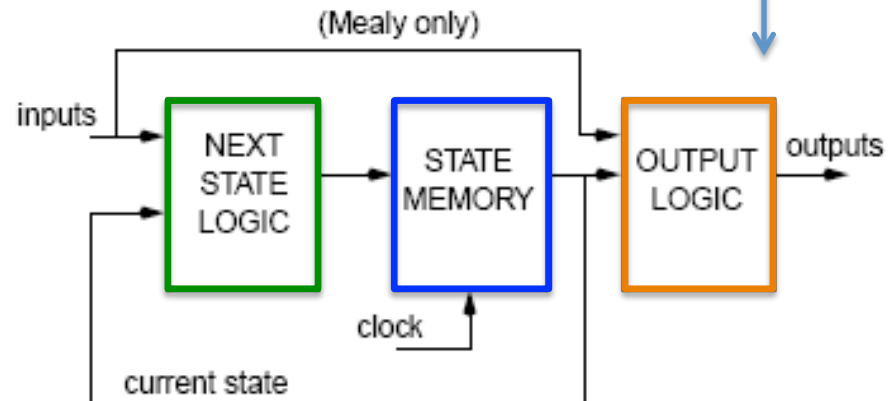
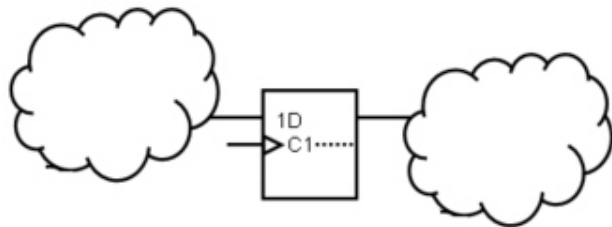


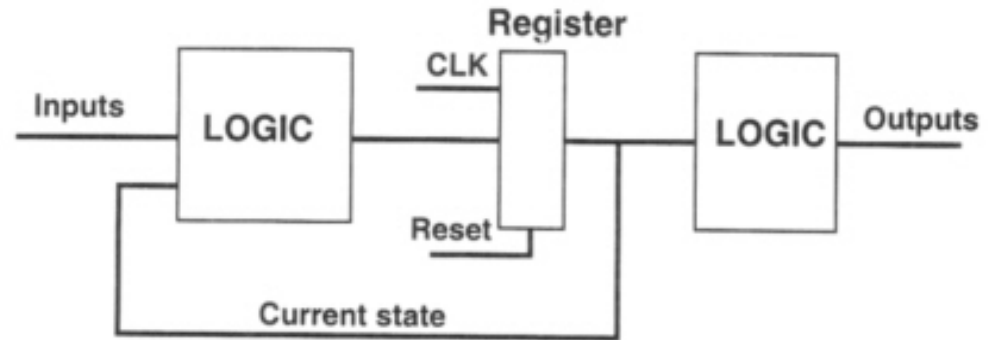
# Finite State Machine

- Model of behavior composed of a **finite number of states**, **input events**, **transitions between those states** (rules or conditions), and **actions**.
- Effective method for implementing control functionality.
- FSM design - just a step beyond sequential design:
  - HW implementation **requires a register to store state variables**, **a block of combinational logic which determines the state transition**, and **a block of combinational logic that determines the output of the FSM**.
- Current state is stored in registers and updated synchronous to the clock
- Two phases:
  - calculate new state
  - new state is sampled into a register

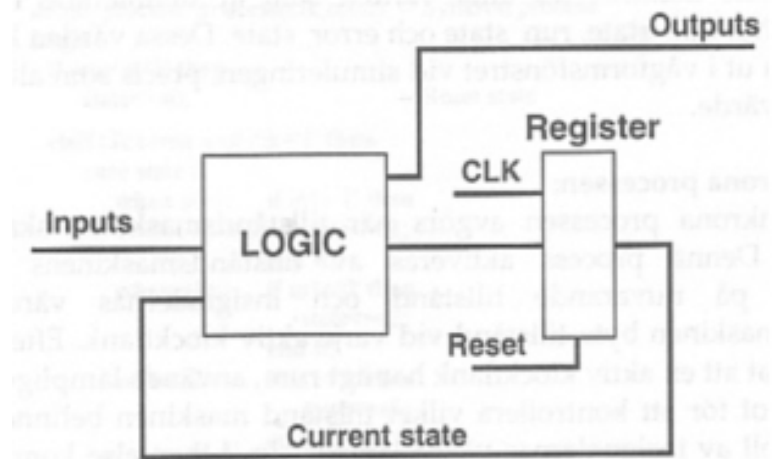


# Two main types

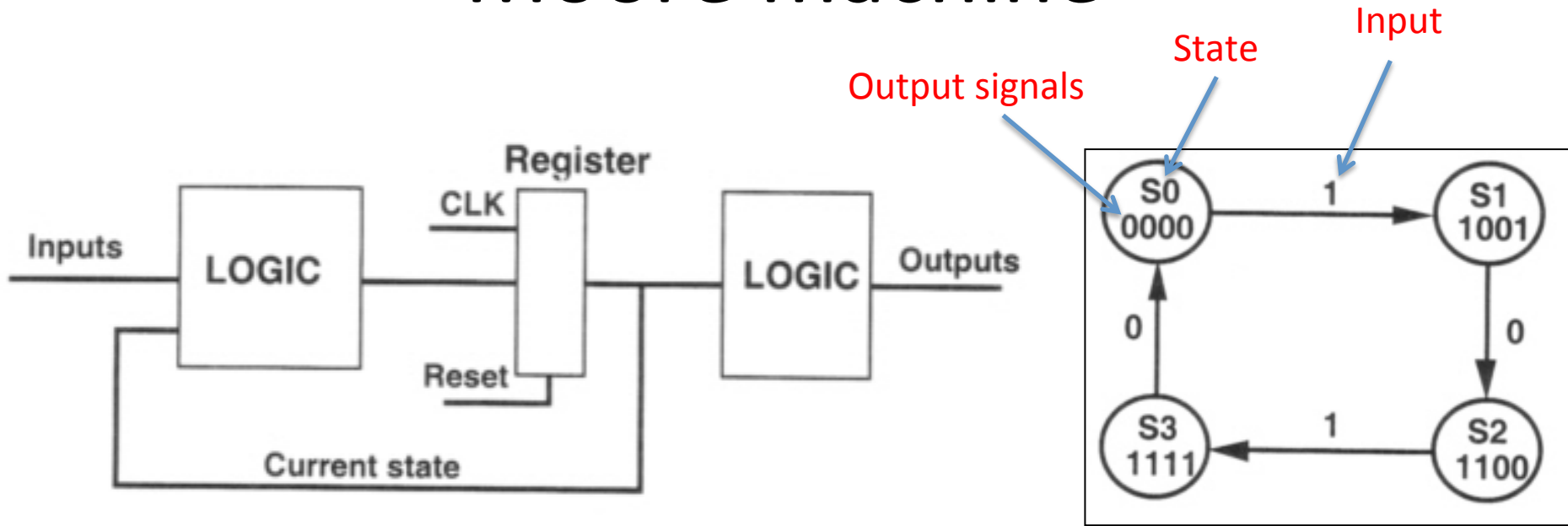
- Moore machine
  - Edward F. Moore, 1956



- Mealy machine
  - George H. Mealy, 1955

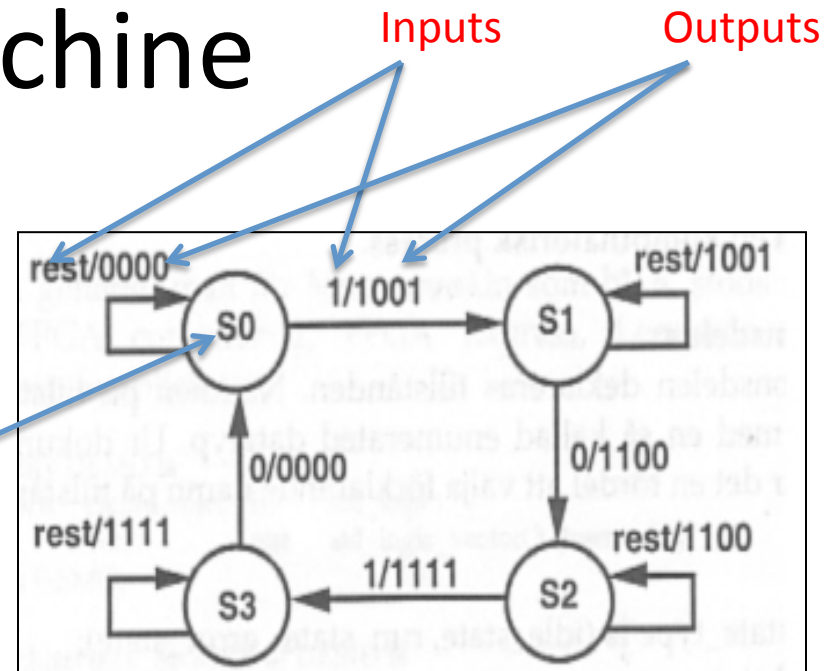
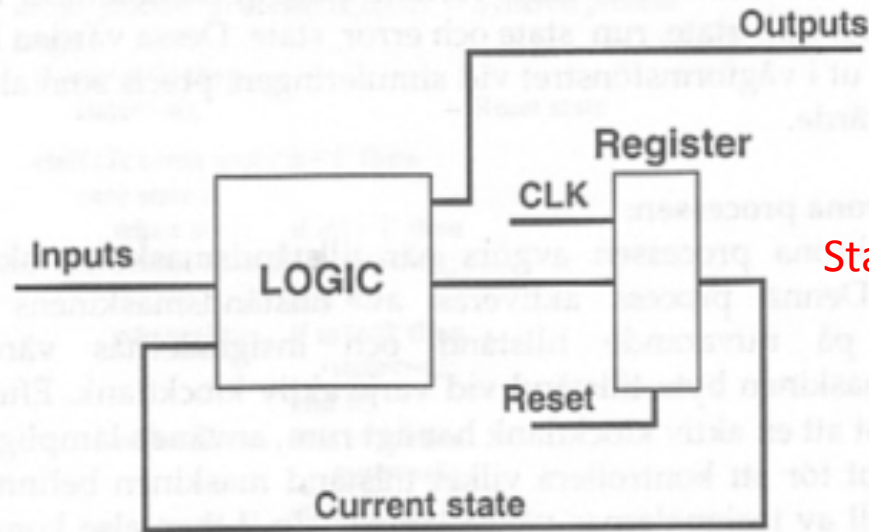


# Moore machine



- Outputs of Moore machine are a function of the present state only
- Output transitions are “synchronous” to system clock
- Propagation delay through output logic nevertheless leads to asynchronous outputs which can lead to slower operating frequencies

# Mealy machine



- Outputs of Mealy machine are a function of the present state and all the inputs.
- Outputs transitions are asynchronous to the clock
  - They change immediately when the inputs change
- => A Mealy machine works one clock cycle in advance of a Moore machine

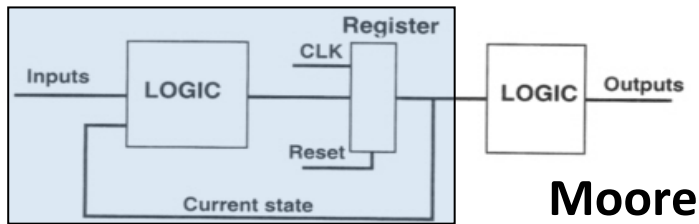
# Design of FSM in VHDL

## Declaration of states

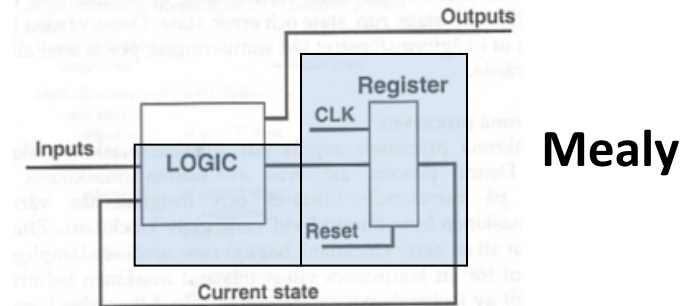
Choose explanatory names using enumerated types

```
--Declaration part-----  
type state_type is (IDLE,START,TASK1,DELAY,TASK2,STOP);  
signal current_state : state_type;  
-----
```

## Clocked process

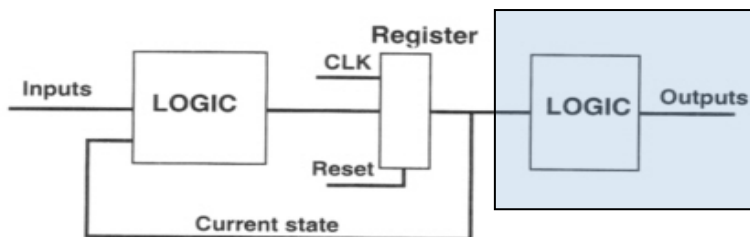


Moore

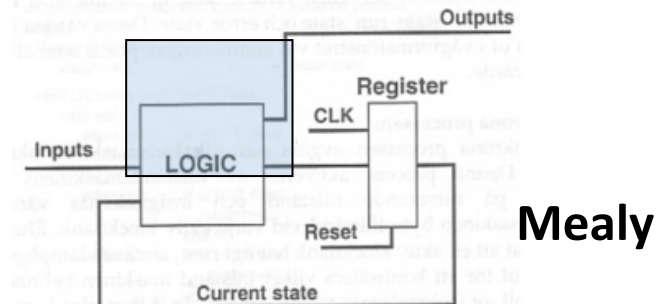


Mealy

## Combinational output process



Moore



Mealy

# VHDL structure

Declaration



Clocked process



Combinational  
process



```
library ieee;
use ieee.std_logic_1164.all;

entity statemachine is
port(--inputs,outputs);
end statemachine;

architecture skeleton of statemachine is

--Declaration part-----
type state_type is (IDLE,START,TASK1,DELAY,TASK2,STOP);
signal current_state : state_type;
-----

begin
-----
--synchronous process-----
state_proc: process(clock,reset) --clocked process
begin
    --update active state
end process;
-----

--Combinatoric output process-----


output_proc: process(current_state      ,[inputs])
begin
    --moore & mealy      --mealy
    --output assignment
end process;
-----

end skeleton;
```

# Enumerated data type

- It is possible to define your own enumerated data type in VHDL
- This data type allows a user to specify the list of legal values that a variable or signal of the defined type may be assigned
- Commonly used for state machines:

```
type state_type is (start, idle, waiting, run);  
signal state : state_type
```



4 states → 2 bit vector

- Most synthesis tools can build logic from enumerated types



# Examples of predefined types

```
type std_ulogic is ( 'U', -- Uninitialized
                    'X', -- Forcing unknown
                    '0', -- Forcing      0
                    '1', -- Forcing      1
                    'Z', -- High impedance
                    'W', -- Weak          unknown
                    'L', -- Weak          0
                    'H', -- Weak          1
                    ' - ', -- Don't care
                    );
```

std\_logic\_1164.vhd

```
type std_logic is resolved std_ulogic;
```

---

type boolean is (false, true);  Converted to 1 and 0 during synthesis

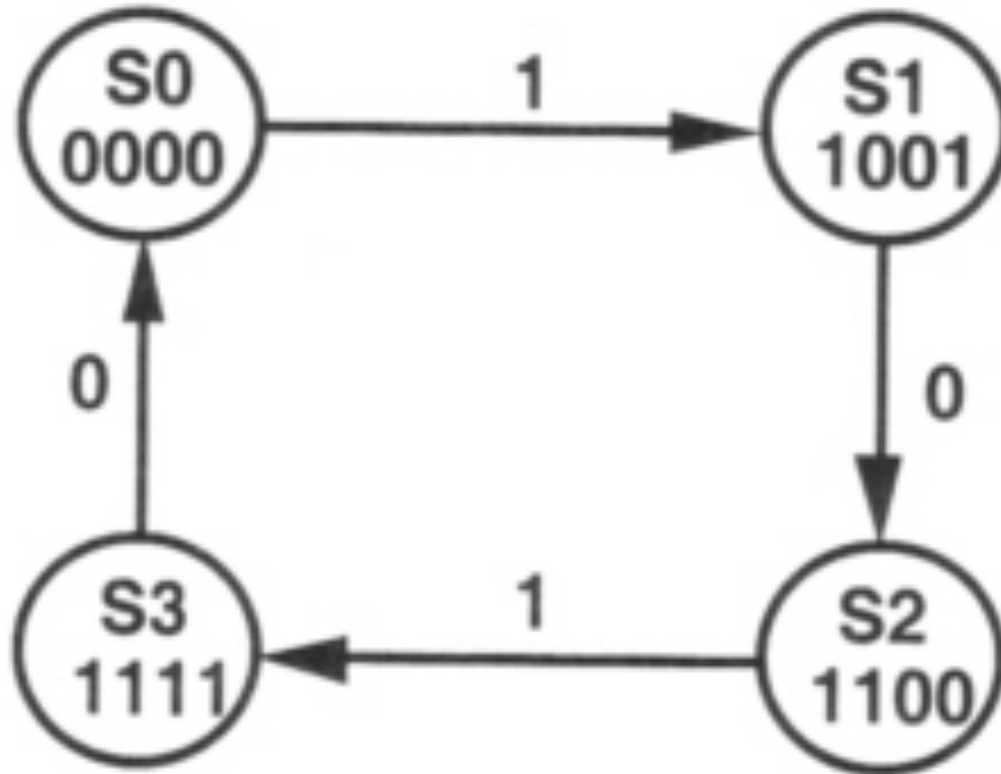
```
type bit is ('0', '1');
```

standard.vhd

# Design of FSM in VHDL

- 1 process
- 2 processes
  - Clocked state process
  - Combinational output process
- 3 processes
  - Combinational coding of next state
  - Clocked update of present state (present state  $\leq$  next state)
  - Combinational output process

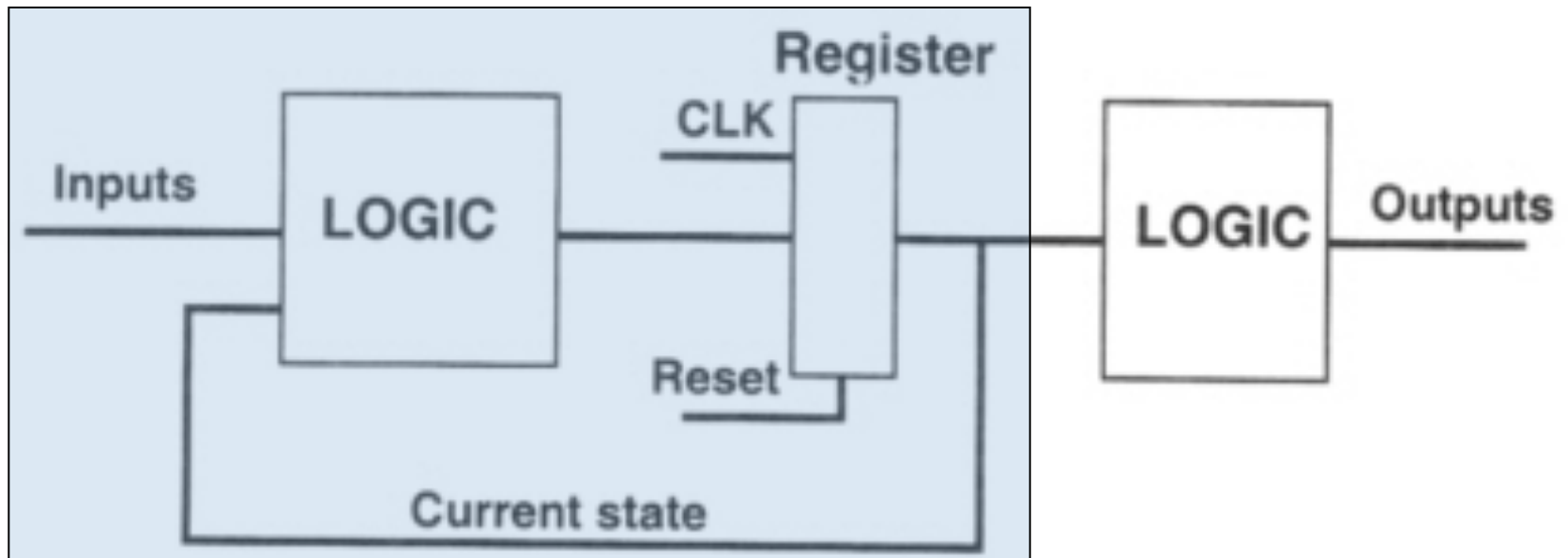
# Example – Moore (2 processes)



# Declaration

```
1 library ieee;
2 use ieee.std_logic_1164.all;
3
4 entity statemachine is
5 port(clk, test ,areset : in std_logic;
6       out1: out std_logic_vector(3 downto 0));
7 end statemachine;
8
9 architecture moore of statemachine is
10
11 type state_type is (s0,s1,s2,s3);           --state declaration
12 signal state: state_type;
```

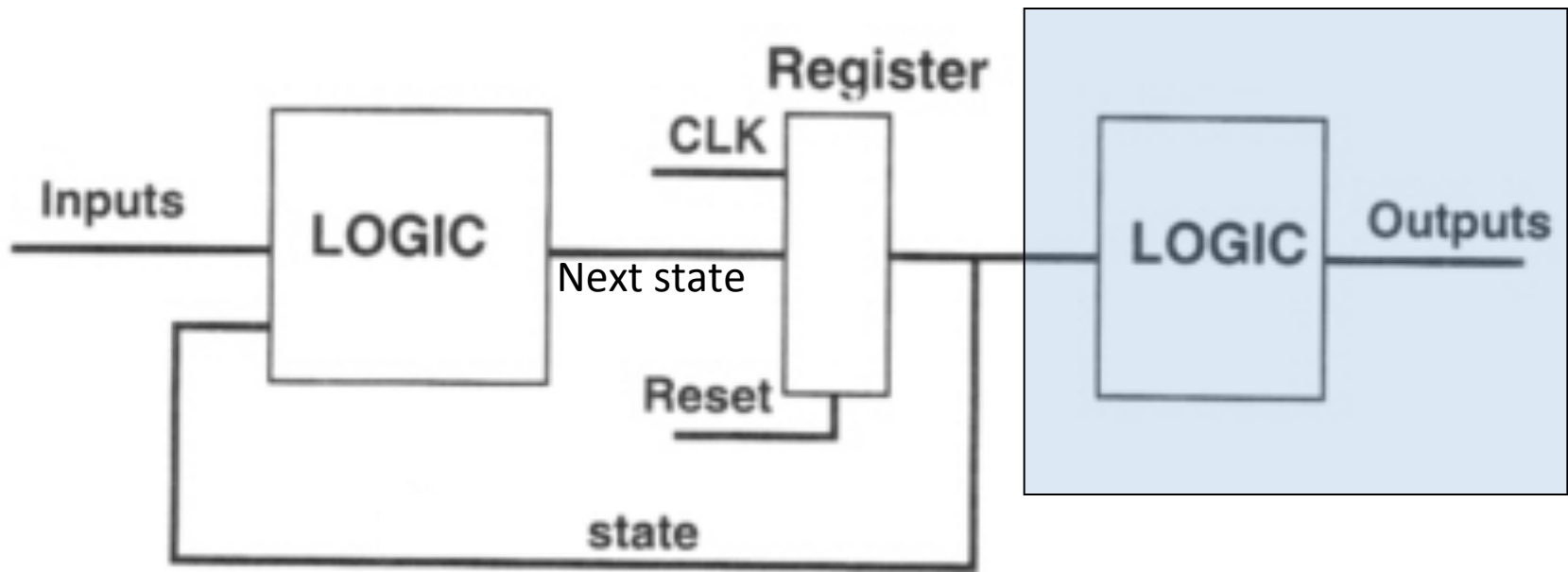
# Clocked process



# Clocked process

```
16 state_proc: process(clk, areset) --clocked process
17 begin
18     if areset = '1' then
19         state <= s0;
20     elsif rising_edge(clk) then
21         --state <= state; --keep old value if not changed
22         case state is
23             when s0 =>
24                 if test = '1' then
25                     state <= s1;
26                 end if;
27             when s1 =>
28                 if test = '0' then
29                     state <= s2;
30                 end if;
31             when s2 =>
32                 if test = '1' then
33                     state <= s3;
34                 end if;
35             when s3 =>
36                 if test = '0' then
37                     state <= s0;
38                 end if;
39         end case;
40     end if;
41 end process;
```

# Combinational output process



Outputs of Moore machine are a function of the present state only.

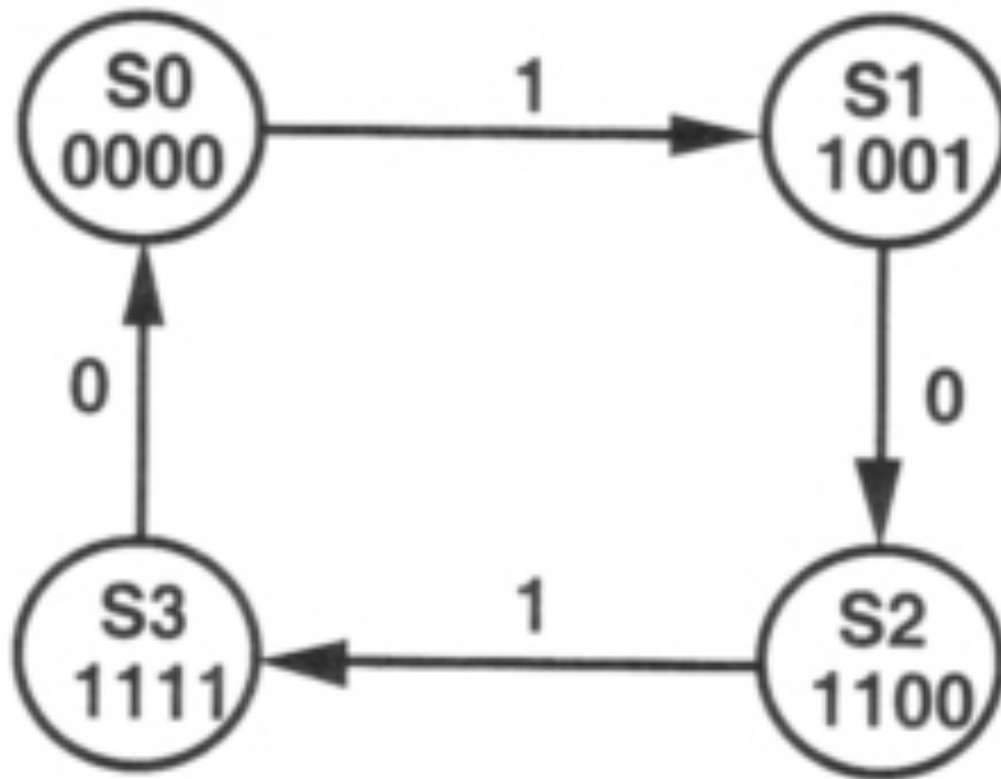
# Combinational output process

```
43 output_proc: process(state)  --combinatoric process
44 begin
45     case state is
46         when s0 =>
47             out1 <= "0000";
48         when s1 =>
49             out1 <= "1001";
50         when s2 =>
51             out1 <= "1100";
52         when s3 =>
53             out1 <= "1111";
54     end case;
55 end process;
56 end moore;
```

Outputs of Moore machine are a function of the present state only.



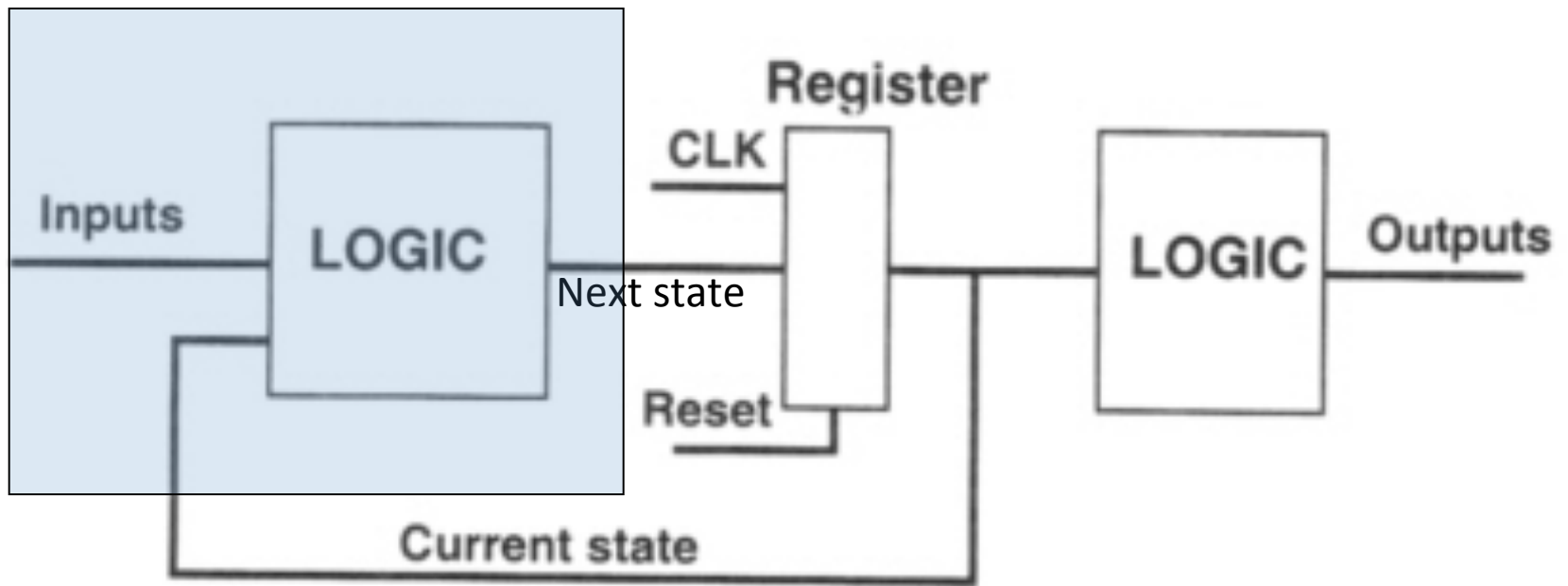
# Example - Moore (3 processes)



# Declaration

```
1 library ieee;
2 use ieee.std_logic_1164.all;
3
4 entity statemachine is
5 port(clk, test ,areset : in std_logic;
6       out1: out std_logic_vector(3 downto 0));
7 end statemachine;
8
9 architecture moore of statemachine is
10
11 type state_type is (s0,s1,s2,s3);           --state declaration
12 signal next_state: state_type;
13 signal current_state : state_type;
14
```

# Decoding of next state

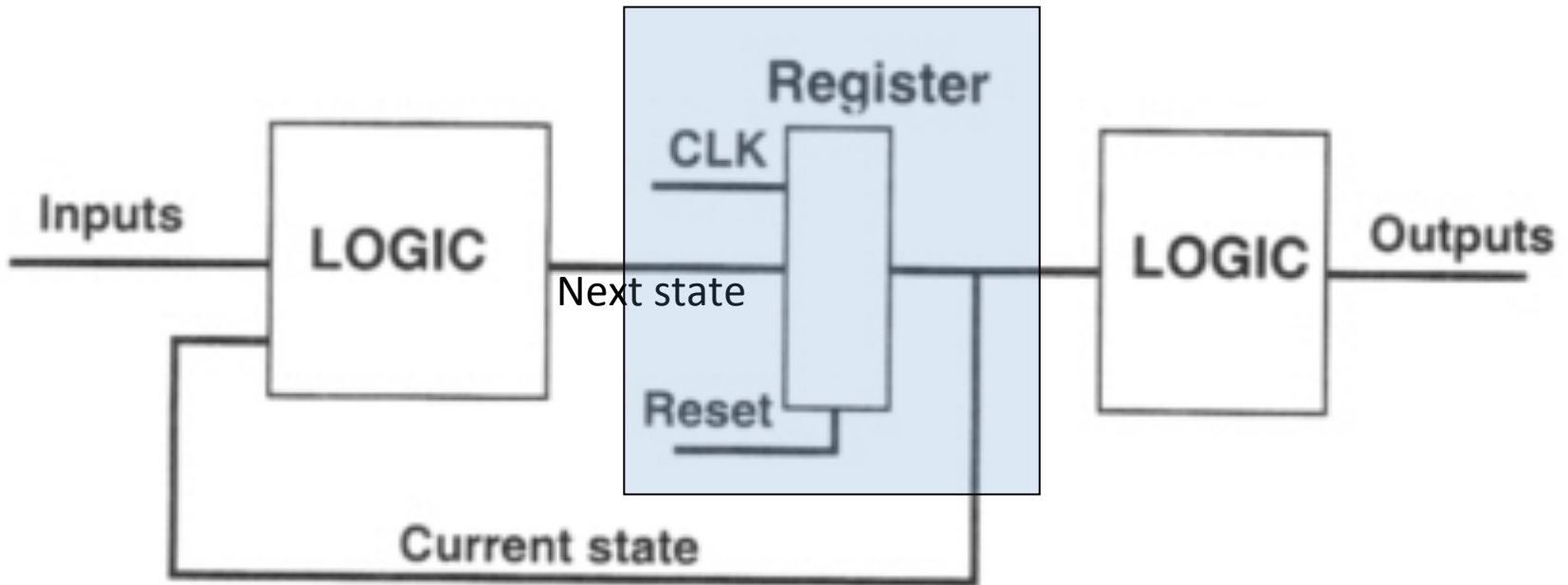


# Decoding of next state

```
17 state_decode_proc: process(current_state, test) --combinatoric process
18 begin
19 |
20     case current_state is
21         when s0 =>
22             if test = '1' then
23                 next_state <= s1;
24             else
25                 next_state <= s0;
26             end if;
27         when s1 =>
28             if test = '0' then
29                 next_state <= s2;
30             else
31                 next_state <= s1;
32             end if;
33         when s2 =>
34             if test = '1' then
35                 next_state <= s3;
36             else
37                 next_state <= s2;
38             end if;
39         when s3 =>
40             if test = '0' then
41                 next_state <= s0;
42             else
43                 next_state <= s3;
44             end if;
45     end case;
46 end process;
```

# Clocked process

- Update of present state (current state)

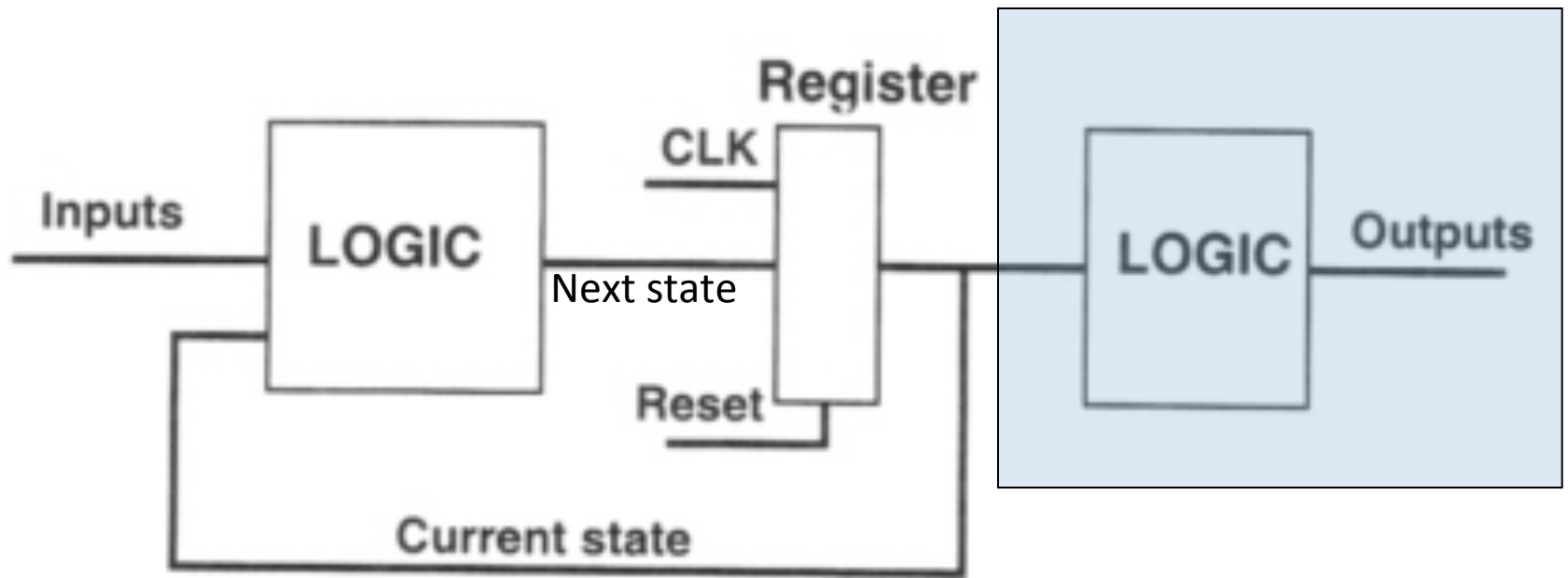


# Clocked process

- Update of present state (current state)

```
48 state_proc: process(clk, areset) --clocked process
49 begin
50     if areset = '1' then
51         current_state <= s0;
52     elsif rising_edge(clk) then
53         current_state <= next_state;
54     end if;
55 end process;
```

# Combinational output process



Outputs of Moore machine are a function of the present state only.

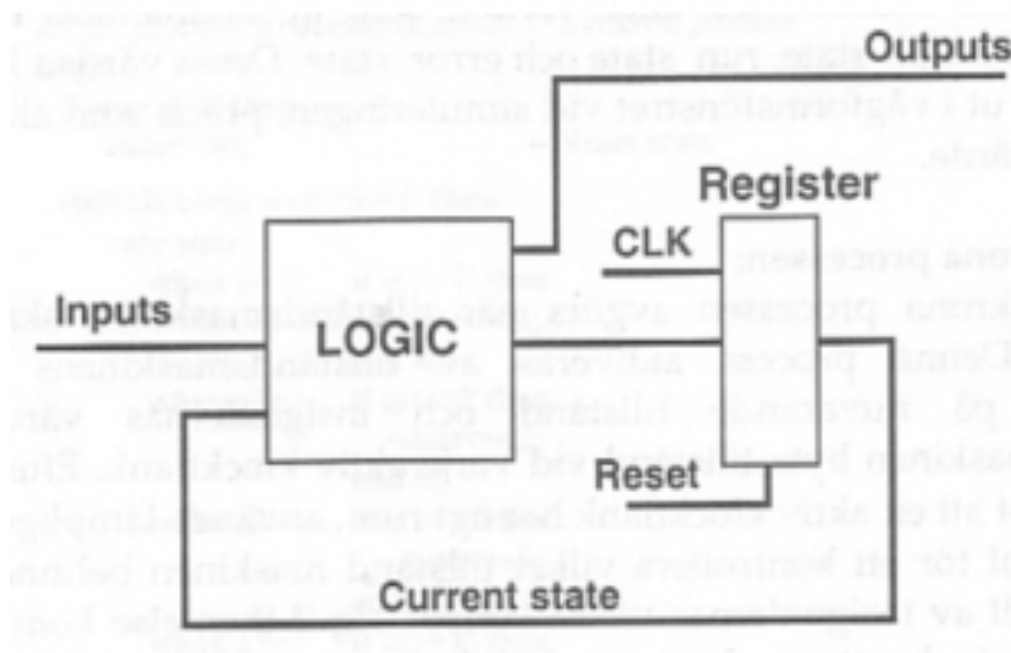
# Combinational output process

```
58 output_proc: process(current_state)  --combinatoric process
59 begin
60     case current_state is
61         when s0 =>
62             out1 <= "0000";
63         when s1 =>
64             out1 <= "1001";
65         when s2 =>
66             out1 <= "1100";
67         when s3 =>
68             out1 <= "1111";
69     end case;
70 end process;
71 end moore;
```

Outputs of Moore machine are a function of the present state only.

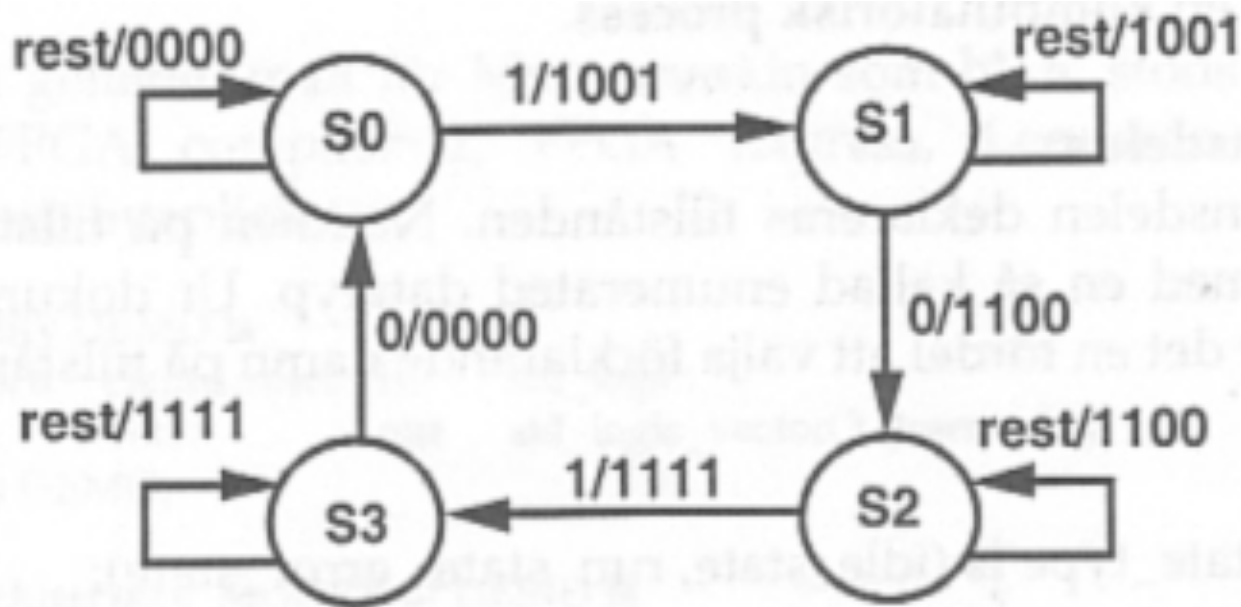


# Mealy



- Outputs of Mealy machine are a function of the present state and all the inputs.

# Mealy



- Outputs of Mealy machine are a function of the present state and all the inputs.

# Declaration

```
library ieee;
use ieee.std_logic_1164.all;

entity statemachine_mealy is
port(clk, test ,areset : in std_logic;
      out1: out std_logic_vector(3 downto 0));
end statemachine_mealy;

architecture mealy of statemachine_mealy is

type state_type is (s0,s1,s2,s3);           --state declaration
signal state: state_type;
```

*Identical to Moore machine*

# Clocked process

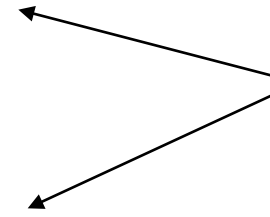
```
state_proc: process(clk, areset) --clocked process
begin
    if areset = '1' then
        state <= s0;
    elsif rising_edge(clk) then
        case state is
            when s0 =>
                if test = '1' then
                    state <= s1;
                end if;
            when s1 =>
                if test = '0' then
                    state <= s2;
                end if;
            when s2 =>
                if test = '1' then
                    state <= s3;
                end if;
            when s3 =>
                if test = '0' then
                    state <= s0;
                end if;
        end case;
    end if;
end process;
```

*Identical to Moore machine*

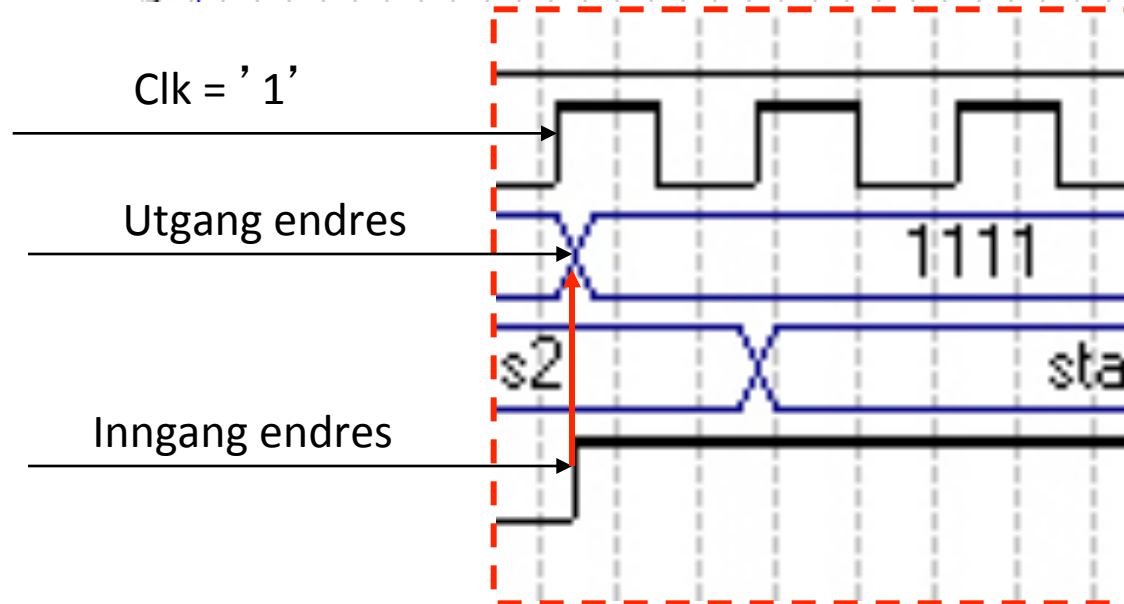
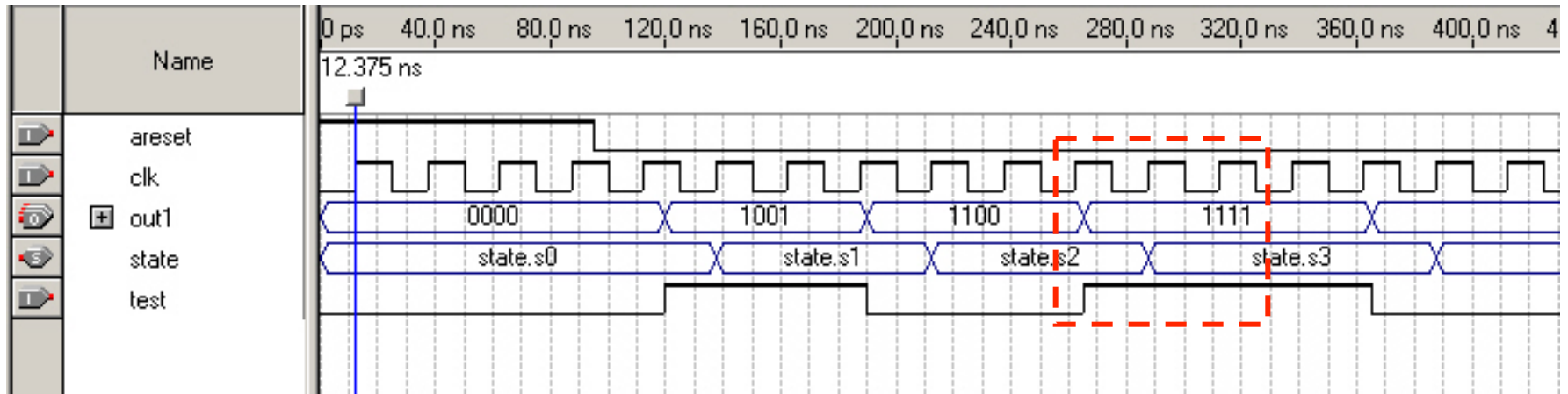
# Combinational output process

```
output_proc: process(state,test)  --combinatoric process
begin
  case state is
    when s0 =>
      if test = '1' then
        out1 <= "1001";
      else
        out1 <= "0000";
      end if;
    when s1 =>
      if test = '0' then
        out1 <= "1100";
      else
        out1 <= "1001";
      end if;
    when s2 =>
      if test = '1' then
        out1 <= "1111";
      else
        out1 <= "1100";
      end if;
    when s3 =>
      if test = '1' then
        out1 <= "0000";
      else
        out1 <= "1111";
      end if;
  end case;
end process;
```

**Outputs directly  
dependent on  
value of inputs**

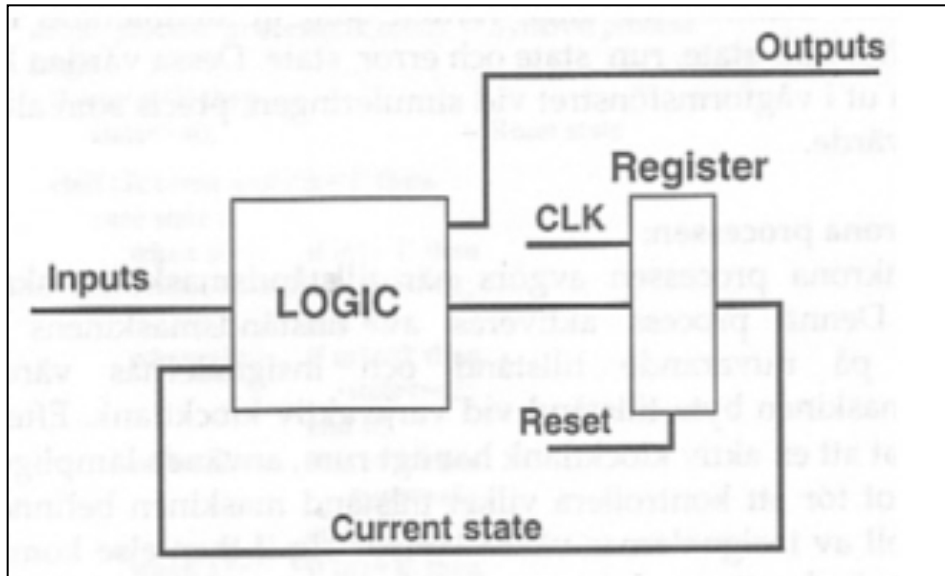
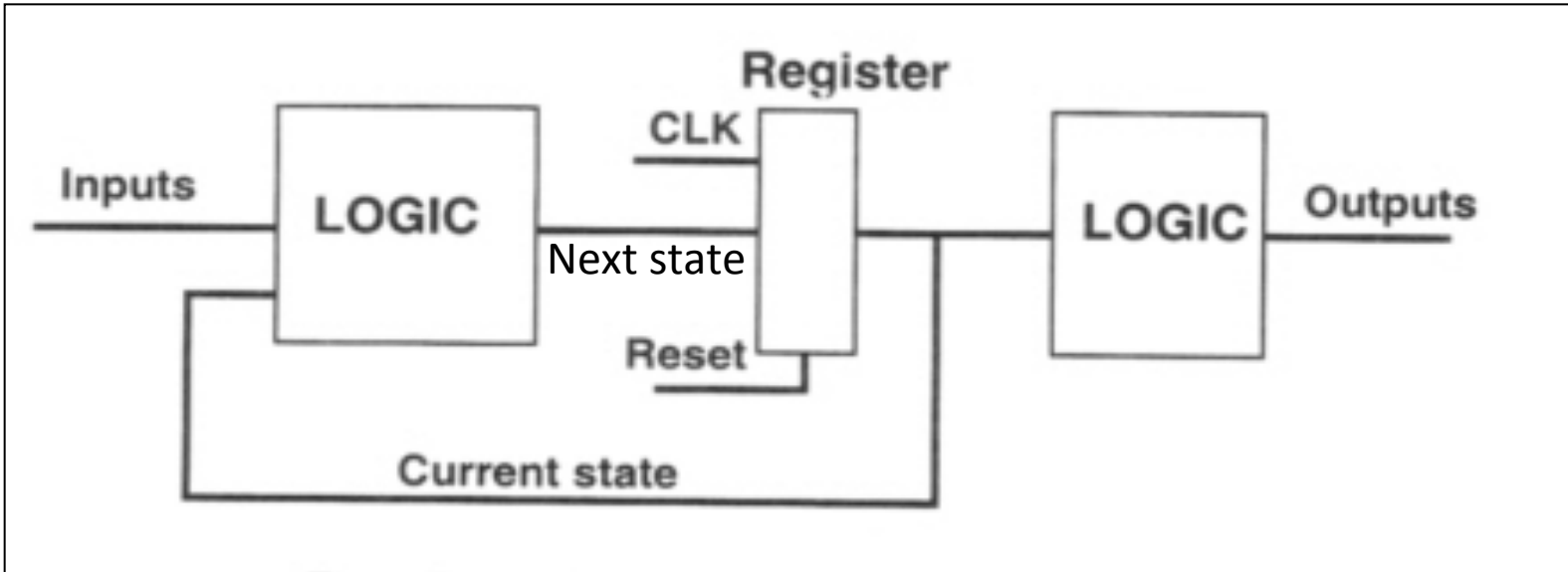


# Mealy



Utgang følger både inngang og nå-tilstand, mens nå-tilstand skifter på klokken

# Mealy vs Moore



# Mealy vs Moore

