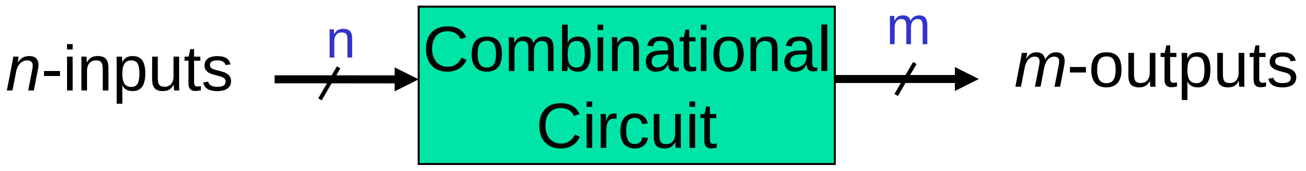
Sequential Circuits: Basic Concept

Overview

- Sequential Circuits
  - Sequential vs. Combinational Circuits
- Types of Sequential Circuits
  - Synchronous vs. Asynchronous Circuits
- Sequential Circuit Representations
  - Boolean Equations
    - Characteristic (Flip-Flop Input) Equations
    - Primary Output Equations
  - State Table
  - State Diagram
- Finite State Machines
  - Mealy vs. Moore Machines

Combinational vs. Sequential Circuits

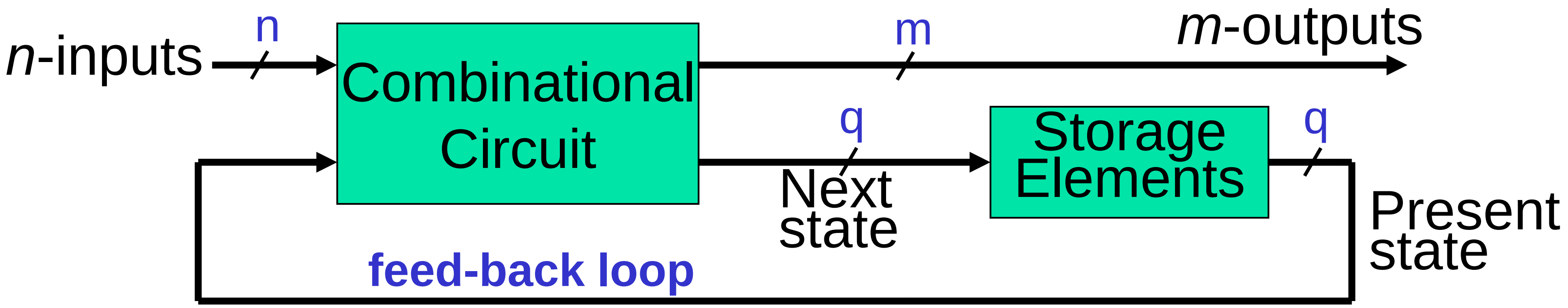
- Combinational circuits are memory-less and do not have feed-back loops. Thus, the output values depend ONLY on the current input values.



- Able to perform useful operations (add, subtract, multiply, encode, decode, multiplex, demultiplex, etc…).
- However, the performance of useful sequences of operations using only combinational circuits requires cascading many structures together.
- Thus, the hardware to do this is very costly and inflexible.
- To perform useful or flexible sequences of operations, we need to construct circuits that can store information between the operations.
- Such circuits are called Sequential Circuits.

Sequential Circuits

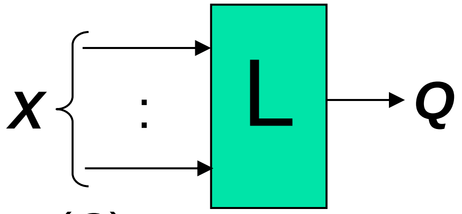
- Sequential circuits consist of combinational logic as well as memory elements (used to store certain circuit states). Outputs depend on BOTH current input values and previous input values (kept in the storage elements).



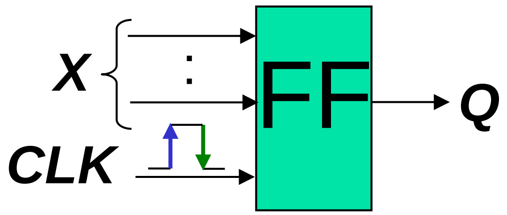
- There is a **feed-back loop** in the sequential circuits.
- The storage elements are circuits that are capable of storing binary information.
- One storage element can store one bit of information.

Storage Elements

- Two types of storage elements are used in Sequential Circuits: Latches and Flip-Flops.
- **L**atches (D, SR,JK, T)
  - General description of a latch:
    - 1-bit storage device with several inputs (***X***) and an output (***Q***).
    - Output is changed ***Q = f(X)*** only when specific combinations occur at the inputs ***X;*** otherwise the output remains unchanged (storage mode).



- **F**lip-**F**lops (D, SR,JK, T)
  - General description of a Flip-Flop:
    - 1-bit storage device with several inputs (***X***), an output (***Q***), and a specific ***trigger*** input (***CLK***).
    - Output is changed ***Q = f(X)*** on response of a pulse at the trigger input ***CLK*** (on the rising or falling edge of the pulse). When a pulse is absent at input CLK the output remains unchanged (storage mode).

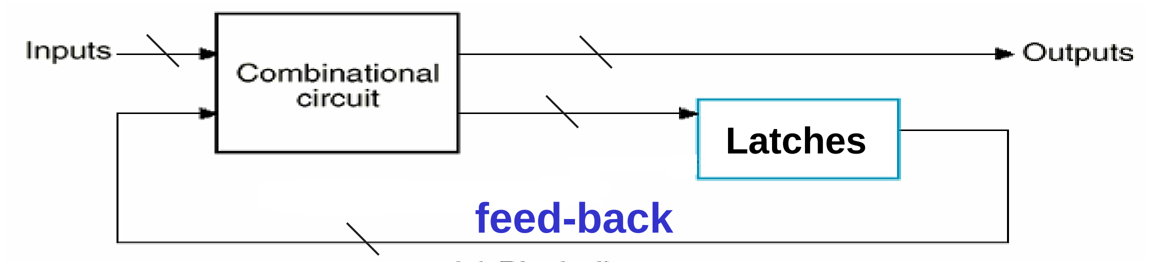


- More details about Latches and Flip-Flops later!

Types of Sequential Circuits

- Sequential circuits are divided into two main types depending on the storage elements that are used in the feed-back loop.
- Synchronous Sequential Circuits
  - Circuits with Flip-Flops in the feedback loop
- Asynchronous Sequential Circuits
  - Circuits without storage elements in the feed-back loop OR
  - Circuits with Latches in the feed-back loop
- We will not study asynchronous circuits
  - Only brief description and comparison will follow.

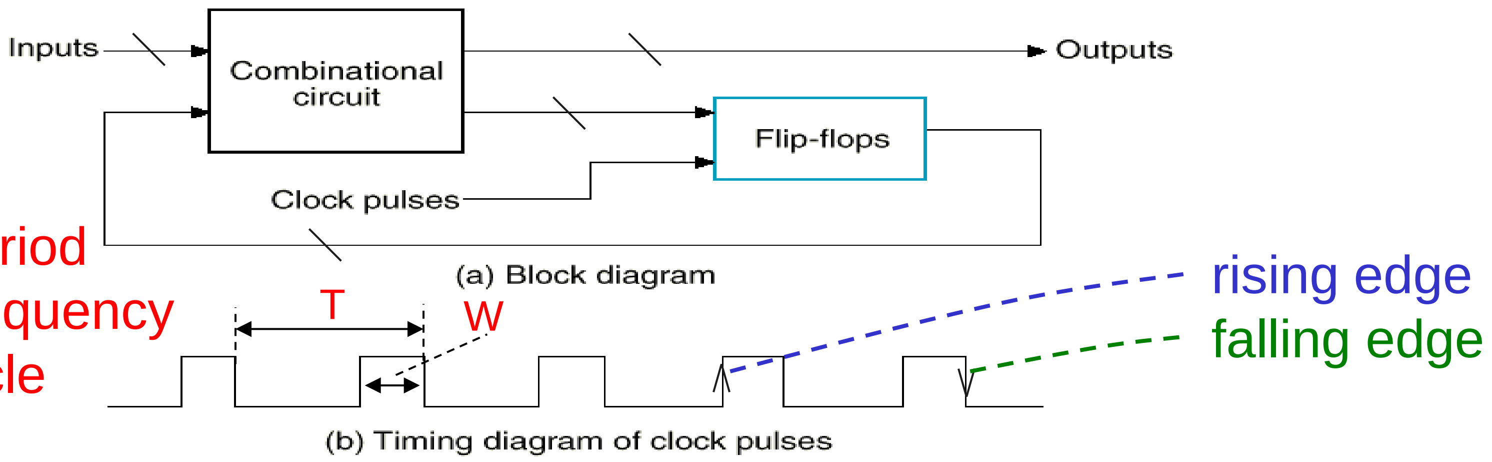
Asynchronous Sequential Circuits



(a) Block diagram

- The behavior is dependent on the order of input signal changes over continuous time, and output can change at **any** time (clockless).
- The Latches receive their inputs from the combinational circuit.
- The Latches can change state as soon as their inputs are changed.
  - Thus, a transition from one state to the other may occur at **any** time.
  - This makes the design and analysis of the circuit very difficult.
- If a circuit is not designed properly its behavior may become unpredictable:
  - The new state may change the Latches again through the feed-back loop.
  - This may lead to a succession of changes of state instead of a single change even if we do not change the inputs of the circuit:
    - The circuit comes to a stable state after several state transitions.
    - The transition from one state to another may never stop.

Synchronous Sequential Circuits



T – clock period
1/T – clock frequency
W/T – duty cycle

- The behavior can be defined from knowledge of its signals at discrete instants of time.
  - Circuit achieves synchronization by using a timing signal called the ***clock***
- The Flip-Flops receive their inputs from the combinational circuit.
- The Flip-Flops can change state only in response to a clock pulse on rising or falling edge.
- When a clock pulse is absent the Flip-Flop outputs cannot change even if the combinational circuit driving the flip-flop inputs changes in value.
- Thus, a transition from one state to the other occurs only at fixed time intervals dictated by the clock pulse, giving synchronous operation.

Synchronous vs. Asynchronous Sequential Circuits

- Storage Elements:
  - In synchronous sequential circuits - FLIP-FLOPs.
  - In asynchronous sequential circuits - either Latches or gate circuits with feedback producing the effect of latch operation.
- State Changes:
  - In a synchronous sequential circuit a change of state occurs only in response to a synchronizing clock pulse. All the FLIP-FLOPs are clocked simultaneously by a common clock pulse.
  - In an asynchronous sequential circuit, the state of the circuit can change immediately when an input change occurs. It does not use a clock.
- Circuit Speed:
  - In synchronous sequential circuits, the speed of operation depends on the maximum allowed clock frequency
    - because states are changed only on response of a clock pulse.
  - Asynchronous sequential circuits do not require clock pulses and they can change state with the input change.
    - Therefore, in general the asynchronous sequential circuits are faster than the synchronous sequential circuits.

Synchronous vs. Asynchronous Sequential Circuits (cont.)

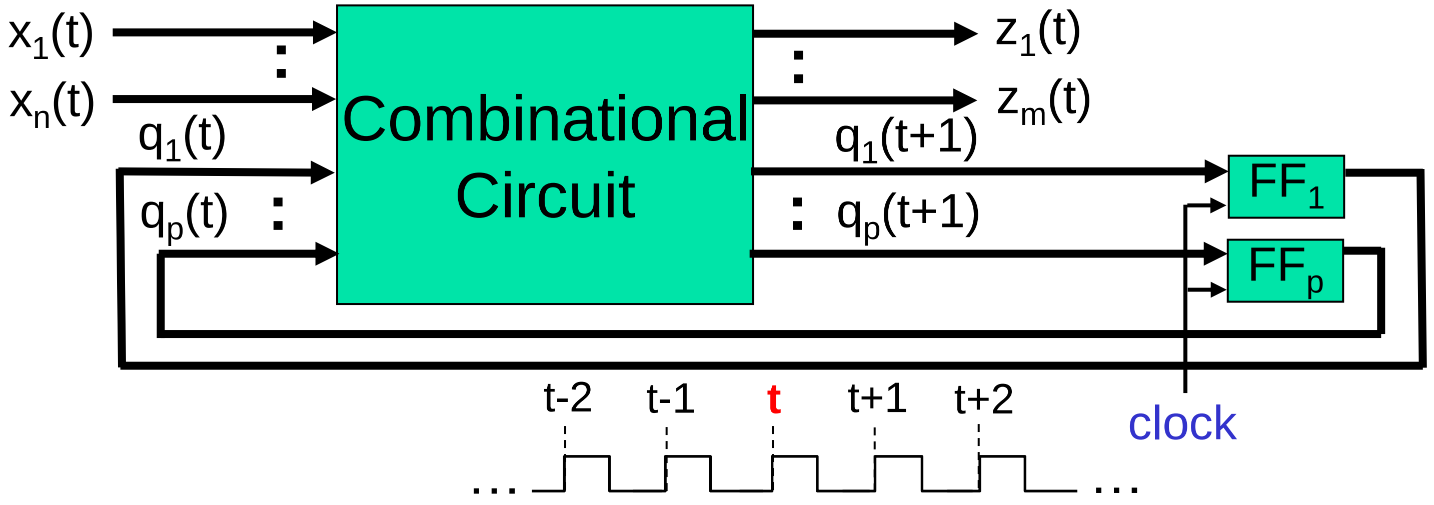
- Input Changes:
  - In synchronous sequential circuits input changes are assumed to occur between clock pulses. The circuit must be in the stable state before next clock pulse arrives.
  - In asynchronous sequential circuits input changes should occur only when the circuit is in a stable state.
- Number of Inputs that can Change:
  - In synchronous sequential circuits, any number of inputs can change simultaneously (during the absence of the clock).
  - In asynchronous sequential circuits only one input is allowed to change at a time. If more than one inputs change simultaneously, the circuit makes erroneous state transitions due to different delay paths for each input variable.
- Output Changes
  - In synchronous sequential circuits outputs change at any time when inputs change or on response to a clock pulse.
  - In asynchronous sequential circuits outputs change at any time when inputs are changed.

Synchronous vs. Asynchronous Sequential Circuits (cont.)

- Nowadays, most of the sequential circuits are synchronous due to
  - their predictability and
  - relatively easy design and analysis.
- An asynchronous circuit is preferred over synchronous circuit when high speed of operations are required
  - asynchronous circuits respond immediately without having to wait for a clock pulse.
- Asynchronous sequential circuits cost less in terms of number of gates than the synchronous circuits
  - therefore, for economical reasons, they find useful applications.

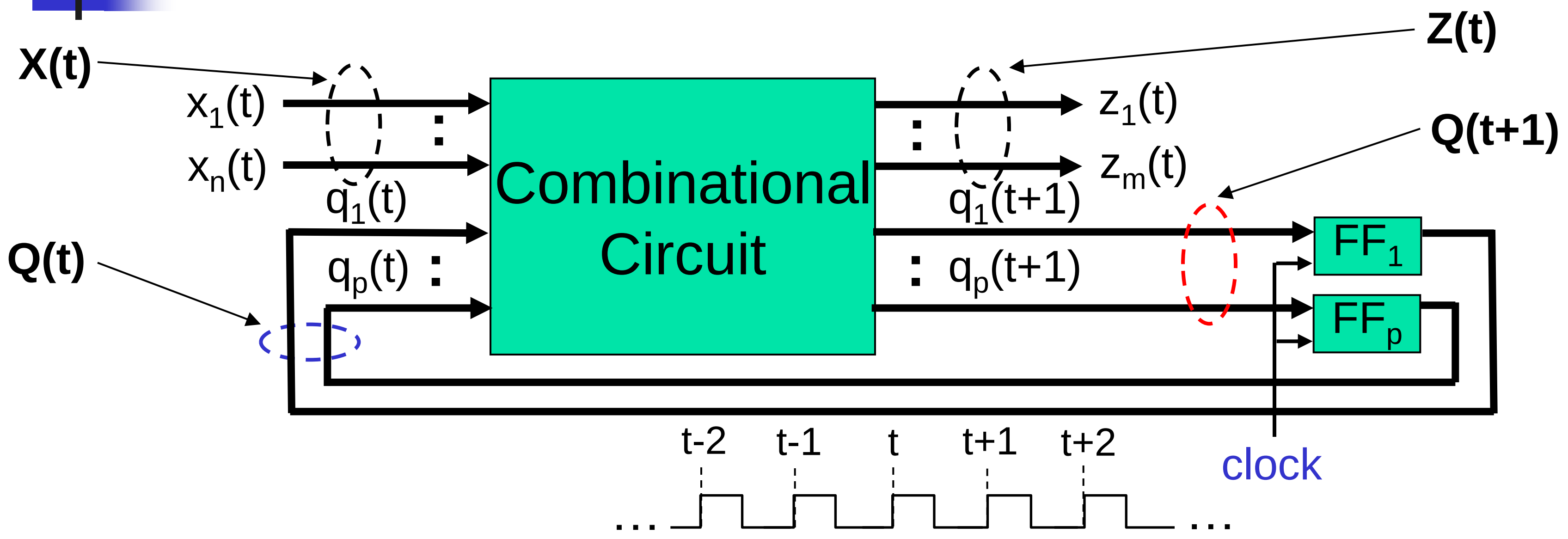
Synchronous Sequential Circuits: Definitions and Notations

- Consider the sequential circuit below which has ***n*** inputs, ***m*** outputs, and ***p*** Flip-Flops (storage elements):



- x_j(t) – the present value of input x_j ,i.e., the value prior to the application of a clock pulse at time t.
- z_j(t) – the present value of output z_j ,i.e., the value prior to the application of a clock pulse at time t.
- q_j(t) – the present value stored by FF_j ,i.e., the value prior to the application of a clock pulse at time t.
- q_j(t+1) – the next value to be stored by FF_j ,i.e., the value stored after the application of a clock pulse at time t.

Synchronous Sequential Circuits: Definitions and Notations (cont.)

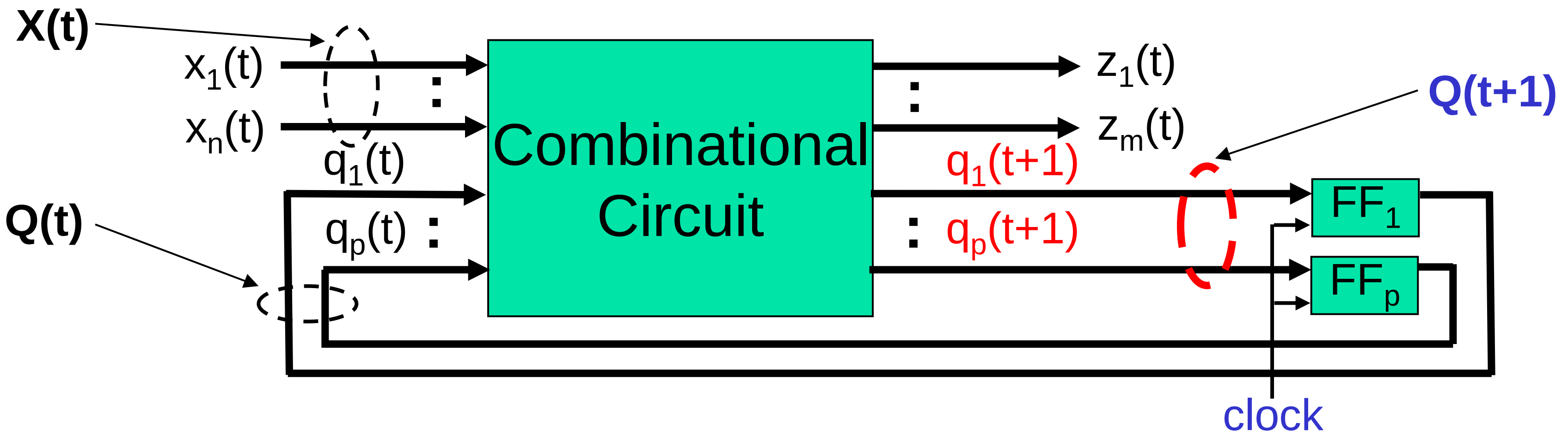


- **Input Vector X(t)** = ($x_1(t),\ldots,x_n(t)$) – the values at the inputs prior to the application of a clock pulse at time t.
- **Output Vector Z(t)** = ($z_1(t),\ldots,z_m(t)$) – the values at the outputs prior to the application of a clock pulse at time t.
- **Present State Q(t)** = ($q_1(t),\ldots,q_p(t)$) – the values stored by the Flip-Flops prior to the application of a clock pulse at time t.
- **Next State Q(t+1)** = ($q_1(t+1),\ldots,q_p(t+1)$) – the values stored by the Flip-Flops after the application of a clock pulse at time t. In other words, the state one clock period later.

Sequential Circuit Representations

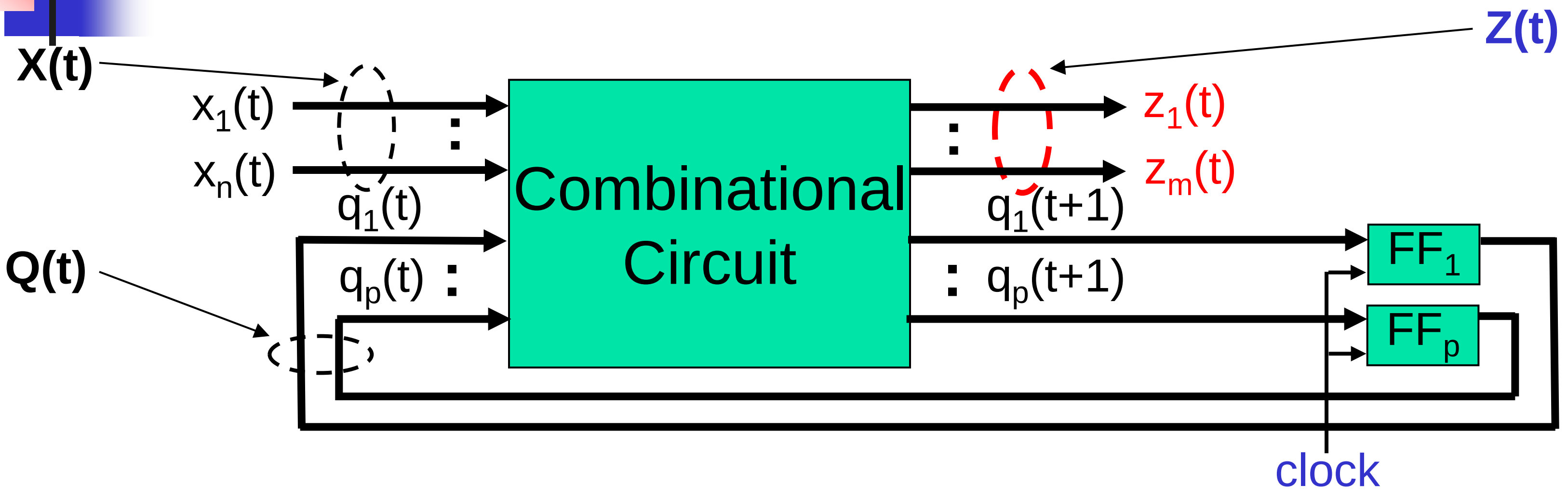
- Boolean Equations
  - Characteristic (Flip-Flop Input) Equations
  - Primary Output Equations
- State Table
- State Diagram

Characteristic (FF Input) Equations



- An algebraic representation used to fully specify the combinational logic that drives the inputs of the FFs.
- Set of Boolean Equations:
  $q_1(t+1) = f_1(\ x_1(t),\ldots,x_n(t),q_1(t),\ldots,q_p(t)\ )$
  $\vdots$
  $q_p(t+1) = f_p(\ x_1(t),\ldots,x_n(t),q_1(t),\ldots,q_p(t)\ )$
- Short Notation: $Q(t+1) = F(\ X(t),\ Q(t)\ )$
- The next state Q(t+1) depends on the present state Q(t) and the input vector X(t).

Primary Output Equations



- An algebraic representation used to fully specify the combinational logic that defines the outputs of the circuit.
- Set of Boolean Equations:

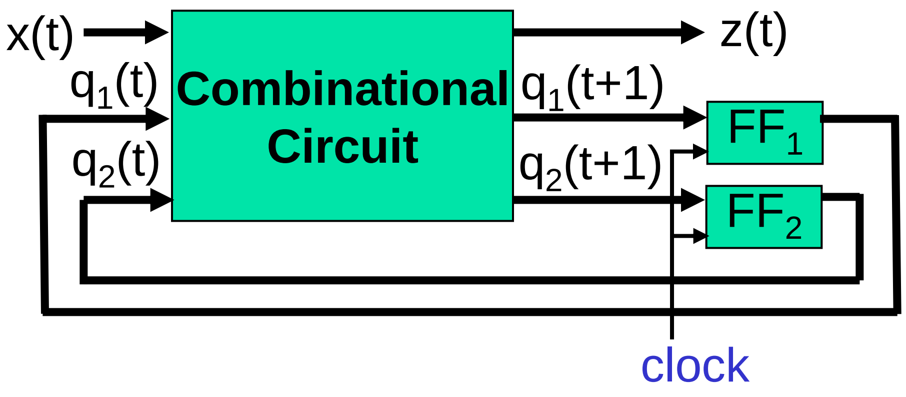
  $z_1(t) = g_1(x_1(t),\dots,x_n(t),q_1(t),\dots,q_p(t))$

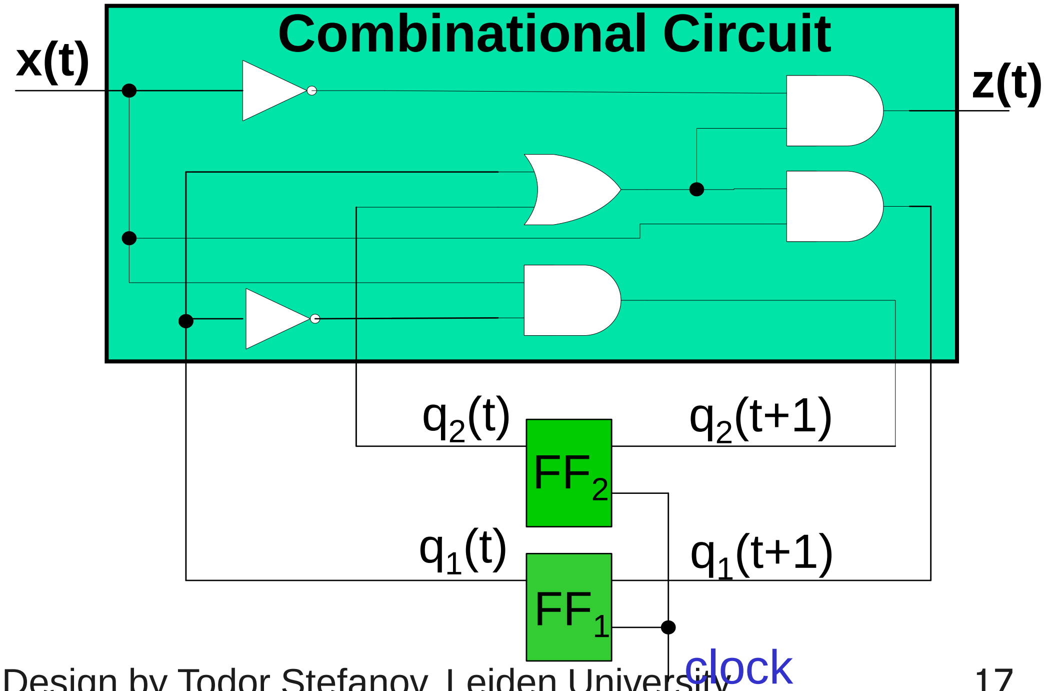
  $\vdots$

  $z_m(t) = g_m(x_1(t),\dots,x_n(t),q_1(t),\dots,q_p(t))$
- Short Notation: $Z(t) = G(X(t), Q(t))$
- The output vector Z(t) depends on the present state Q(t) and the input vector X(t).

Example: Equations

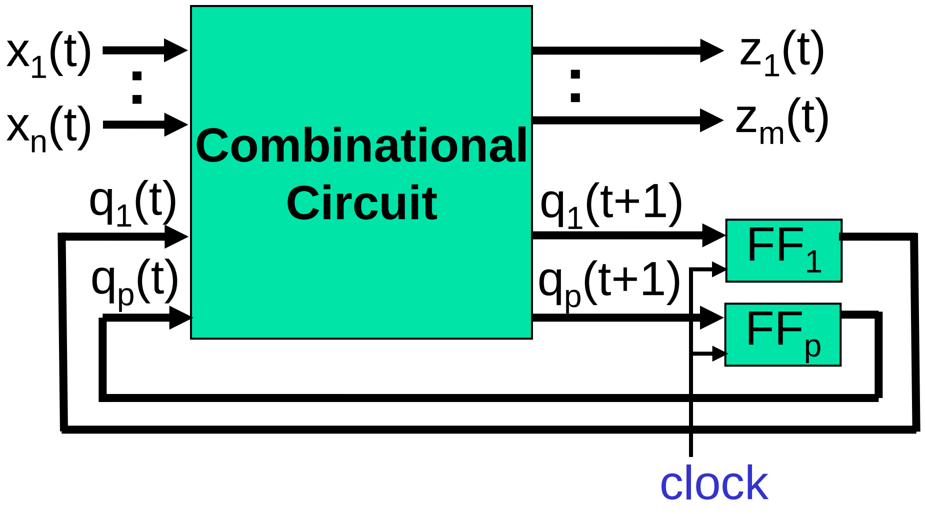
- Consider simple sequential circuit:
  - 1 input , 1 output , 2 Flip-Flops
- Characteristic (FF input) equations:
  - $q_1(t+1) = f_1(\, x(t), q_1(t), q_2(t)\,)$
    $= q_1(t) \bullet x(t) + q_2(t) \bullet x(t)$
    $= (\, q_1(t) + q_2(t)\,) \bullet x(t)$
  - $q_2(t+1) = f_2(\, x(t), q_1(t), q_2(t)\,)$
    $= q_1(t)' \bullet x(t)$
- Primary output equations:
  - $z(t) = g(\, x(t), q_1(t), q_2(t)\,)$
    $= q_1(t) \bullet x(t)' + q_2(t) \bullet x(t)'$
    $= (\, q_1(t) + q_2(t)\,) \bullet x(t)'$





State Table

- Enumerates the relationship between inputs, outputs, and states of the sequential circuit.
- Given a circuit with ***n*** inputs, ***m*** outputs, and ***p*** Flip-Flops:
  - The corresponding state table contains ***2^{n+p}*** rows.
  - The corresponding state table contains ***n+2p+m*** columns.



Inputs			Present State			Next State			Outputs		
$x_1(t)$	…	$x_n(t)$	$q_1(t)$	…	$q_p(t)$	$q_1(t+1)$	…	$q_p(t+1)$	$z_1(t)$	…	$x_m(t)$
0	…	0	0	…	0						
0	…	0	0	…	1						
…											
1	…	1	1	…	0						
1	…	1	1	…	1						

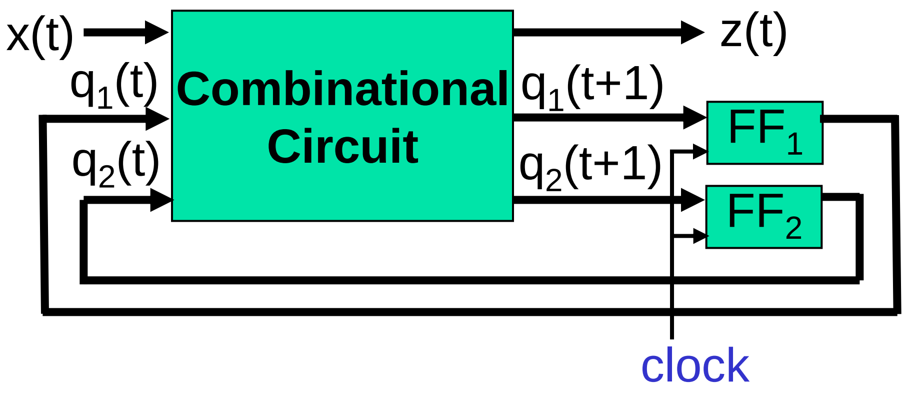
2^{n+p}

n+p+p+m

Example: State Table

- Consider previous example:
  - 1 input , 1 output , 2 Flip-Flops
- Equations:

$q_1(t+1) = x(t)\bullet q_2(t) + x(t)\bullet q_1(t)$
$q_2(t+1) = x(t)\bullet q_1(t)'$
$z(t) = x(t)'\bullet q_2(t) + x(t)'\bullet q_1(t)$



- State Table
  - $2^2 = 4$ states
    Q1 = 00
    Q2 = 01
    Q3 = 10
    Q4 = 11
  - How do you read this table?

Inputs	Present State		Next State		Outputs
$x(t)$	$q_1(t)$	$q_2(t)$	$q_1(t+1)$	$q_2(t+1)$	$z(t)$
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	0	1
0	1	1	0	0	1
1	0	0	0	1	0
1	0	1	1	1	0
1	1	0	1	0	0
1	1	1	1	0	0

State Table to/from Equations

- <u>State Table to Equations</u>
  - Treat the State Table as a Truth Table of Boolean functions to get the Characteristic and Primary Output Equations.
  - You can derive the CSOP or CPOS form of the Equations directly using the right part of the State Table.
  - You can simplify the Equations using algebraic manipulations or K-maps.

right part of the table

Inputs			Present State				Next State			Outputs		
$x_1(t)$	…	$x_n(t)$	$q_1(t)$	…	$q_p(t)$		$q_1(t+1)$	…	$q_p(t+1)$	$z_1(t)$	…	$x_m(t)$
0	…	0	0	…	0							
0	…	0	0	…	1							
						…						
1	…	1	1	…	0							
1	…	1	1	…	1							

- <u>State Table from Equations</u>
  - The Characteristic and Primary Output Equations are Boolean functions.
  - Represent the Equations in a common Truth Table. This is actually the State Table.

Example: State Table to Equations

Inputs	Present State		Next State		Outputs
$x(t)$	$q_1(t)$	$q_2(t)$	$q_1(t+1)$	$q_2(t+1)$	$z(t)$
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	0	1
0	1	1	0	0	1
1	0	0	0	1	0
1	0	1	1	1	0
1	1	0	1	0	0
1	1	1	1	0	0

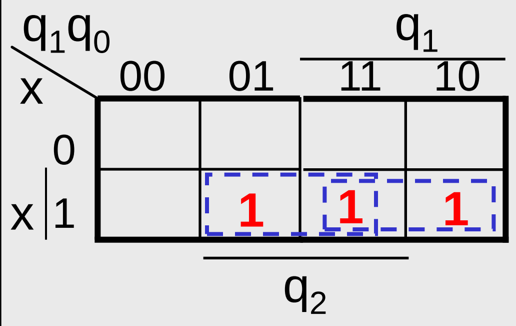
Equations in CSOP form

$$q_1(t+1) = x(t)\bullet q_1(t)'\bullet q_2(t) + x(t)\bullet q_1(t)\bullet q_2(t)' + x(t)\bullet q_1(t)\bullet q_2(t)$$

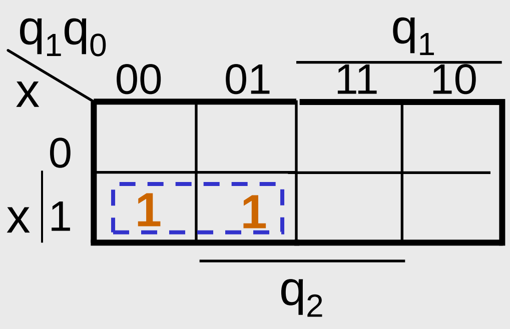
$$q_2(t+1) = x(t)\bullet q_1(t)'\bullet q_2(t)' + x(t)\bullet q_1(t)'\bullet q_2(t)$$

$$z(t) = x(t)'\bullet q_1(t)'\bullet q_2(t) + x(t)'\bullet q_1(t)\bullet q_2(t)' + x(t)'\bullet q_1(t)\bullet q_2(t)$$

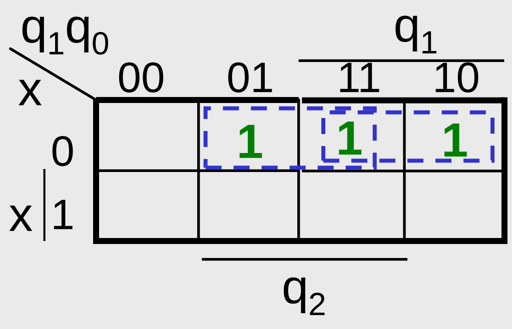
Simplified Equations



$$q_1(t+1) = x(t)\bullet q_2(t) + x(t)\bullet q_1(t)$$



$$q_2(t+1) = x(t)\bullet q_1(t)'$$



$$z(t) = x(t)'\bullet q_2(t) + x(t)'\bullet q_1(t)$$

Example: State Table from Equations

$q_1(t+1) = x(t) \bullet q_2(t) + x(t) \bullet q_1(t)$
$q_2(t+1) = x(t) \bullet q_1(t)'$
$z(t) = x(t)' \bullet q_2(t) + x(t)' \bullet q_1(t)$

Follow the procedure described below

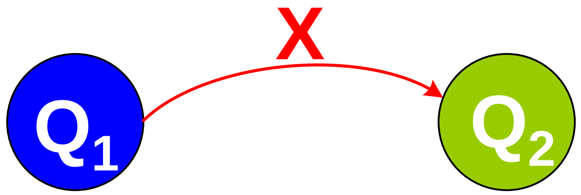
Inputs	Present State		Next State		Outputs
$x(t)$	$q_1(t)$	$q_2(t)$	$q_1(t+1)$	$q_2(t+1)$	$z(t)$
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	0	1
0	1	1	0	0	1
1	0	0	0	1	0
1	0	1	1	1	0
1	1	0	1	0	0
1	1	1	1	0	0

- Identify the variables that are used on the right-hand side of the equations
  - $x(t)$, $q_1(t)$, and $q_2(t)$ $\rightarrow$ 3 variables
- List all possible combinations of the above variables:
  - $2^3 = 8$ combinations
- For each combination evaluate the Equations:
  - Find the value of $q_1(t+1)$, $q_2(t+1)$, and $z(t)$

State Diagram

- Graph representation of a state table.

Inputs			Present State			Next State			Outputs		
$x_1(t)$	…	$x_n(t)$	$q_1(t)$	…	$q_p(t)$	$q_1(t+1)$	…	$q_p(t+1)$	$z_1(t)$	…	$x_m(t)$
0	…	0	0	…	0						
0	…	0	0	…	1						
1	…	1	1	…	0						
1	…	1	1	…	1						

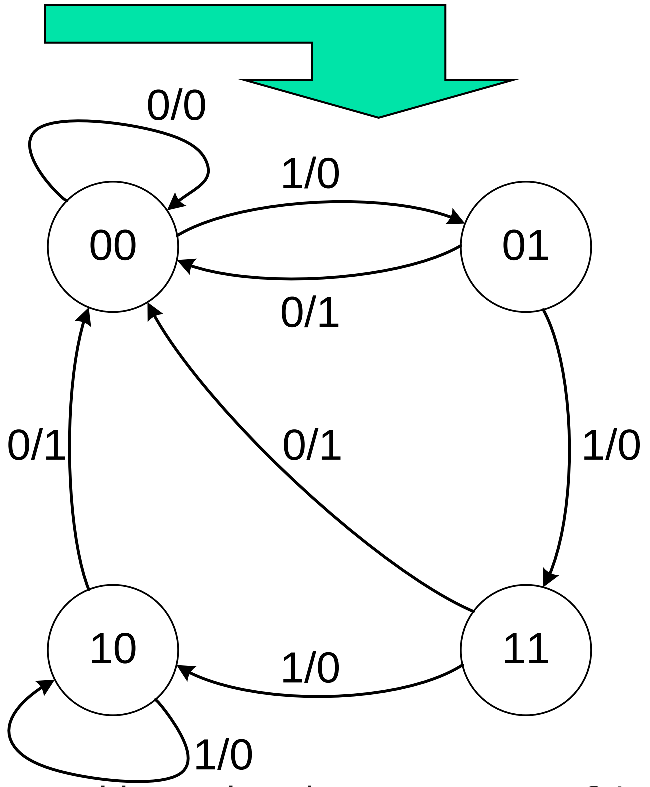


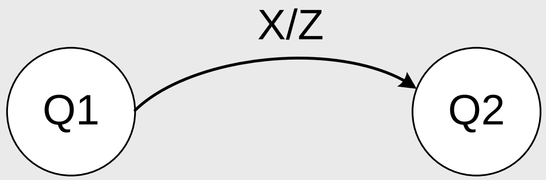
- Graph ***node*** with label ***Q*** denotes state ***Q***.
- Graph ***edge*** with label ***X*** denotes transition between two states when input ***X*** is applied.
- The graph has ***2^p*** ***nodes*** and ***2^{n+p}*** ***edges***. Why?

Example: State Diagram

Inputs	Present State			Next State		Outputs
x(t)	q_1(t)	q_2(t)		q_1(t+1)	q_2(t+1)	z(t)
0	0	0		0	0	0
0	0	1		0	0	1
0	1	0		0	0	1
0	1	1		0	0	1
1	0	0		0	1	0
1	0	1		1	1	0
1	1	0		1	0	0
1	1	1		1	0	0

- Possible states = { 00, 01, 10, 11 }
  → 4 nodes in the diagram
- Possible Transitions = #rows in table
  → 8 edges in the diagram



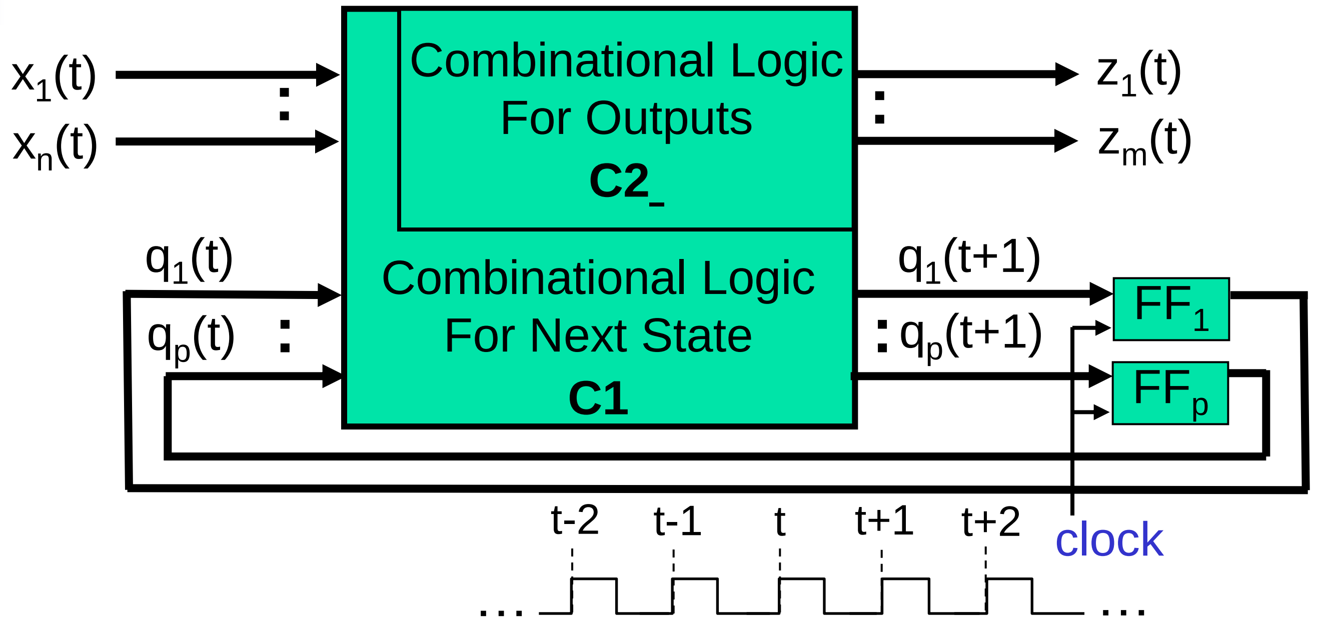


Reads as:
When at state ***Q1*** and apply vector ***X***,
we get output ***Z*** and proceed to state ***Q2.***

Finite State Machines

- The synchronous sequential circuits also can be called Finite State Machines (FSM). Why?
- A sequential circuit with p Flip-Flops
  - can have ***2^p*** distinct states.
  - can have ***2^p x 2^p = 2^{2p}*** distinct transitions.
- So, it has finite number of distinct states and transitions, hence the name.
- FSM is also one of the so called Models of Computation.
- There are two types of FSM:
  - Mealy model FSM
  - Moore model FSM

Mealy Finite State Machine



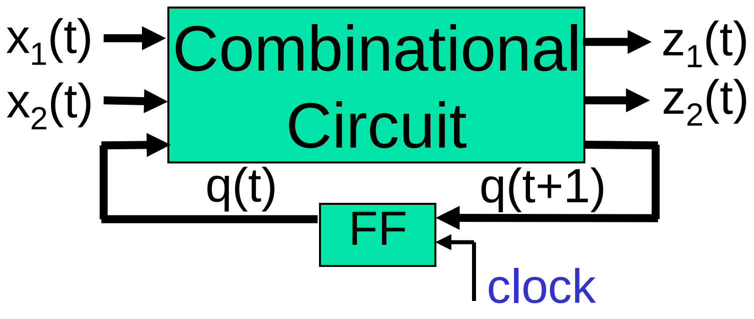
- Both outputs and next state depend directly on both primary inputs AND present state.

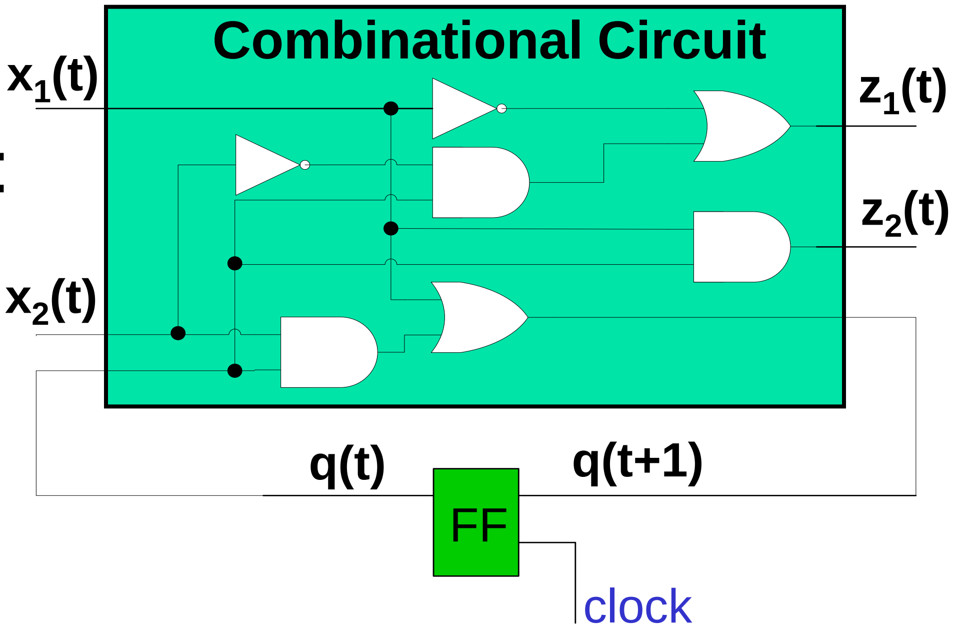
$$Q(t+1) = C1(\ X(t), Q(t)\ )$$
$$Z(t) \qquad = C2(\ X(t), Q(t)\ )$$

- Outputs are asynchronous, i.e., they can change in response to any changes in the inputs, independent of the clock.

Example: Mealy Machine

- Consider simple sequential circuit:
  - 2 input , 2 output , 1 Flip-Flop
- Characteristic (FF input) equations:
  - $q(t+1) = x_1(t) + x_2(t) \bullet q(t)$
- Primary output equations:
  - $z_1(t) = x_1(t)' + x_2(t)' \bullet q(t)$
  - $z_2(t) = x_1(t) \bullet q(t)$

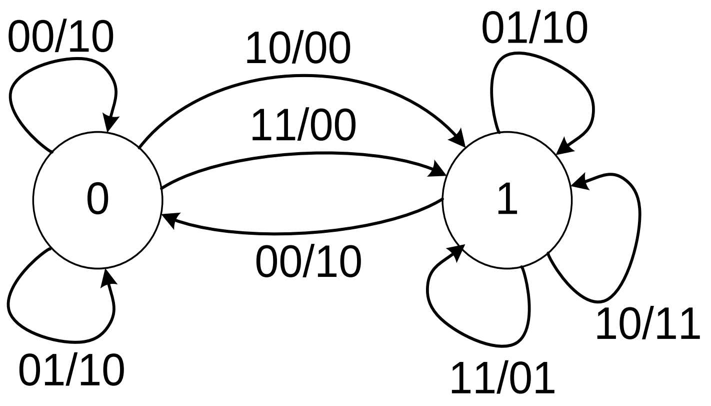


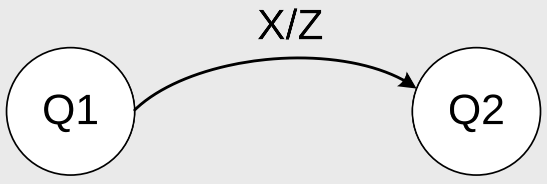


Example: Mealy Machine (cont.)

- Possible states = { 0,1 }, thus 2 nodes in state diagram
- Possible Transitions = #rows in the table, thus 8 edges in state diagram

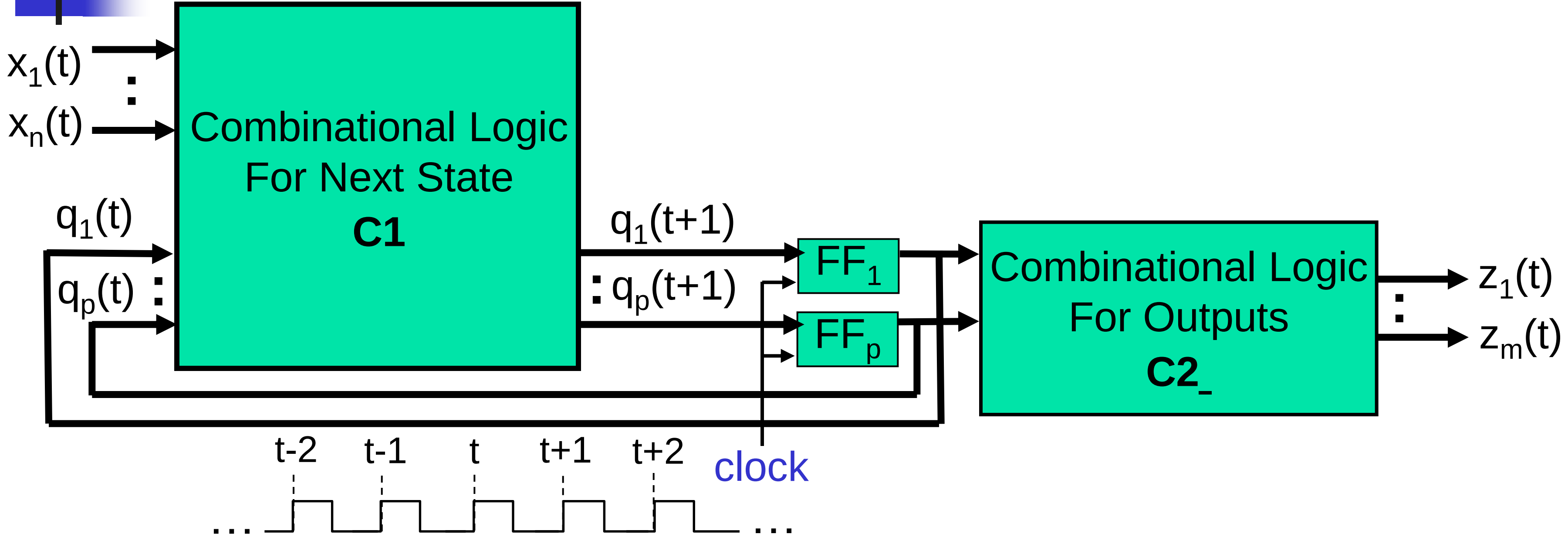
Inputs		PS	NS	Outputs	
$x_1(t)$	$x_2(t)$	$q(t)$	$q(t+1)$	$z_1(t)$	$z_2(t)$
0	0	0	0	1	0
0	0	1	0	1	0
0	1	0	0	1	0
0	1	1	1	1	0
1	0	0	1	0	0
1	0	1	1	1	1
1	1	0	1	0	0
1	1	1	1	0	1





Reads as:
When at state ***Q1*** and apply vector ***X***, we get output ***Z*** and proceed to state ***Q2.***

Moore Finite State Machine



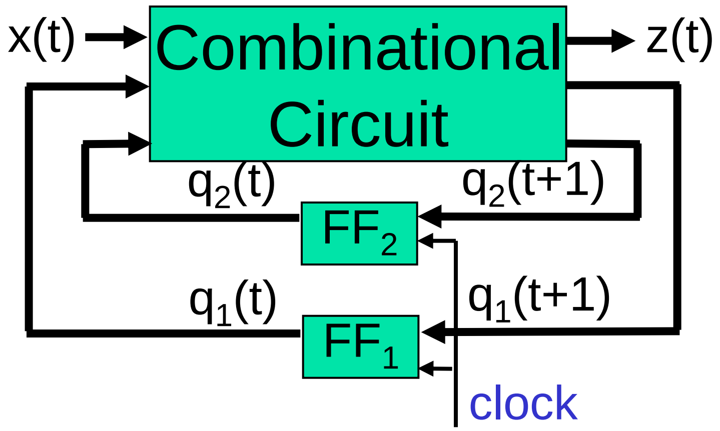
- Next state depends directly on both primary inputs AND present state. Outputs depend directly only on present state.

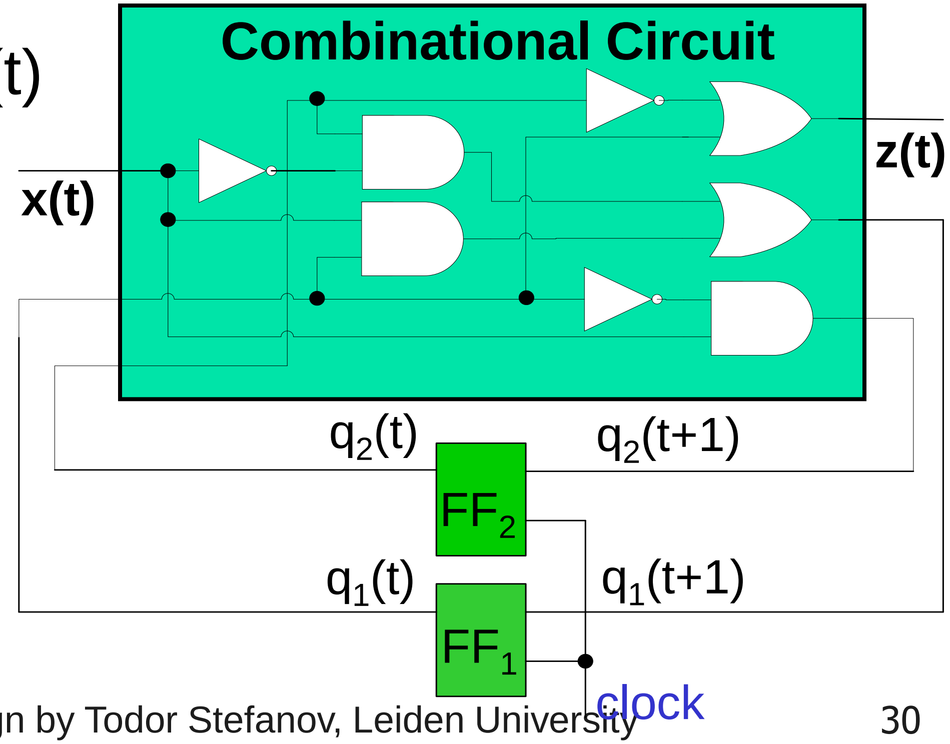
$$
\begin{aligned}
Q(t+1) &= C1(\, X(t), Q(t)\,) \\
Z(t) &= C2(\, Q(t)\,) = C2(\; C1(\, X(t-1), Q(t-1)\,)\;)
\end{aligned}
$$

- Moore outputs are synchronous with the clock, only changing with state transitions.

Example: Moore Machine

- Consider simple sequential circuit:
  - 1 input , 1 output , 2 Flip-Flops
- Characteristic (FF input) equations:
  - $q_1(t+1) = x(t) \bullet q_1(t) + x(t)' \bullet q_2(t)$
  - $q_2(t+1) = x(t) \bullet q_1(t)'$
- Primary output equations:
  - $z(t) = q_1(t) + q_2(t)'$

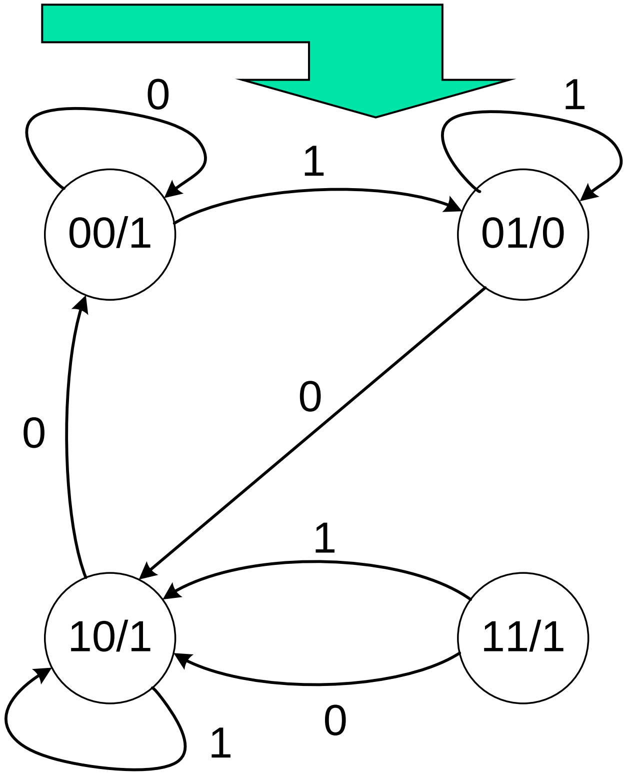


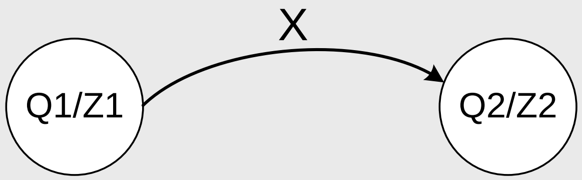


Example: Moore Machine (cont.)

Inputs	Present State		Next State		Outputs
$x(t)$	$q_1(t)$	$q_2(t)$	$q_1(t+1)$	$q_2(t+1)$	$z(t)$
0	0	0	0	0	1
0	0	1	1	0	0
0	1	0	0	0	1
0	1	1	1	0	1
1	0	0	0	1	1
1	0	1	0	1	0
1	1	0	1	0	1
1	1	1	1	0	1

- Possible states = { 00, 01, 10, 11 } $\rightarrow$ 4 nodes in the diagram
- Possible Transitions = #rows in table $\rightarrow$ 8 edges in the diagram

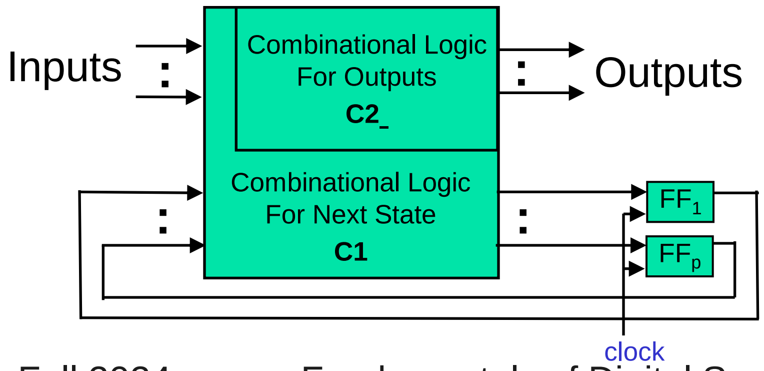


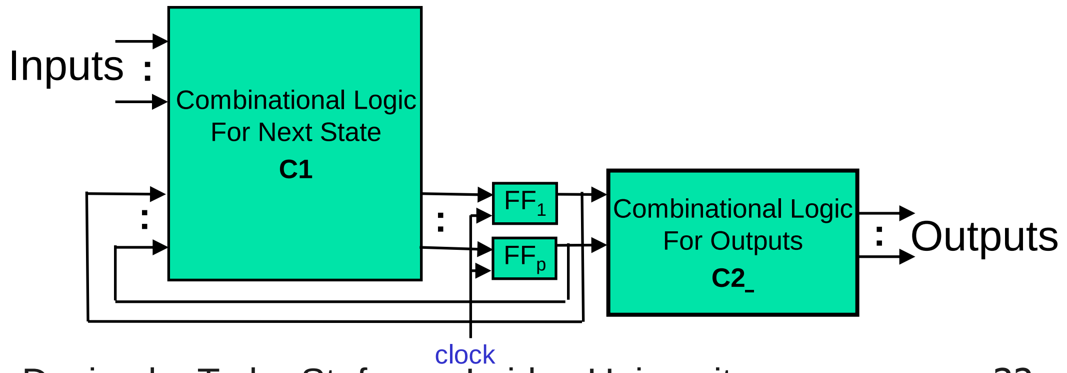


Reads as:
When at state ***Q1*** with output ***Z1*** and apply input ***X***, we proceed to state ***Q2*** with output ***Z2.***

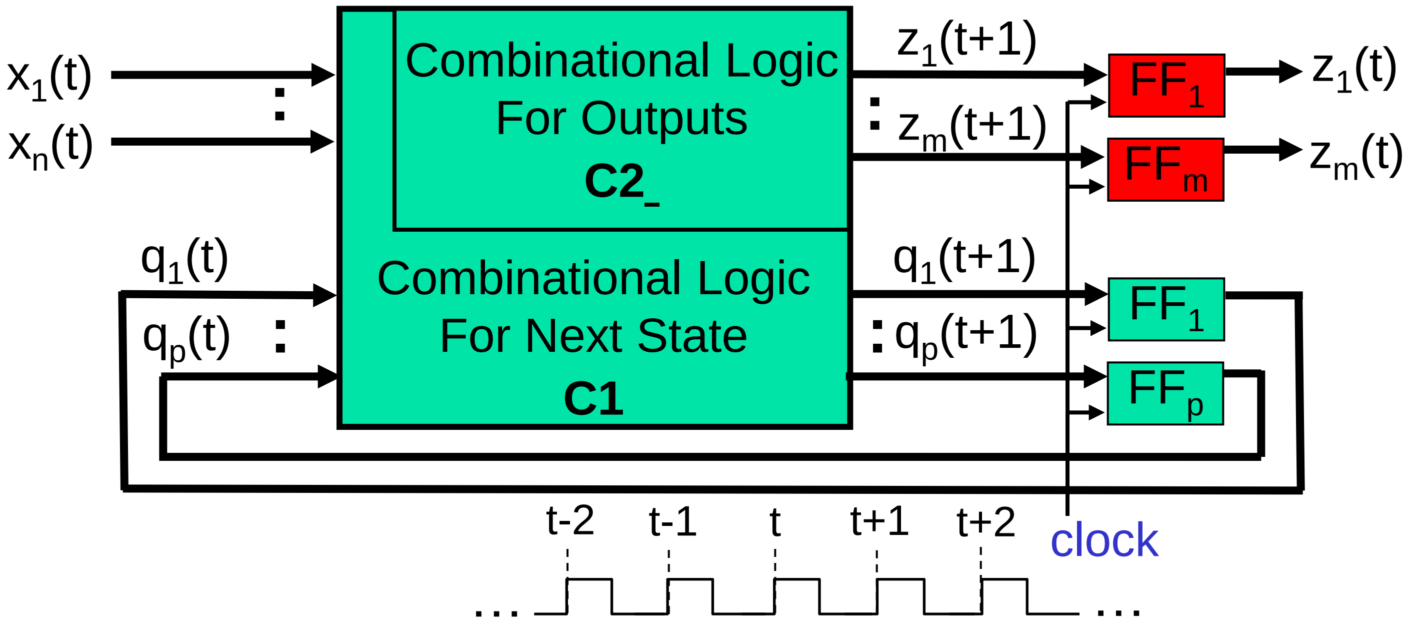
Comparison of ***Mealy*** and ***Moore*** Machines

- Mealy Machines tend to have less states
  - Different outputs on arcs rather than on states
- Mealy Machines react faster to inputs
  - React in same cycle – don't need to wait for clock.
  - In Moore machines, more logic may be necessary to decode state into outputs – more gate delays after.
- Moore Machines are safer to use
  - Outputs change at clock edge (always one cycle later).
  - In Mealy machines, input change can cause output change as soon as logic is done – a big problem when two machines are interconnected – asynchronous feedback may occur.


Inputs
Combinational Logic For Outputs C2_
Outputs
Combinational Logic For Next State C1
FF_1
FF_p
clock


Inputs
Combinational Logic For Next State C1
FF_1
FF_p
Combinational Logic For Outputs C2_
Outputs
clock

Registered ***Mealy*** Machine (Really Moore)



- Synchronous (or registered) Mealy Machine
  - Registered outputs
  - Avoids “glitchy” outputs
- Registered Mealy Machine is actually Moore Machine. Why?
  - View outputs as expanded state vector
- So, we get Moore Machine with no output decoding