Synchronous Sequential Circuits

Logic Circuits:

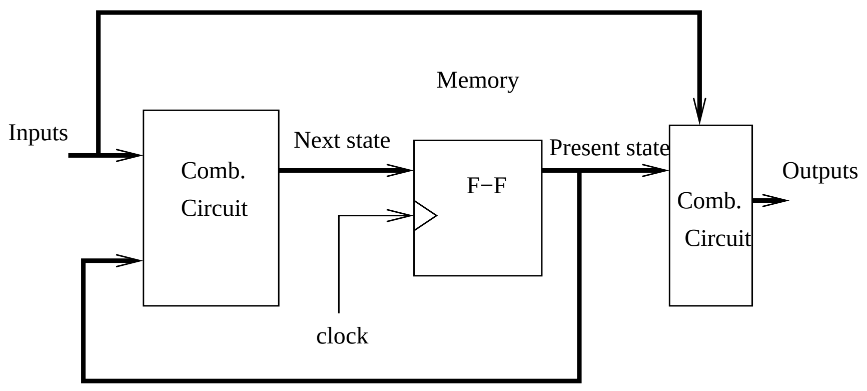
- **Combinational:**
  Multiplexers, Decoders,..
- **Sequential:**
  - **Synchronous:**
    1-Moore
    2-Mealy
  - **Asynchronous**

Structure of Sequential Circuits

Next state = f(present state, inputs)

Output = f(present state, inputs) for Mealy

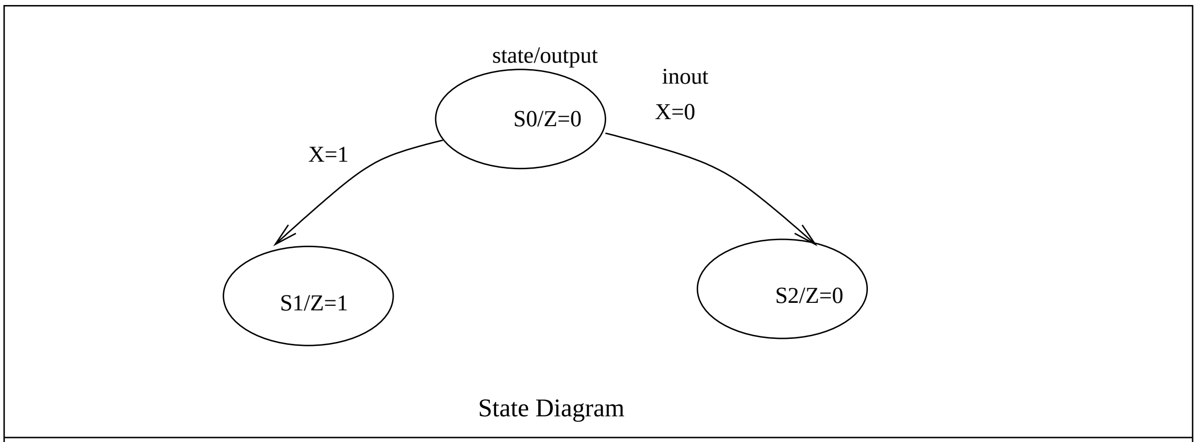
Output= f(present state) for Moore

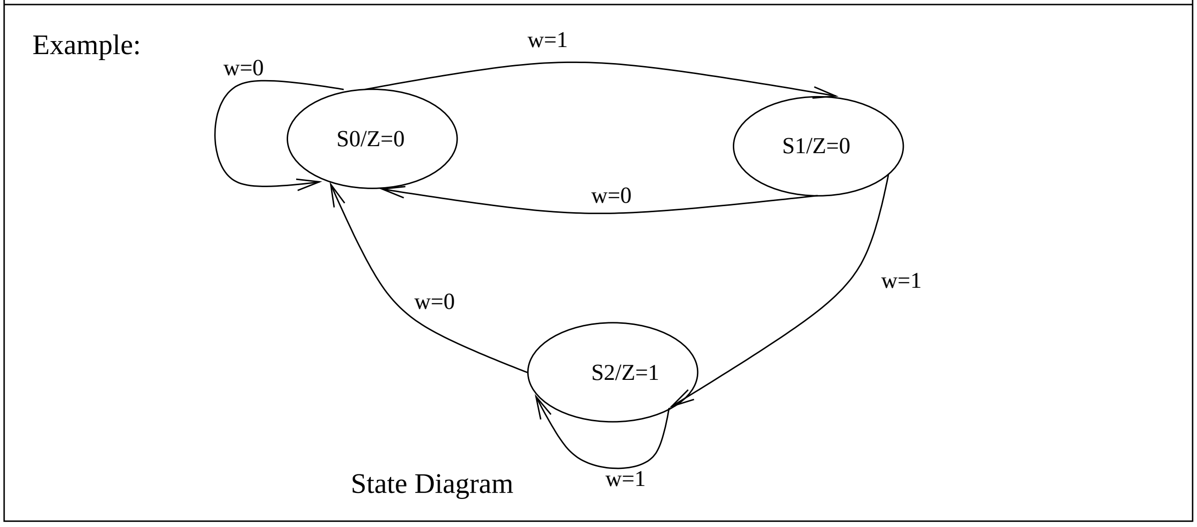


Structure of Sequential Circuits

Design Steps of Sequential Circuits

- **1-State Diagram**
  Describes the behavior of sequential circuit: For a given present state, and inputs it gives next state and outputs
  Example: Generate output=1 if w=1 for two consecutive times (clock edge)





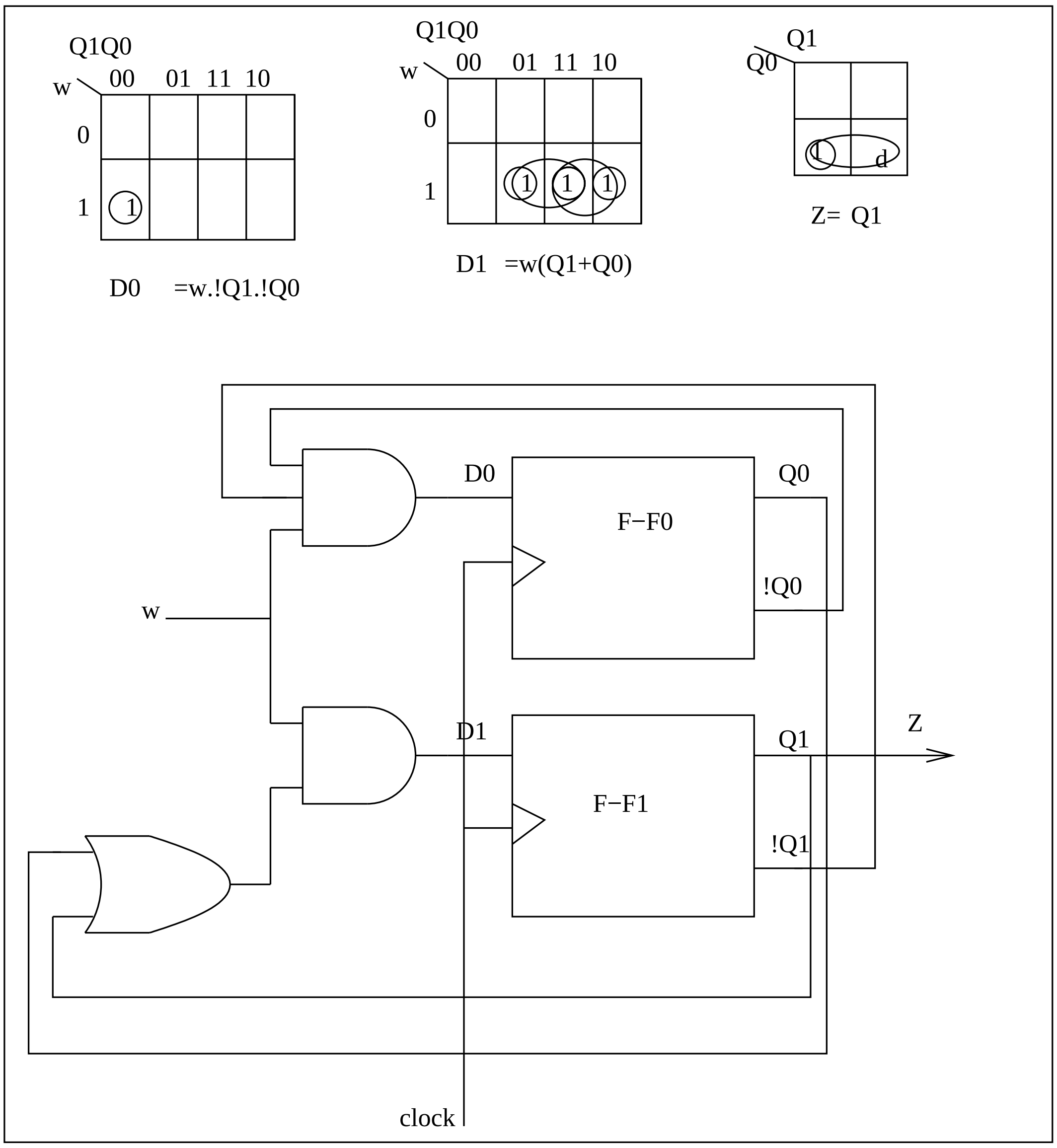
- **2-State Table**

  All states encoded by number of F-F=$log_2 n$ where n=number of states.

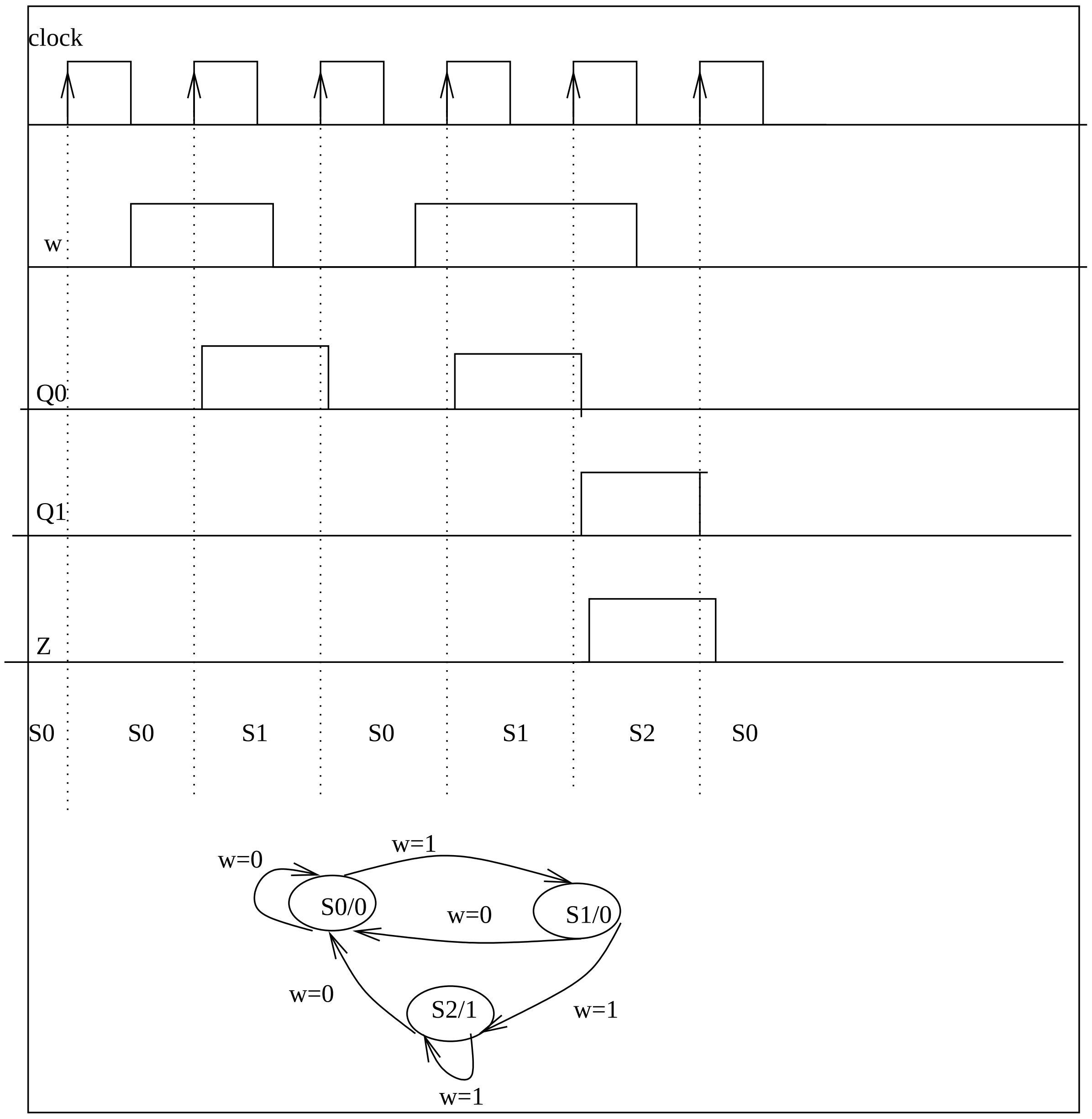
  Need 2 F-F for 3 states

present state Q1Q0	**next state w=0 D1D0**	**next state w=1 D1D0**	**outpu Z**
S0=00	S0=00	S1= 01	0
S1=01	S0=00	S2= 10	0
S2=10	S0=00	S2= 10	1
S3=11	SD	SD	D

- **3-Implementation**
  D0= !Q1.!Q0.w, D1= !Q1.Q0.w + Q1.!Q0.w,
  Z=Q1.!Q0

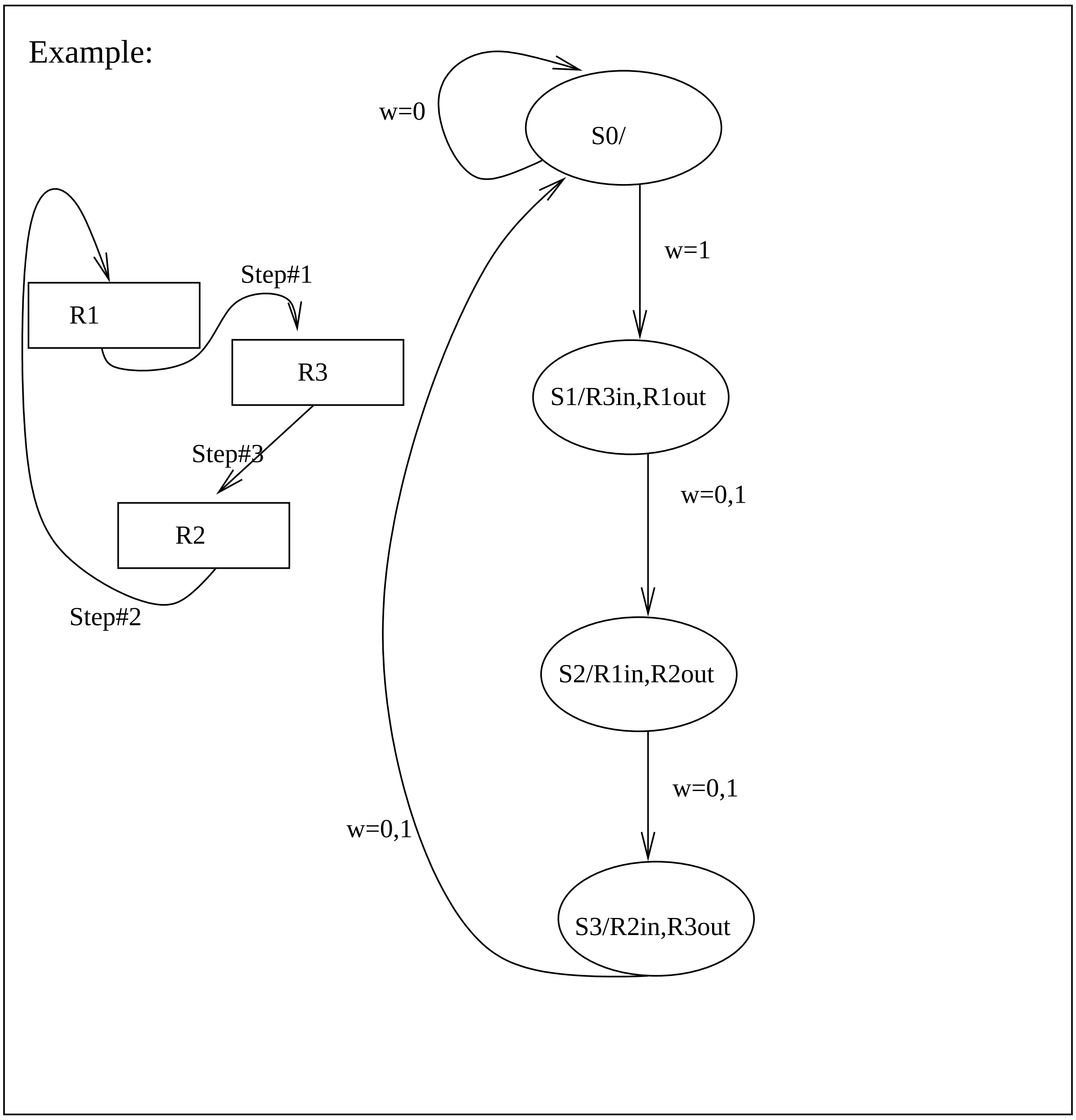


Timing Diagram



Example

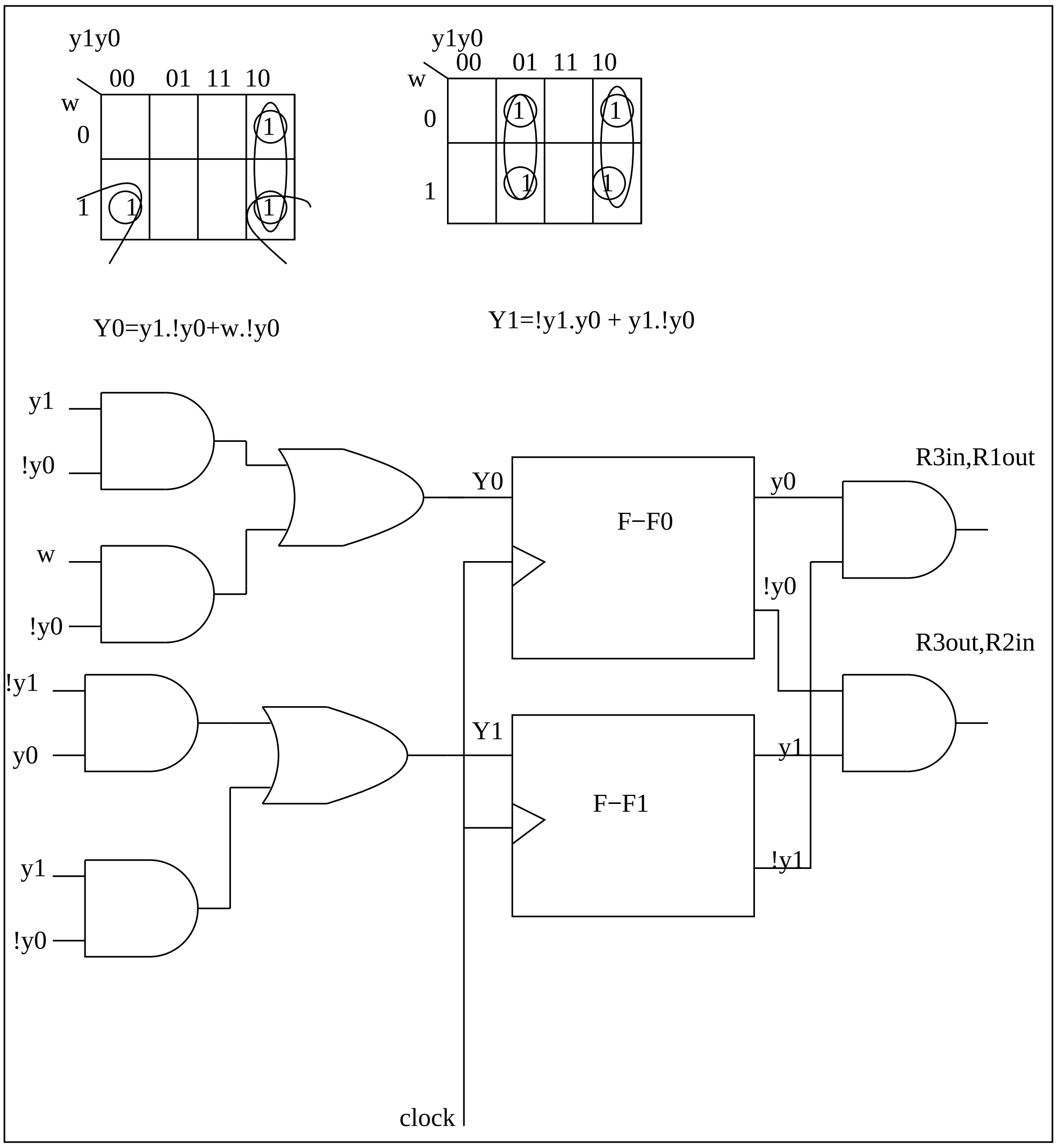
Design a FSM (Finite State Machine) to swap the content of R1, R2



Example: State Table

present state y1y0	next state w=0 Y1Y0	next state w=1 Y1Y0	outputs R1,R2,R3
S0=00	S0=00	S1= 01	
S1=01	S2=10	S2= 10	R3in,R1ou
S2=10	S3=11	S3= 11	R1in,R2ou
S3=11	S0=00	S0=00	R2in,R3ou

Example: Implementation



State Assignment

Assigning values to each state could affect cost of implementation.

If we assign S0=00, S1=01, S2=11 might optimize the cost of implementation.

ONE-HOT Encoding:

Each state uses a F-F

S0,S1,S2,S3 will use 4 F-F as

S0=0001, S1=0010, S2 = 0100, S3=1000

Cost more F-F but reduces complexity of combinational circuit

present state y4y3y2y1	**next state w=0 Y4Y3Y2Y1**	**next state w=1 Y4Y3Y2Y1**	**outputs R1,R2,**
S0=0001	S0=0001	S1=0010	
S1=0010	S2=0100	S2= 0100	R3in,R1
S2=0100	S3=1000	S3= 1000	R1in,R2
S3=1000	S0=0001	S0=0001	R2in,R3

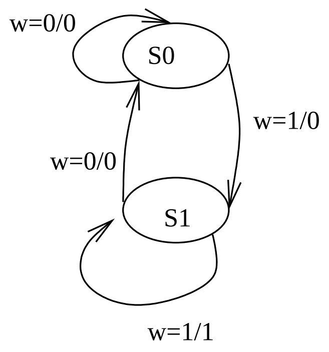
Y1= y4 + !w.y1, Y2=w.y1, Y3= y2, Y4=y3

R2out,R1in= y3

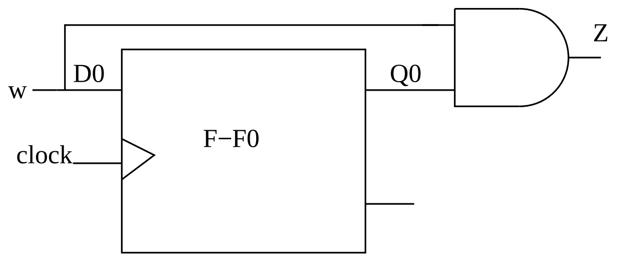
Mealy State Machine

The output is generated from present state ad inputs
Does not need to wait for next state (faster)
Less reliable (output changes as input changes)

Example: Design a Mealy FSM that detects sequence of w=1,1 and generates Z=1 when it occurs.



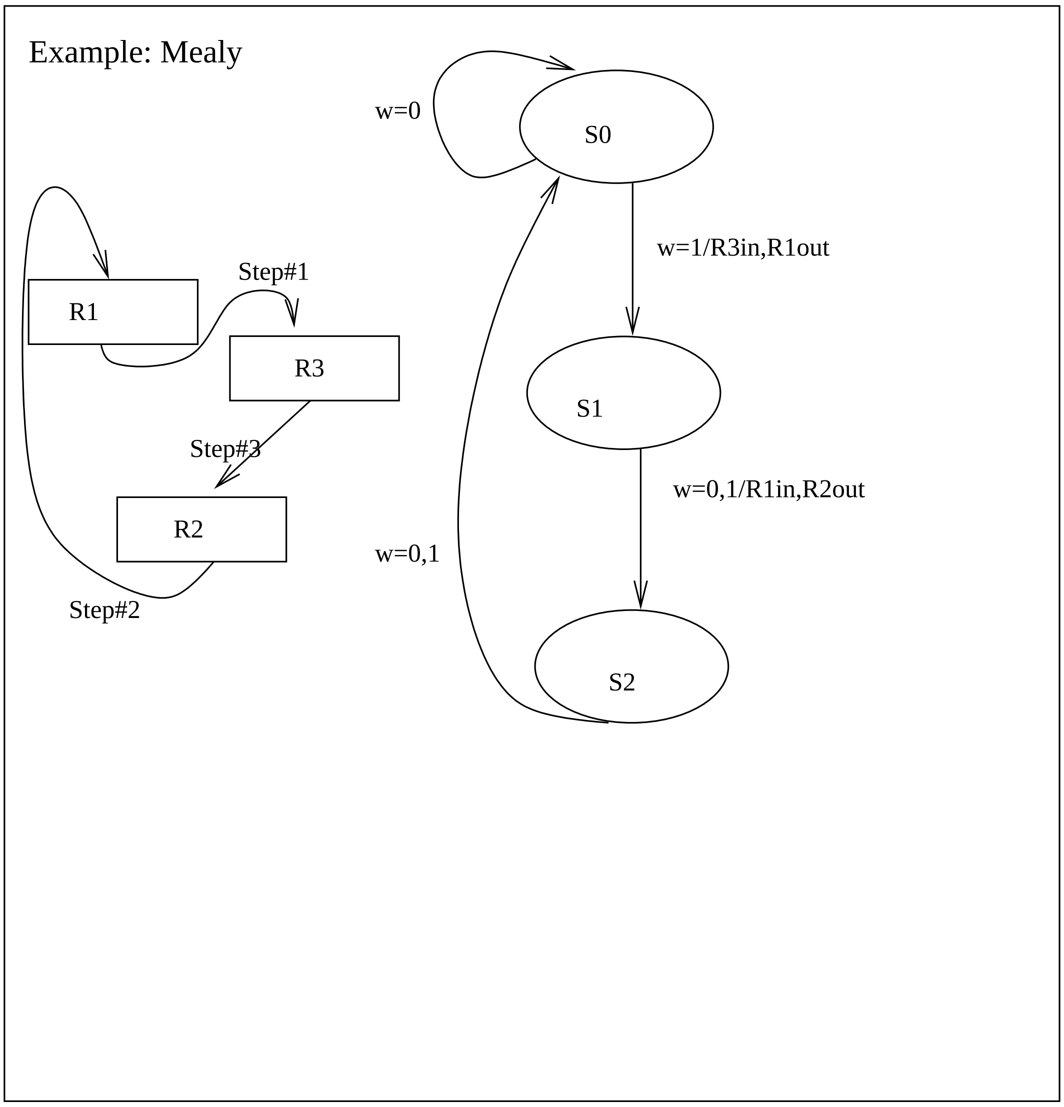
State Machine for Mealy



present state Q0	**next state w=0 D0**	**next state w=1 D0**	**output Z when** w=0,1
S0=0	S0=0	S1= 1	0, 0
S1=1	S0=0	S1= 1	0, 1

D0= w.(!Q0 +Q0)=w , Z=w.Q0

Design a Mealy FSM to swap content of two registers



present state Q1Q0	**next state w=0 D1D0**	**next state w=1 D1D0**	**outputs w=0,1**
S0=00	S0=00	S1= 01	0, R3in,R
S1=01	S2=10	S2= 10	R1in,R2o
S2=10	S0=00	S0= 00	R2in,R3o

D0= w.!Q1.!Q0

D1=!Q1.Q0

R3in=R1out=w.!Q1.!Q0

R1in=R2out=!Q1.Q0

R2in=R3out= Q1.!Q0

Moore FSM VHDL Code for detecting w=1,1 sequence

```
 LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY moore IS
  PORT( Clock,Resetn,w   : IN STD_LOGIC;
        Z   : OUT STD_LOGIC );
END moore;

ARCHITECTURE Behavior OF moore IS
  TYPE State_type IS (A,B,C);
  SIGNAL y: State_type;
BEGIN
  PROCESS(Resetn, Clock)
  BEGIN
     IF Resetn = '0' THEN
              y<=A;

  ELSEIF (Clock'EVENT AND Clock = '1') THEN
```

```
            CASE y IS
             WHEN A =>
              IF w='0' THEN
                  y<=A;
              ELSE
                  y<=B;
              END IF;
            WHEN B =>
              IF w='0' THEN
                  y<=A;
              ELSE
                  y<=C;
              END IF;
            WHEN C =>
              IF w='0' THEN
                  y<=A;
              ELSE
                  y<=C;
              END IF;
             END CASE;
      ENDIF;
     END PROCESS;
       z<= '1' WHEN y=C ELSE '0';
END Behavior;
```

Mealy FSM VHDL Code for detecting w=1,1 sequence

```
 LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY mealy IS
  PORT( Clock,Resetn,w   : IN STD_LOGIC;
        Z   : OUT STD_LOGIC );
END mealy;

ARCHITECTURE Behavior OF mealy IS
  TYPE State_type IS (A,B);
  SIGNAL y: State_type;
BEGIN
  PROCESS(Resetn, Clock)
  BEGIN
     IF Resetn = '0' THEN
              y<=A;

  ELSEIF (Clock'EVENT AND Clock = '1') THEN
```

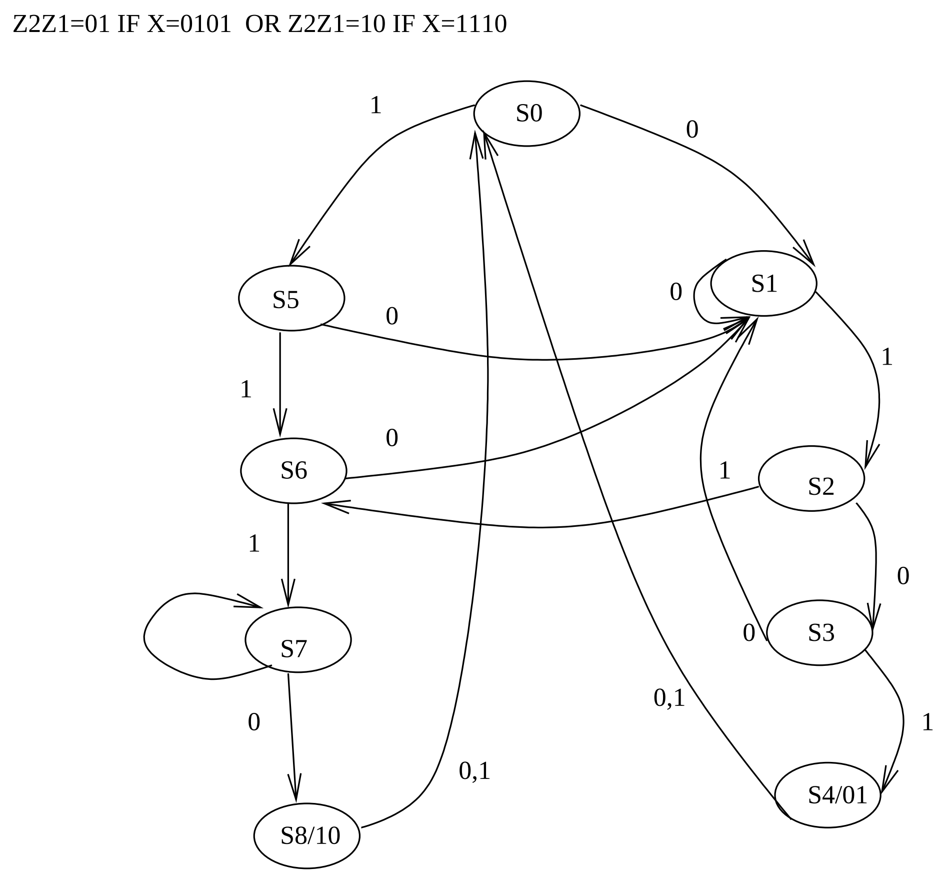
```
         CASE y IS
          WHEN A =>
           IF w='0' THEN  y<=A;
           ELSE  y<=B;
           END IF;
         WHEN B =>
           IF w='0' THEN y<=A;
           ELSE  y<=B;
           END IF;
          END CASE;
   ENDIF;
  END PROCESS;
  PROCESS(y,w)
  BEGIN
     CASE y IS
        WHEN A =>
        z<='0';
        WHEN B=>
        z<= w;
      END CASE
  END PROCESS
END Behavior;
```

Sequence Detector

Example: Design a sequence detector for:

1- If X is 0101 make Z2Z1=01

2-If X=1110 make Z2Z1=10



Sequence Detector (Moore)

©N. Mekhiel

State Minimization

Concept: Any two states are equivalent if for all possible inputs, they produce the same sequence.

Partioning for state Minimization

- Partion states into blocks, each block contains equivalent states
- Partion each block to subblocks based on the successor states if they are found in different blocks
- Partion ends when new partion is the same as previous.
- At the end of partion, all states in any one block are equivalent

State Minimization Example

present state	next state w=0	next state w=1	output Z
A	B	C	1
B	D	F	1
C	F	E	0
D	B	G	1
E	F	C	0
F	E	D	0
G	F	G	0

```
Partion#1: {A,B,C,D,E,F,G}  one block


Partion#2    Z=1                    Z=0
            {A,B,D}               {C,E,F,G}

Partion#3  w=0     w=1             w=0      w=1
         {B,D,B},{C,F,G}       {F,F,E,F}, {ECDG}
                                             D not


Partion#4    {A,B,D},          {C,E,G}, {F}

             w=0    w=1          w=0  w=1
            {B,D,B}, {C,F,G}     ----  ----- - -
                        F not in same block


Partion#5  {A,D}, {B},           {C,E,G}{F}
            w=0, 1


 Partion#5 is the same as Partion #4


 A=D,  C=E=G
```

Incompletely Specified FSM

present state	next state w=0	next state w=1	output w=0 w=
A	B	C	0 0
B	D	-	0 -
C	F	E	0 1
D	B	G	0 0
E	F	C	0 1
F	E	D	0 1
G	F	-	0 -

```
Assume Z=0 for all unspecified outputs

Partion#1: {A,B,C,D,E,F,G}  one block


Partion#2    Z=1                    Z=0
            {C,E,F}               {A,B,D,G}

Partion#3  w=0     w=1           w=0      w=1
         {F,F,E},{E,C,D}       {B,D,B,F}, {C,-,G,-
                      D             F        G n

Partion#4    {CE},{F}            {A,B},{D},{G}
             0  1                 0
            {F,F}{EC}             {BD}
                                    D not in same

 Partion#5 C,E}{F}{A}{B}{D}{G}
 Partion#5=Partion#6 then C=E  Only 6 states
```

```
Assume Z=1 for all unspecified outputs

Partion#1: {A,B,C,D,E,F,G}  one block


Partion#2    Z=1                         Z=0
           {B,C,E,F,G}                   {A,D}

Partion#3  w=0     w=1                w=0      w=1
         {D,F,F,E,F},{-,E,C,D-}        {B,,B},{C,E}
          D                 D            not in sme

Partion#4   {B}{CEG},{F},                  {A,D}
                0  1                        0  1
            {F,F,F}{EC-}                   {B,B}{C,E
                     - make it C
Partion#5 =Partion#4  and C=E=G  also A=D  Four st
```