradas en momentos con un sincronismo inadecuado con las señales de reloj, valores transitorios que no se corresponden con la especificación del circuito.
Nota: a veces a los *glitches* también se les llama "riesgos" (se explican un poco más adelante en más detalle)

- La máquina de Moore por tanto es SÍNCRONA, tenemos salida después del ciclo de reloj.
- La máquina de Mealy es ASÍNCRONA. El reloj controla los cambios de estado en los elementos de memoria FF, pero cuando cambien las entradas puede causar un cambio de estado.

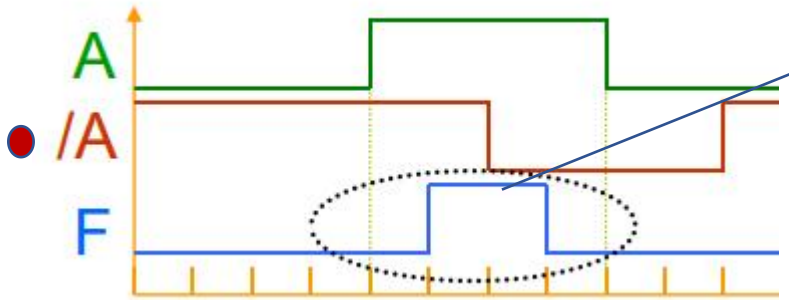
- Inciso sobre timing, glitch, ….

El *glitch*

- La problemática del *timing* es algo que ya aparecía en los circuitos secuenciales.
- Consideremos el circuito de la figura donde los tiempo de propagación t- son de 1 ns para la puerta AND y 2ns para el inversor. El circuito así definido debería dar una salida siempre a LOW.



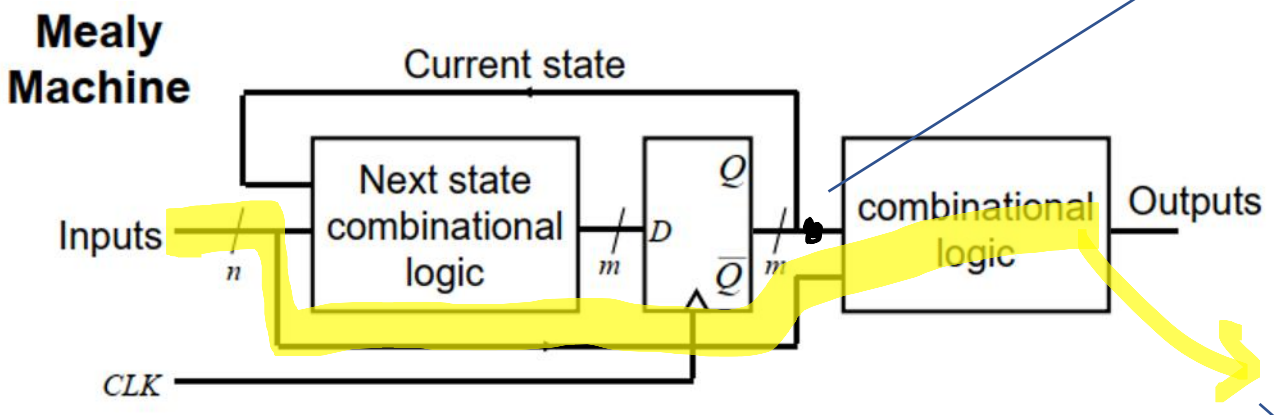
- Analicemos un cronograma para el mismo:



Hay un momento en el que la salida toma el valor HIGH (incorrecto) a causa de que el inversor es más lento en actualizar su salida que la puerta AND.

En algunos circuitos esto puede ser un problema y en otros no. Depende del diseño.

El *glitch* (2)



En este punto se actualiza Q con el CLK

El flujo de señales por este otro camino de datos va por fuera del sincronismo.

Por tanto no se garantiza que todas las entradas lleguen a la vez.
Puede haber un momento con transitorios

Moore vs Mealy

- En general es más simple de construir un diseño según el estilo de la máquina de Moore, al no tener que preocuparnos tanto de los efectos de *glitch*.
- Sin embargo nos encontraremos a menudo con que una aproximación al problema con la máquina de Mealy nos lleva a un resultado con menos estados implicados (**menor módulo**)
  - Es cómo si los estados que se ahorran, no se necesita tenerlos guardados en elementos de memoria, porque se van generando cuando se necesitan al combinar el estado actual con las entradas en cada momento.

Ejemplo de aplicación. Máquina de Moore.

- **Control de semáforo**.
  - Se trata de una aplicación del método de desarrollo de una máquina según la aproximación de Moore.
  - El diseño de un contador síncrono con *flip-flops* en la parte 2 de este tema también estaba concebido como una máquina de Moore.
  - Al igual que en aquel caso se trata de una máquina de estados bastante simple.

Ejemplo de aplicación. Comparación máquina de Mealy- máquina de Moore.

- **Circuito detector de errores**.
  - Una aplicación de este circuito sería el análisis de una trama de datos serie bit a bit transmitida por un canal en tiempo real, para detectar situaciones de error de transmisión.
  - En nuestro ejemplo consideraremos “error” al caso en que aparece una secuencia de 2 valores LOW o 3 valores HIGH consecutivos. Un caso más realista podría incluir detectar alguna situación de fin de trama y realizar una operación lógica con los bits completos de la trama recibida.
  - Objetivo del circuito:
    - Mostrar en una salida Z un valor alto cuando se detecta un error
    - Realizar la implementación con biestables tipo D

Ejemplo de aplicación. Reconocedor de patrones.

- **Circuito reconocedor de patrones binarios**
  - Para una entrada consistente en un *stream* de bits en serie diseñar un circuito secuencial basado en máquina de estados finitos capaz de encontrar los patrones “1,1,0” y “1,0,1” en una secuencia de entrada serie.
  - Cuando detectemos el patrón activaremos una salida Z=1 como indicador.
  - El circuito debe ser capaz de detectar el patrón de forma solapada como se indica en el diagrama siguiente