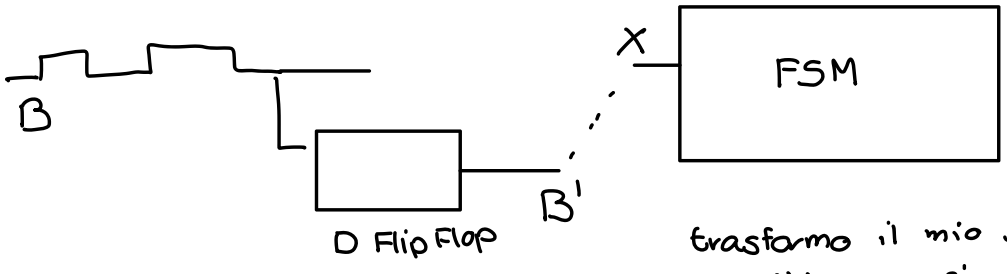
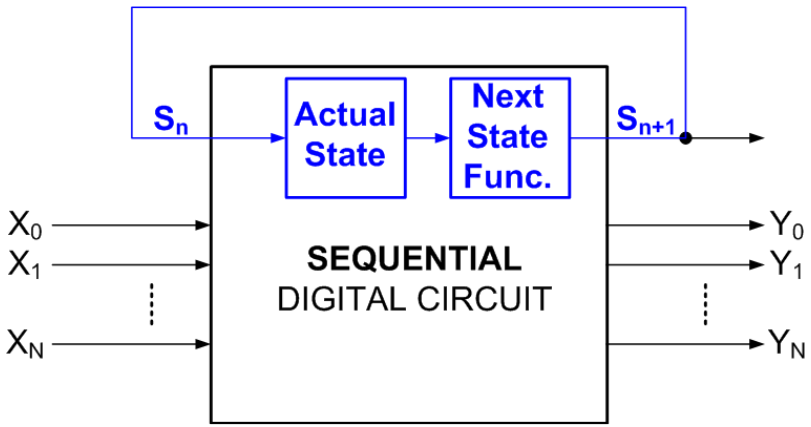
-> Finite State Machines



trasformo il mio segnale di input asincrono in sincrono

Finite-State-Machines

Introduction



Sequential Digital Circuits can be:

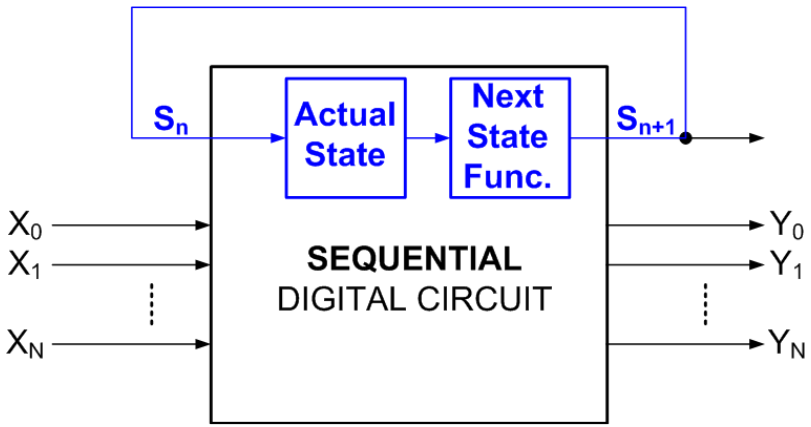
- Synchronous → they use a clock signal (and edge-triggered FF)
- Asynchronous → they use asynchronous memory elements (latches)

Synchronous Digital Circuits are more robust because they are less sensitive to:

- Static and dynamic hazard (the system does not evolve towards unwanted states if clock signal period is longer than any switching time delay)
- Unwanted glitches and/or disturbs, affecting Actual State

Finite-State-Machines

Introduction



FSMs are <u>COMPLETELY DEFINED BY</u> the following signals:

- Input Signals (X)
- Output Signals (Y)

and

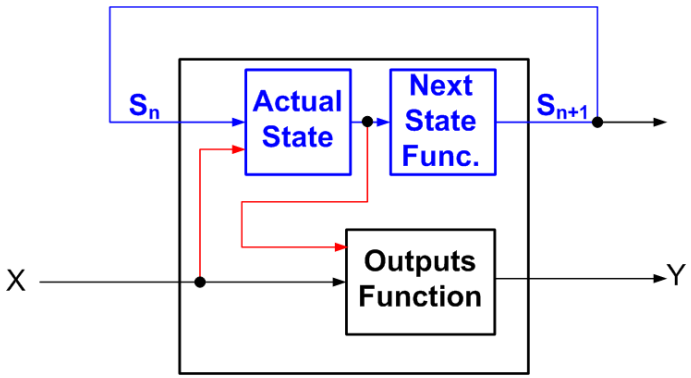
- States List (Actual State)
- Outputs Function (f_{OUT})
- Next-State Function (f_{NS})

Finite-State-Machines
Outline

- Introduction
- **Mealy and Moore FSM ←←←**
- State Diagram
- State Table
- Sequential Synchronous FSM Circuital Synthesis
- Exercise 1
- Exercise 2
- Exercise 3

Finite-State-Machines

Mealy FSMs



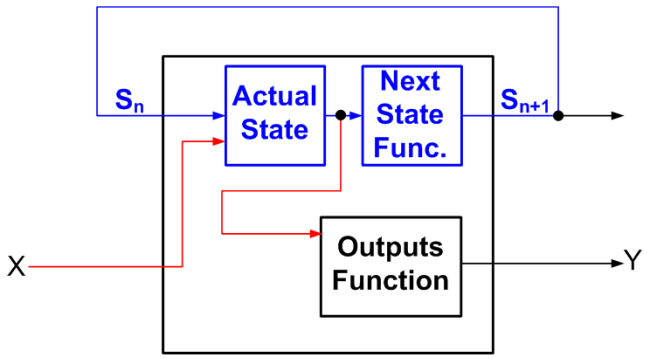
In Mealy-FSM the output Y is a function of both actual state (Sn) and input (X).

Mealy FSM are <u>COMPLETELY DEFINED BY</u>:

- Input Signals (X)
- Output Signals (Y)
- States List (Actual State)
- Outputs Function f_{OUT}, where $Y=f_{OUT}(X,S_n)$
- Next-State Function f_{NS}, where $S_{n+1}=f_{NS}(X,S_n)$

Finite-State-Machines

Moore FSMs



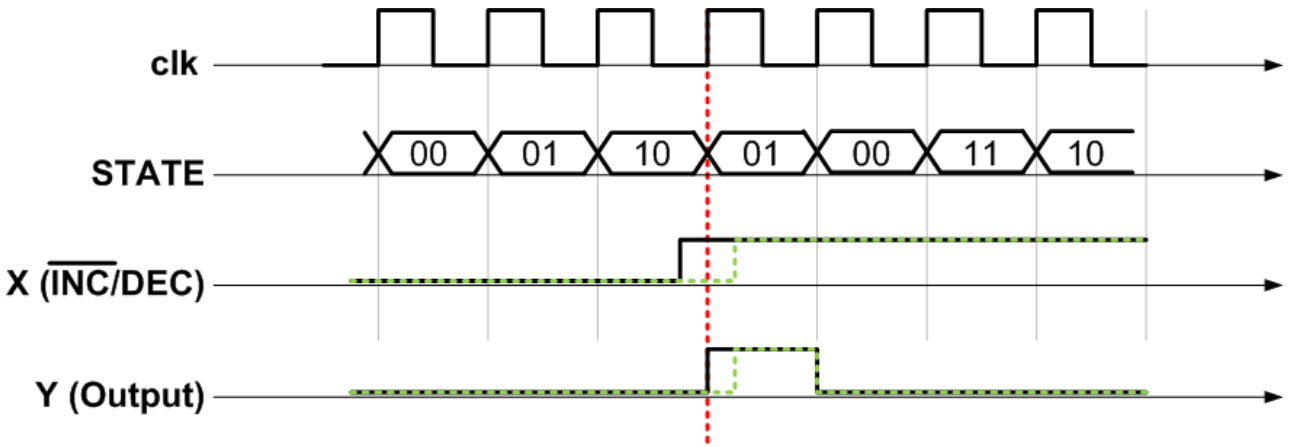
In Moore-FSM the output Y is a function of only actual state (Sn).

Moore FSM are <u>COMPLETELY DEFINED BY</u>:

- Input Signals (X)
- Output Signals (Y)
- States List (Actual State)
- Outputs Function f_{OUT}, where $Y=f_{OUT}(S_n)$
- Next-State Function f_{NS}, where $S_{n+1}=f_{NS}(X,S_n)$

Finite-State-Machines

Mealy FSMs



Case Study → 2b-Counter with '01' Output Flag
Operation: Y (Output) is '1' when Counter State is "01" and X='1'

Input Signals: clk, X (increment or decrement signal)
Output Signals: STATE<0:1>, Y
Output Logic Function: $Y = f_{OUT}(S_n, X) = \overline{S_n(1)} \cdot S_n(0) \cdot X$
→→→ **Mealy FSMs** can have asynchronous behavior (Y depends on X)
Time evolution depends on X delays

Finite-State-Machines

Design Procedure

FSM Design Procedure

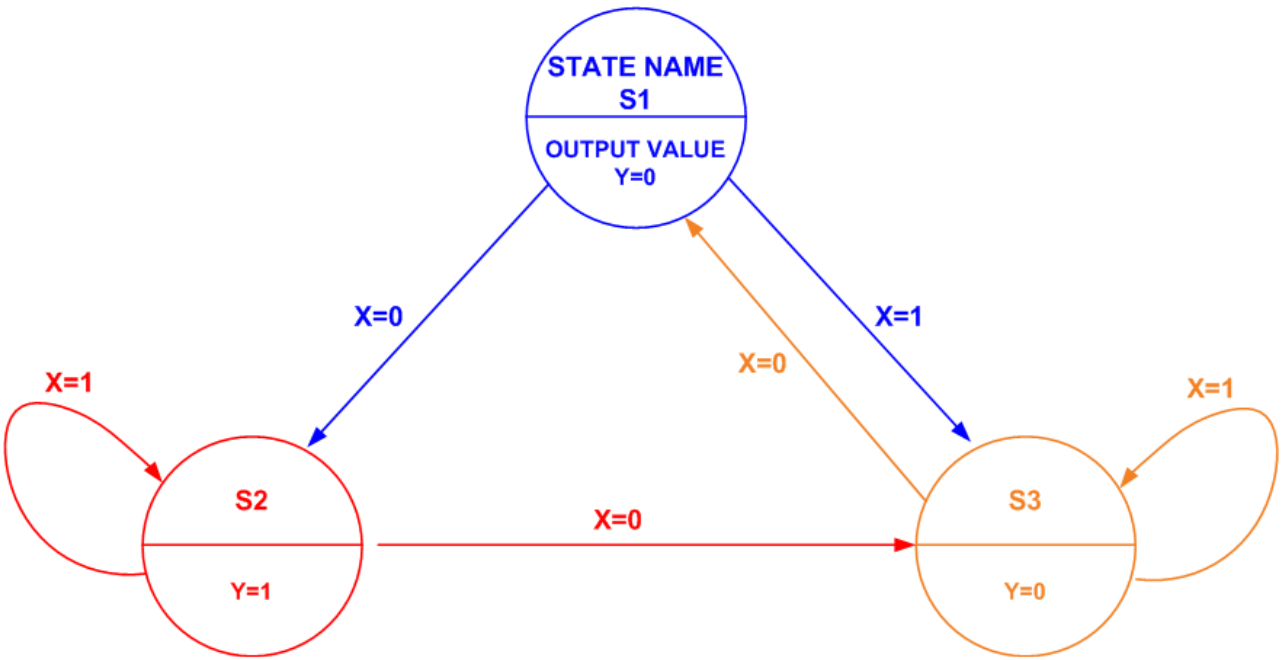
- **Define**
  - Input Signals (X)
  - Output Signals (Y)
  - States List (Actual State)
- **State-Diagram**
  - Graphical Representation of all states and transitions (from one state to another)
- **State-Table**
  - A table whit two main columns regions:
    - The first region points the actual state (S_n) and the input signals (X)
    - The second the next states and the output signals (S_{n+1} and Y)
- **Next-State Function f_{NS}**
  - where $S_{n+1}=f_{NS}(X,S_n)$ by KMAPs or COMBINATORY logic synthesis
- **Outputs Function f_{OUT}**
  - where $Y=f_{OUT}(S_n)$ or $Y=f_{OUT}(X,S_n)$ by KMAPs or COMBINATORY logic synthesis

Finite-State-Machines
Outline

- Introduction
- Mealy and Moore FSM
- **State Diagram ←←←**
- State Table
- Sequential Synchronous FSM Circuital Synthesis
- Exercise 1
- Exercise 2
- Exercise 3

Finite-State-Machines

State-Diagram



Each STATE has:

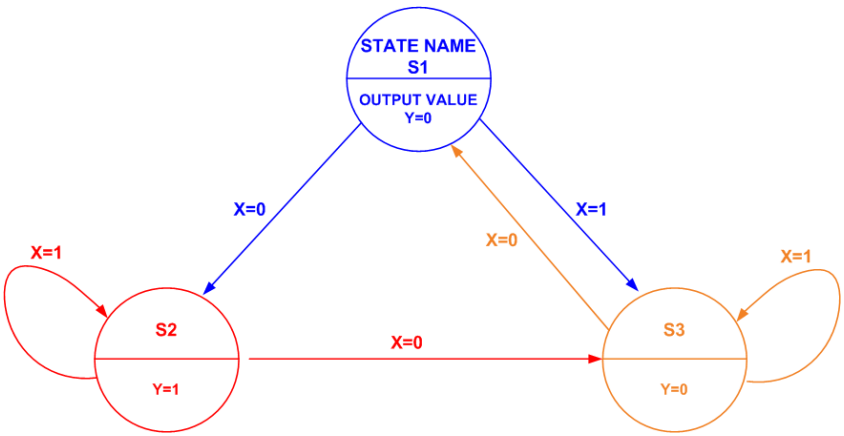
- State Name (Ex: S0, S1, S2, ..)
- Output Value (Ex: Y=0 or Y=1)
- Transition Arrows (one for each input value)

Finite-State-Machines
Outline

- Introduction
- Mealy and Moore FSM
- State Diagram
- **State Table ←←←**
- Sequential Synchronous FSM Circuital Synthesis
- Exercise 1
- Exercise 2
- Exercise 3

Finite-State-Machines

State-Table



ACTUAL STATE	INPUT	NEXT STATE	OUTPUT
S1	0	**S2**	**Y=1**
S1	1	**S3**	**Y=0**
S2	0	**S3**	**Y=0**
S2	1	**S2**	**Y=1**
S3	0	**S1**	**Y=0**
S3	1	**S3**	**Y=0**

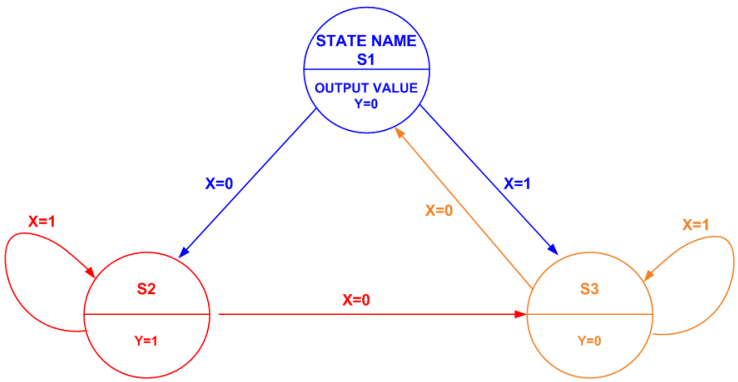
State Table

Each STATE Table has the following columns:

- Actual State Name (Ex: S0, S1, S2, ..)
- Input Value (Ex: X=0 or X=1)
- Next State (Ex: S0, S1, S2, ..)
- Output Value (Ex: Y=0 or Y=1)

Finite-State-Machines

State-Table



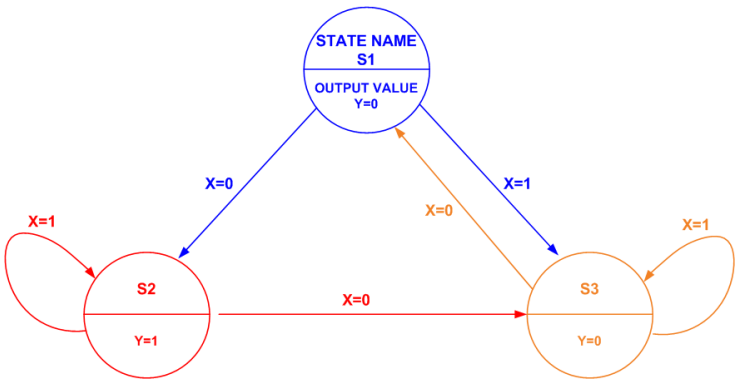
ACTUAL STATE			INPUT	NEXT STATE			OUTPUT
S1	0	0	0	**S2**	**0**	**1**	**Y=1**
S1	0	0	1	**S3**	**1**	**0**	**Y=0**
S2	0	1	0	**S3**	**1**	**0**	**Y=0**
S2	0	1	1	**S2**	**0**	**1**	**Y=1**
S3	1	0	0	**S1**	**0**	**0**	**Y=0**
S3	1	0	1	**S3**	**1**	**0**	**Y=0**

Each STATE Table has the following columns:

- Actual State Name (Ex: S0, S1, S2, ..)
  - **Each State has a specific binary code (EX: S1=00, S2=01, S3=10, etc)**
- Input Value (Ex: X=0 or X=1)
- Next State (Ex: S0, S1, S2, ..)
- Output Value (Ex: Y=0 or Y=1)

Finite-State-Machines

State-Table



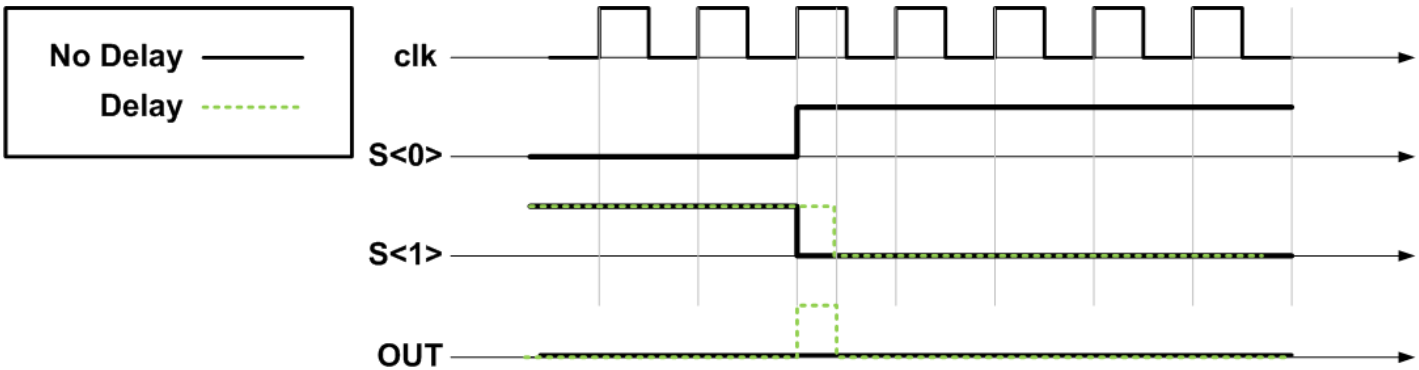
ACTUAL STATE			INPUT	NEXT STATE			OUTPUT
S1	0	0	0	**S2**	**0**	**1**	**Y=1**
S1	0	0	1	**S3**	**1**	**1**	**Y=0**
S2	0	1	0	**S3**	**1**	**1**	**Y=0**
S2	0	1	1	**S2**	**0**	**1**	**Y=1**
S3	1	1	0	**S1**	**0**	**0**	**Y=0**
S3	1	1	1	**S3**	**1**	**1**	**Y=0**

Each STATE Table has the following columns:

- Actual State Name (Ex: S0, S1, S2, ..)
  - **Each State has a specific binary code (EX: S1=00, S2=01, S3=10, etc)**
  - **In case of ≤4 states, Grey Code does not resolve unwanted states evolution (due to delay)**
- Input Value (Ex: X=0 or X=1)
- Next State (Ex: S0, S1, S2, ..)
- Output Value (Ex: Y=0 or Y=1)

Finite-State-Machines

State-Table – Unwanted States and Transitions in Moore FSMs

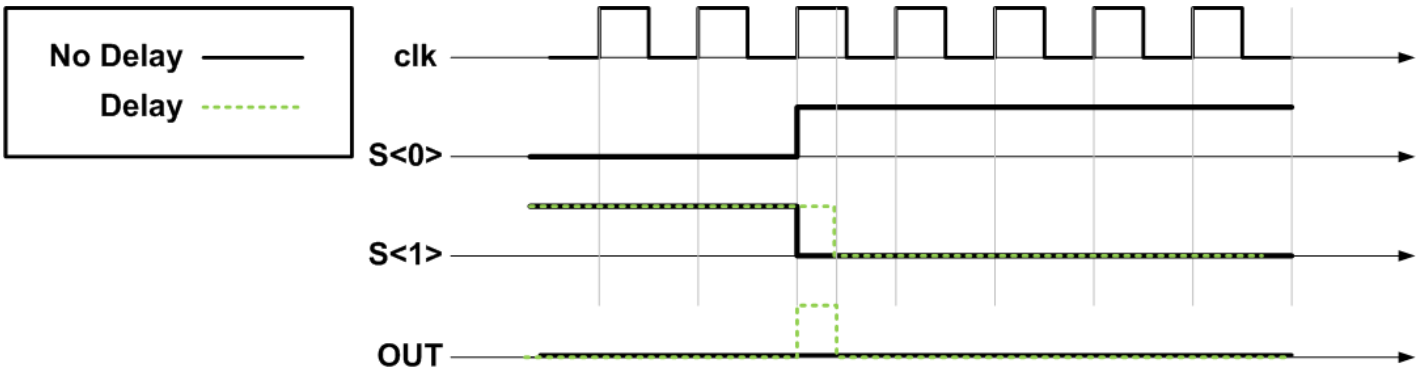


ACTUAL STATE			NEXT STATE			OUTPUT
	S<1>	S<0>		S<1>	S<0>	
S2	1	0	**S1**	**0**	**1**	**Y=0**
S1	0	1	**SX**	**x**	**x**	**Y=0**
S3	1	1	**S0**	**0**	**0**	**Y=1**

- Let analyze the S2→S1 transition (look at the States Table)
- Contemporary switching is “statistically rare”
- Suppose S1 switches-down later than S0 switching-up
- The system is for a small time in S3=11 state
- In this state the output Y is 1, instead of 0

Finite-State-Machines

State-Table – Unwanted States and Transitions in Moore FSMs



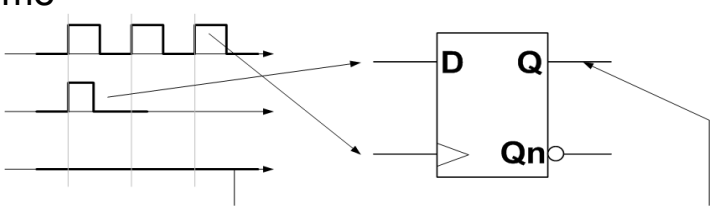
☺ Good Points:

- State S3(=11) is only for a small time, since Moore-FSM are based on DFF that guarantees to evolve from S2 towards S1 state
- The system is naturally forced to evolve towards S1

☹ Bad Points

- Output Signal is '1' for a small time

Solution →



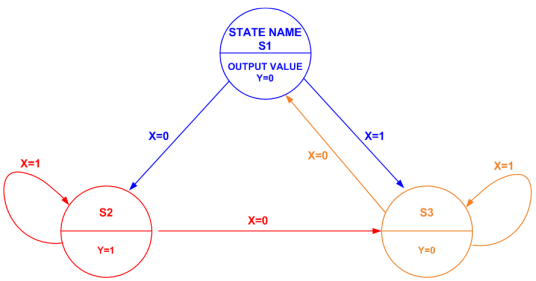
Finite-State-Machines

Outline

- Introduction
- Mealy and Moore FSM
- State Diagram
- State Table
- **Sequential Synchronous FSM Circuital Synthesis ←←←**
- Exercise 1
- Exercise 2
- Exercise 3

Finite-State-Machines

Moore-FSM Design



ACTUAL STATE S_n			INPUT	NEXT STATE S_{n+1}			OUTPUT
	Sn(1)	Sn(0)	X		Sn+1(1)	Sn+1(0)	Y
S1	0	0	0	**S2**	**0**	**1**	**1**
S1	0	0	1	**S3**	**1**	**1**	**0**
S2	0	1	0	**S3**	**1**	**1**	**0**
S2	0	1	1	**S2**	**0**	**1**	**1**
S3	1	1	0	**S1**	**0**	**0**	**0**
S3	1	1	1	**S3**	**1**	**1**	**0**

NEXT-STATE Function $f_{NS}(S_n,X)$:

- NEXT-STATE BITs ($S_{n+1}(1)$ and $S_{n+1}(0)$) are logic functions of
  - $S_n(1)$, $S_n(0)$ (ACTUAL-STATE-BITS) and X (INPUT)
  - Example:
    - Sn+1 Logic Function

$$S_{n+1}(1) = f_{SN1}(S_n, X) = S_n(1) \cdot X + \overline{S_n(0)} \cdot X + \overline{S_n(1)} \cdot S_n(0) \cdot \overline{X}$$

$$S_{n+1}(0) = f_{SN0}(S_n, X) = \overline{S_n(1)} + X$$

    - Y Logic Function

$$Y = f_{OUT}(S_n) = \overline{S_n(0)} \cdot \overline{X} + \overline{S_n(1)} \cdot S_n(0) \cdot X$$

Finite-State-Machines
Outline

- Introduction
- Mealy and Moore FSM
- State Diagram
- State Table
- **Sequential Synchronous FSM Circuital Synthesis**
- **Exercise 1 ←←←**
- **Exercise 2 ←←←**
- **Exercise 3 ←←←**

Finite-State-Machines

Exercise

Exercise 1:
Design a logic circuit able to detect two equal consecutive bits.
In case of positive event, the Y output signal is 1, otherwise is 0.
The circuit receives one single bit (X) and uses one single clock.
Implement by Moore-FSM.

Exercise 2:
Design a logic circuit that detects the 010 sequence.
The circuit has only one digital input of 1 bit resolution.

Exercise 3:
Design a logic circuit that detects the 110 sequence.
The circuit has only one digital input of 1 bit resolution.

Finite-State-Machines

Exercise

Exercise 1:←←←
Design a logic circuit able to detect that two consecutive bits are equal.
In case of positive event, the Y output signal is 1, otherwise is 0.
The circuit receives one single bit (X) and uses one single clock.
Implement by Moore-FSM.

FSM Design Procedure

- **Define**
  - Input Signals (X)
  - Output Signals (Y)
  - States List (Actual State)
- **State-Diagram**
  - Graphical Representation of all states and transitions (from one state to another)
- **State-Table**
  - A table whit two main columns regions:
    - The first region points the actual state (S_n) and the input signals (X)
    - The second the next states and the output signals (S_{n+1} and Y)
- **Next-State Function f_{NS}**
  - where $S_{n+1}=f_{NS}(X,S_n)$ by KMAPs or COMBINATORY logic synthesis
- **Outputs Function f_{OUT}**
  - where $Y=f_{OUT}(S_n)$ or $Y=f_{OUT}(X,S_n)$ by KMAPs or COMBINATORY logic synthesis

Finite-State-Machines
Exercise

Exercise 1:

FSM Design Procedure

- **Define**
  - Input Signals (X)
  - Output Signals (Y)

X, 1b resolution
Y, 1b resolution

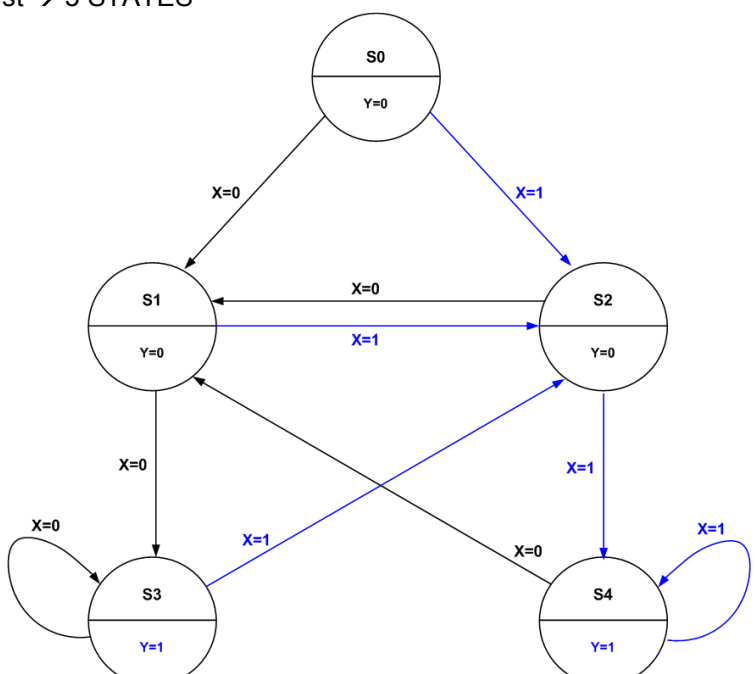
- **State-Diagram**
  - Graphical Representation of all states and transitions (from one state to another)
- **State-Table**
  - A table whit two main columns regions:
    - The first region points the actual state (S_n) and the input signals (X)
    - The second the next states and the output signals (S_{n+1} and Y)
- **Next-State Function f_{NS}**
  - where $S_{n+1}=f_{NS}(X,S_n)$ by KMAPs or COMBINATORY logic synthesis
- **Outputs Function f_{OUT}**
  - where $Y=f_{OUT}(S_n)$ or $Y=f_{OUT}(X,S_n)$ by KMAPs or COMBINATORY logic synthesis

Finite-State-Machines
Exercise

Exercise 1:

FSM Design Procedure

- **State-Diagram**
  - Graphical Representation of all states and transitions (from one state to another)
  - Define States List → 5 STATES



Finite-State-Machines

Exercise

Exercise 1:

FSM Design Procedure

- **State-Table**
  - A table whit two main columns regions:
    - The first region points the actual state (S_n) and the input signals (X)
    - The second the next states and the output signals (S_{n+1} and Y)

ACTUAL STATES				INPUT	NEXT STATES				OUTPUT
STATE NAME	$S_n(2)$	$S_n(1)$	$S_n(0)$	X	STATE NAME	$S_{n+1}(2)$	$S_{n+1}(1)$	$S_{n+1}(0)$	Y
S0	0	0	0	0	S1	0	0	1	0
S0	0	0	0	1	S2	0	1	0	0
S1	0	0	1	0	S3	0	1	1	1
S1	0	0	1	1	S2	0	1	0	0
S2	0	1	0	0	S1	0	0	1	0
S2	0	1	0	1	S4	1	0	0	1
S3	0	1	1	0	S3	0	1	1	1
S3	0	1	1	1	S2	0	1	0	0
S4	1	0	0	0	S1	0	0	1	0
S4	1	0	0	1	S4	1	0	0	1

Finite-State-Machines

Exercise

Exercise 1:

FSM Design Procedure

- **Next-State Function f_{NS}**
  - where $S_{n+1}=f_{NS}(X,S_n)$ by KMAPs or COMBINATORY logic synthesis

ACTUAL STATES				INPUT	NEXT STATES				OUTPUT
STATE NAME	$S_n(2)$	$S_n(1)$	$S_n(0)$	X	STATE NAME	$S_{n+1}(2)$	$S_{n+1}(1)$	$S_{n+1}(0)$	Y
S0	0	0	0	0	S1	0	0	1	0
S0	0	0	0	1	S2	0	1	0	0
S1	0	0	1	0	S3	0	1	1	1
S1	0	0	1	1	S2	0	1	0	0
S2	0	1	0	0	S1	0	0	1	0
S2	0	1	0	1	S4	1	0	0	1
S3	0	1	1	0	S3	0	1	1	1
S3	0	1	1	1	S2	0	1	0	0
S4	1	0	0	0	S1	0	0	1	0
S4	1	0	0	1	S4	1	0	0	1

Critical Transitions occur when at least two Sn bits contemporarily change

Finite-State-Machines

Exercise

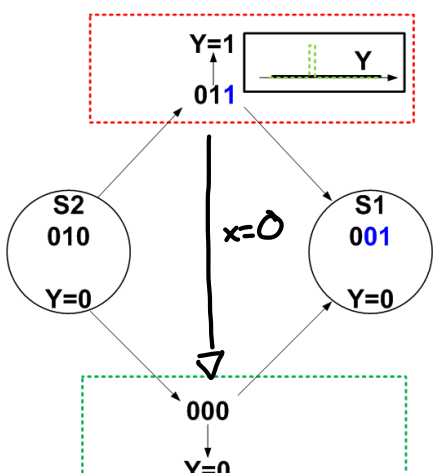
Exercise 1:

FSM Design Procedure

- **Next-State Function f_{NS}**
  - where $S_{n+1}=f_{NS}(X,S_n)$ by KMAPs or COMBINATORY logic synthesis

ACTUAL STATES				INPUT	NEXT STATES				OUTPUT
STATE NAME	$S_n(2)$	$S_n(1)$	$S_n(0)$	X	STATE NAME	$S_{n+1}(2)$	$S_{n+1}(1)$	$S_{n+1}(0)$	Y
S2	0	1	0	0	S1	0	0	1	0

Critical Transitions occur when two Sn bits contemporarily change: 010 → 001



non riesco a cambiare due bit contemporaneamente

mi va bene perché Y=0

Finite-State-Machines
Exercise

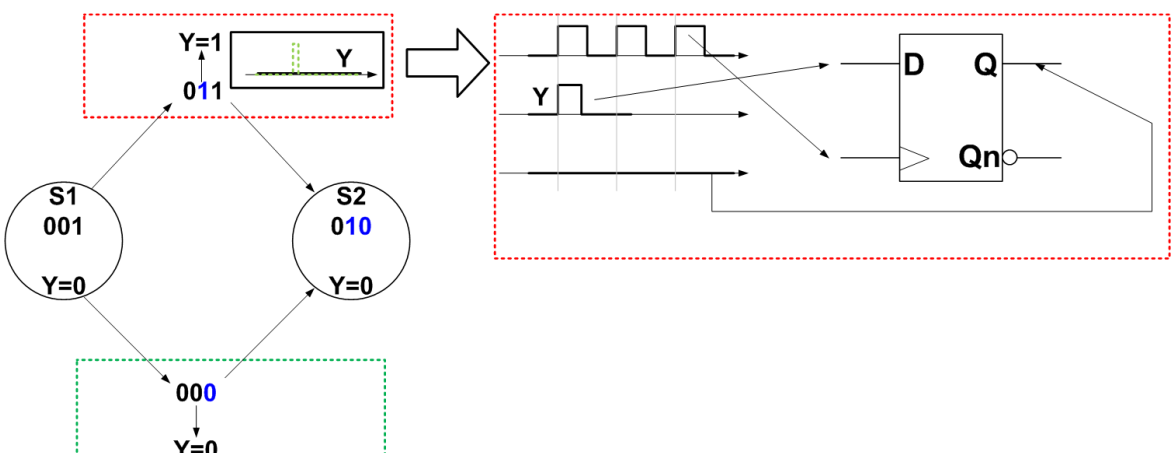
Exercise 1:

FSM Design Procedure

- **Next-State Function f_{NS}**
  - where $S_{n+1}=f_{NS}(X,S_n)$ by KMAPs or COMBINATORY logic synthesis

ACTUAL STATES				INPUT	NEXT STATES				OUTPUT
STATE NAME	$S_n(2)$	$S_n(1)$	$S_n(0)$	X	STATE NAME	$S_{n+1}(2)$	$S_{n+1}(1)$	$S_{n+1}(0)$	Y
S2	0	1	0	1	S4	1	0	0	1

Critical Transitions occur when two Sn bits contemporarily change: 001 → 010



Finite-State-Machines
Exercise

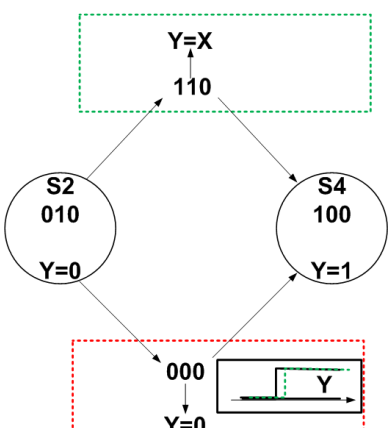
Exercise 1:

FSM Design Procedure

- **Next-State Function f_{NS}**
  - where $S_{n+1}=f_{NS}(X,S_n)$ by KMAPs or COMBINATORY logic synthesis

ACTUAL STATES				INPUT	NEXT STATES				OUTPUT
STATE NAME	$S_n(2)$	$S_n(1)$	$S_n(0)$	X	STATE NAME	$S_{n+1}(2)$	$S_{n+1}(1)$	$S_{n+1}(0)$	Y
S2	0	1	0	1	S4	1	0	0	1

Critical Transitions occur when two Sn bits contemporarily change: 010 → 100



Finite-State-Machines
Exercise

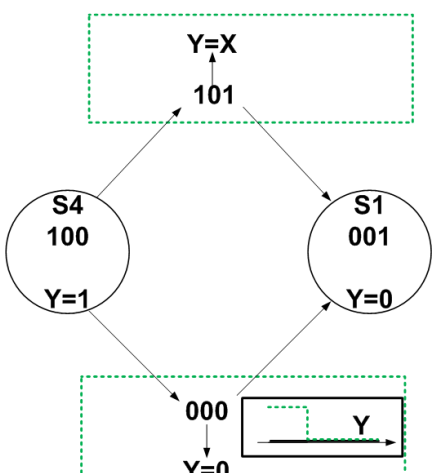
Exercise 1:

FSM Design Procedure

- **Next-State Function f_{NS}**
  - where $S_{n+1}=f_{NS}(X,S_n)$ by KMAPs or COMBINATORY logic synthesis

ACTUAL STATES				INPUT	NEXT STATES				OUTPUT
STATE NAME	$S_n(2)$	$S_n(1)$	$S_n(0)$	X	STATE NAME	$S_{n+1}(2)$	$S_{n+1}(1)$	$S_{n+1}(0)$	Y
S4	1	0	0	0	S1	0	0	1	0

Critical Transitions occur when two Sn bits contemporarily change: 100 → 001



Finite-State-Machines

Exercise

Exercise 1:

FSM Design Procedure

- **Next-State Function f_{NS}**
  - where $S_{n+1}=f_{NS}(X,S_n)$ by KMAPs or COMBINATORY logic synthesis

ACTUAL STATES				INPUT	NEXT STATES				OUTPUT
STATE NAME	$S_n(2)$	$S_n(1)$	$S_n(0)$	X	STATE NAME	$S_{n+1}(2)$	$S_{n+1}(1)$	$S_{n+1}(0)$	Y
S0	0	0	0	0	S1	0	0	1	0
S0	0	0	0	1	S2	0	1	0	0
S1	0	0	1	0	S3	0	1	1	1
S1	0	0	1	1	S2	0	1	0	0
S2	0	1	0	0	S1	0	0	1	0
S2	0	1	0	1	S4	1	0	0	1
S3	0	1	1	0	S3	0	1	1	1
S3	0	1	1	1	S2	0	1	0	0
S4	1	0	0	0	S1	0	0	1	0
S4	1	0	0	1	S4	1	0	0	1

Sn+1(2)		Sn(2) Sn(1)			
		00	01	11	10
Sn(0) X	00	0	0	x	0
	01	0	1	x	1
	11	0	0	x	x
	10	0	0	x	x

$$S_{n+1}(2) = S_n(1) \cdot \overline{S_n(0)} \cdot X + S_n(2) \cdot X$$

Finite-State-Machines

Exercise

Exercise 1:

FSM Design Procedure

- **Next-State Function f_{NS}**
  - where $S_{n+1}=f_{NS}(X,S_n)$ by KMAPs or COMBINATORY logic synthesis

ACTUAL STATES				INPUT	NEXT STATES				OUTPUT
STATE NAME	$S_n(2)$	$S_n(1)$	$S_n(0)$	X	STATE NAME	$S_{n+1}(2)$	$S_{n+1}(1)$	$S_{n+1}(0)$	Y
S0	0	0	0	0	S1	0	0	1	0
S0	0	0	0	1	S2	0	1	0	0
S1	0	0	1	0	S3	0	1	1	1
S1	0	0	1	1	S2	0	1	0	0
S2	0	1	0	0	S1	0	0	1	0
S2	0	1	0	1	S4	1	0	0	1
S3	0	1	1	0	S3	0	1	1	1
S3	0	1	1	1	S2	0	1	0	0
S4	1	0	0	0	S1	0	0	1	0
S4	1	0	0	1	S4	1	0	0	1

Sn+1(1)		Sn(2) Sn(1)			
		00	01	11	10
Sn(0) X	00	0	0	x	0
	01	1	0	x	0
	11	1	1	x	x
	10	1	1	x	x

$$S_{n+1}(1) = S_n(0) + \overline{S_n(2)} \cdot \overline{S_n(1)} \cdot X$$

Finite-State-Machines

Exercise

Exercise 1:

FSM Design Procedure

- **Next-State Function f_{NS}**
  - where $S_{n+1}=f_{NS}(X,S_n)$ by KMAPs or COMBINATORY logic synthesis

ACTUAL STATES				INPUT	NEXT STATES				OUTPUT
STATE NAME	$S_n(2)$	$S_n(1)$	$S_n(0)$	X	STATE NAME	$S_{n+1}(2)$	$S_{n+1}(1)$	$S_{n+1}(0)$	Y
S0	0	0	0	0	S1	0	0	1	0
S0	0	0	0	1	S2	0	1	0	0
S1	0	0	1	0	S3	0	1	1	1
S1	0	0	1	1	S2	0	1	0	0
S2	0	1	0	0	S1	0	0	1	0
S2	0	1	0	1	S4	1	0	0	1
S3	0	1	1	0	S3	0	1	1	1
S3	0	1	1	1	S2	0	1	0	0
S4	1	0	0	0	S1	0	0	1	0
S4	1	0	0	1	S4	1	0	0	1

Sn+1(0)		Sn(2) Sn(1)			
		00	01	11	10
Sn(0) X	00	1	1	x	1
	01	0	0	x	0
	11	0	0	x	x
	10	1	1	x	x

$$S_{n+1}(0) = \overline{X}$$

Finite-State-Machines

Exercise

Exercise 1:

FSM Design Procedure

- **Outputs Function f_{OUT}**
  - where $Y=f_{OUT}(S_n)$ by KMAPs or COMBINATORY logic synthesis

STATES				OUTPUT
STATE NAME	**$S_n(2)$**	**$S_n(1)$**	**$S_n(0)$**	**Y**
S0	0	0	0	0
S1	0	0	1	0
S2	0	1	0	0
S3	0	1	1	1
S4	1	0	0	1
SX	1	0	1	X
SX	1	1	0	X
SX	1	1	1	X

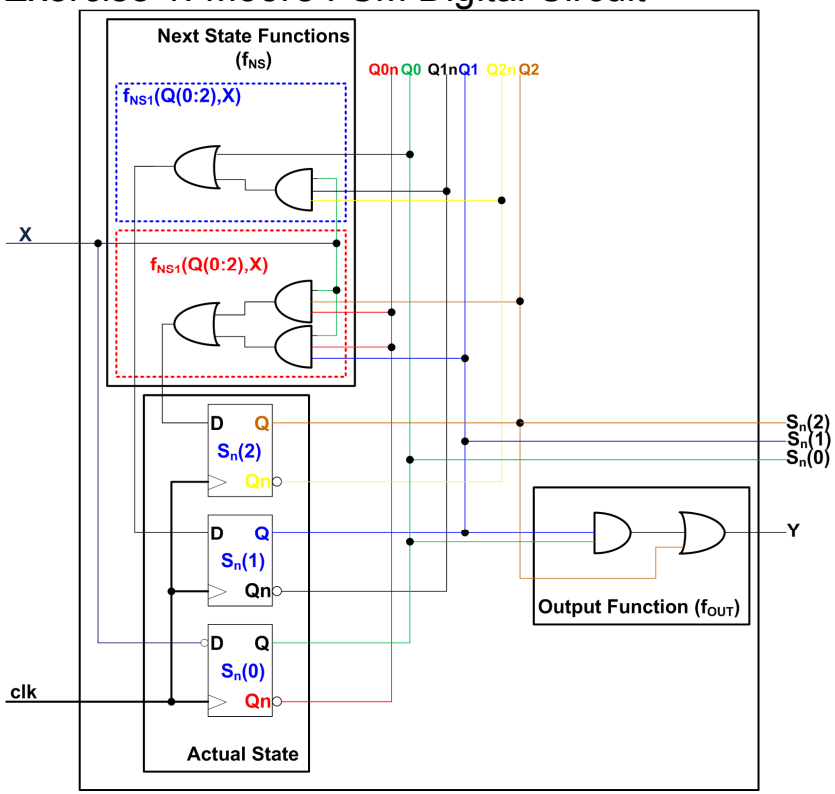
Y		**Sn(2) Sn(1)**			
		00	**01**	**11**	**10**
Sn(0)	**0**	0	0	x	1
	1	0	1	x	x

$$Y = S_n(2) + S_n(1) \cdot S_n(0)$$

Finite-State-Machines

Exercise

Exercise 1: Moore FSM Digital Circuit



Actual State:
$S_n(i)=Q(i)$
$(i=0,1,2)$

Next State:
$S_{n+1}(i)=D(i)$

Next-State Functions:
$D(i) = f_{NSi}(Q(0:2) , X)$

$$D(2) = Q(1) \cdot \overline{Q(0)} \cdot X + Q(2) \cdot \overline{Q(0)} \cdot X$$
$$D(1) = Q(0) + \overline{Q(2)} \cdot \overline{Q(1)} \cdot X$$
$$D(0) = \overline{X}$$

Output Function ($f_{OUT}(S_n)$):
$$Y = Q(2) + Q(1) \cdot Q(0)$$

Finite-State-Machines

Outline

- Introduction
- Mealy and Moore FSM
- State Diagram
- State Table
- Sequential Synchronous FSM Circuital Synthesis
- Exercise 1
- Exercise 2
- Exercise 3