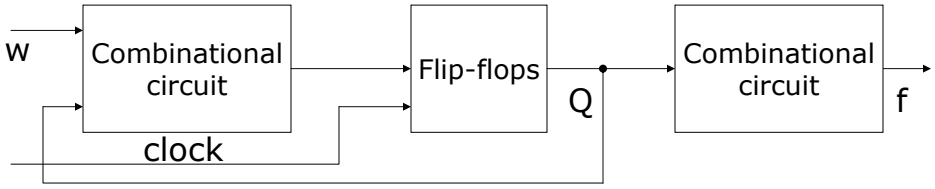
ECE380 Digital Logic

Synchronous Sequential Circuits:
State Diagrams, State Tables

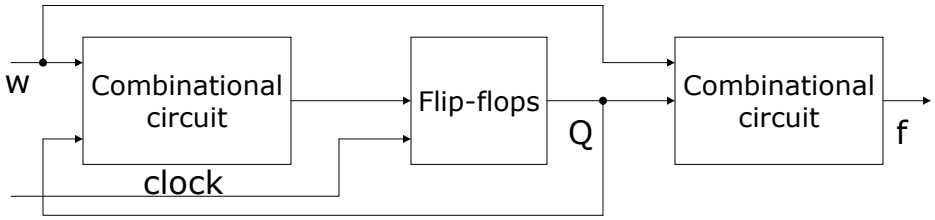
Synchronous sequential circuits

- Circuits where a clock signal is used to control operation are called ***synchronous sequential circuits***
  - The term ***active clock edge*** refers to the clock edge that causes a change in state (positive or negative)
- Realized using combinational logic and one or more flip-flops
- Two models for synchronous sequential circuits
  - **Moore model**: circuit outputs depend only on the present state of the circuit
  - **Mealy model**: circuit outputs depend on the present state of the circuit and the primary inputs
- Sequential circuits are also called ***finite state machines*** (**FSM**)

Moore versus Mealy machines



Moore state machine



Mealy state machine

Basic design steps

- We will introduce techniques for sequential circuit design via a simple example
- Design a circuit that meets the following specifications:
  - The circuit has one input, ***w***, and one output, ***z***
  - All changes in the circuit occur on the positive edge of the clock signal
  - Output ***z***=1 if the input ***w*** was 1 during the two immediately preceding clock cycles
- From this specification it is obvious that ***z*** cannot depend solely of the value of ***w***

Sequences of signals

- The example input and output sequence below aides in the description of the circuit

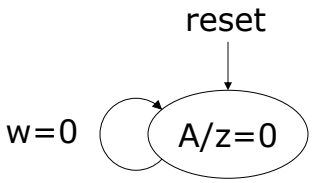
Clock cycle	t_0	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}
w	0	1	0	1	1	0	1	1	1	0	1
z	0	0	0	0	0	1	0	0	1	1	0

State diagram

- The first step in designing an FSM is determining how many states are needed and which transitions are possible from one state to another
  - No preset procedure for this
  - The designer must think about what the circuit is to accomplish
- A good beginning is to define a **reset** state that the circuit should enter when power is applied or when a reset signal is received

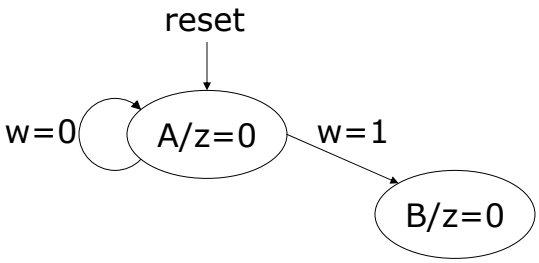
State diagram

- For our example, assume the starting state is called A
- As long as w=0, the circuit should do nothing and z=0



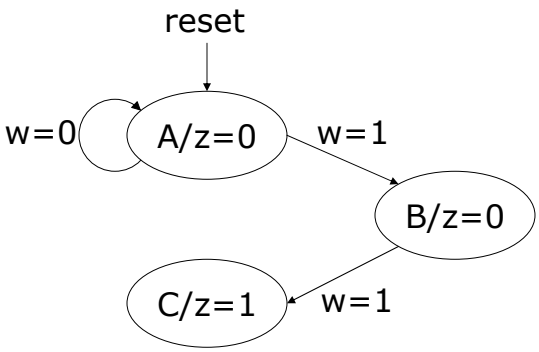
State diagram

- When w=1, the circuit should 'remember' this by transitioning to a new state (**B**)
- This transition should occur at the next positive edge of the clock signal

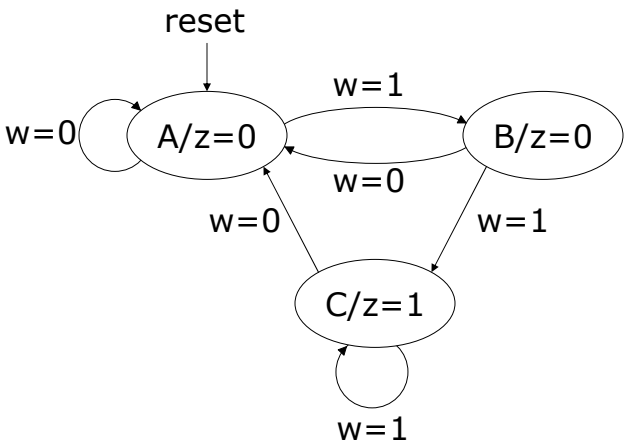


State diagram

- When in state **B** and w=1, the circuit should 'remember' this by transitioning to a new state (**C**)



Complete state diagram



Moore model state diagram

State table

- A state diagram describes circuit functionality, but does not describe circuit implementation
- Translation to a tabular form is necessary
- The ***state table*** should contain
  - All transitions from each **present state** to each **next state** for all valuations of the input signals
  - The output, ***z***, is specified with respect to the present state

Present state	Next state		Output z
	w=0	w=1	
A	A	B	0
B	A	C	0
C	A	C	1

State assignment

- The states are defined in terms of variables (**A**, **B**, and **C**)
- Each state is represented by a particular valuation of ***state variables***
- Each state variable is implemented with a flip-flop
- Since three states have to be realized, it is sufficient to use two state variables
  - Use y_2y_1 for the present state (present state variables)
  - Use Y_2Y_1 for the next state (next state variables)

State-assigned table

	Present state y_2y_1	Next state		Output z
		w=0	w=1	
		Y_2Y_1	Y_2Y_1	
A	00	00	01	0
B	01	00	10	0
C	10	00	10	1
	11	dd	dd	d

Note the addition of the y_2y_1=11 state. Although it is not used, it is needed for completeness.

Next-state and output maps

- K-maps are constructed from the state table for:
  - Circuit outputs (**z** in this case)
  - Inputs for the flip-flops (next-state K-maps)
- Constructing the next-state maps depends on the type of flip-flop (D, T, JK) used for the implementation
  - D is the most straightforward: next-state maps are constructed directly from the state table since
    - $Q(t+1)=Q^+=D$
  - T and JK implementations will be covered later

State table and next-state maps

	Present state y_2y_1	Next state w=0 Y_2Y_1	Next state w=1 Y_2Y_1	Output z
A	00	00	01	0
B	01	00	10	0
C	10	00	10	1
	11	dd	dd	d

w \ y_2y_1	00	01	11	10
0	0	0	d	0
1	1	0	d	0

$Y_1 = wy_1'y_2'$

w \ y_2y_1	00	01	11	10
0	0	0	d	0
1	0	1	d	1

$Y_2 = w(y_1 + y_2)$

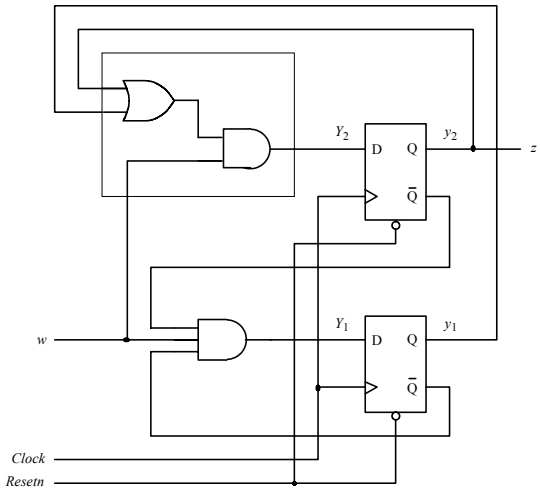
State table and output map

	Present state y_2y_1	Next state w=0 Y_2Y_1	Next state w=1 Y_2Y_1	Output z
A	00	00	01	0
B	01	00	10	0
C	10	00	10	1
	11	dd	dd	d

y_2 \ y_1	0	1
0	0	0
1	1	d

$z = y_2$

Circuit diagram



Timing diagram