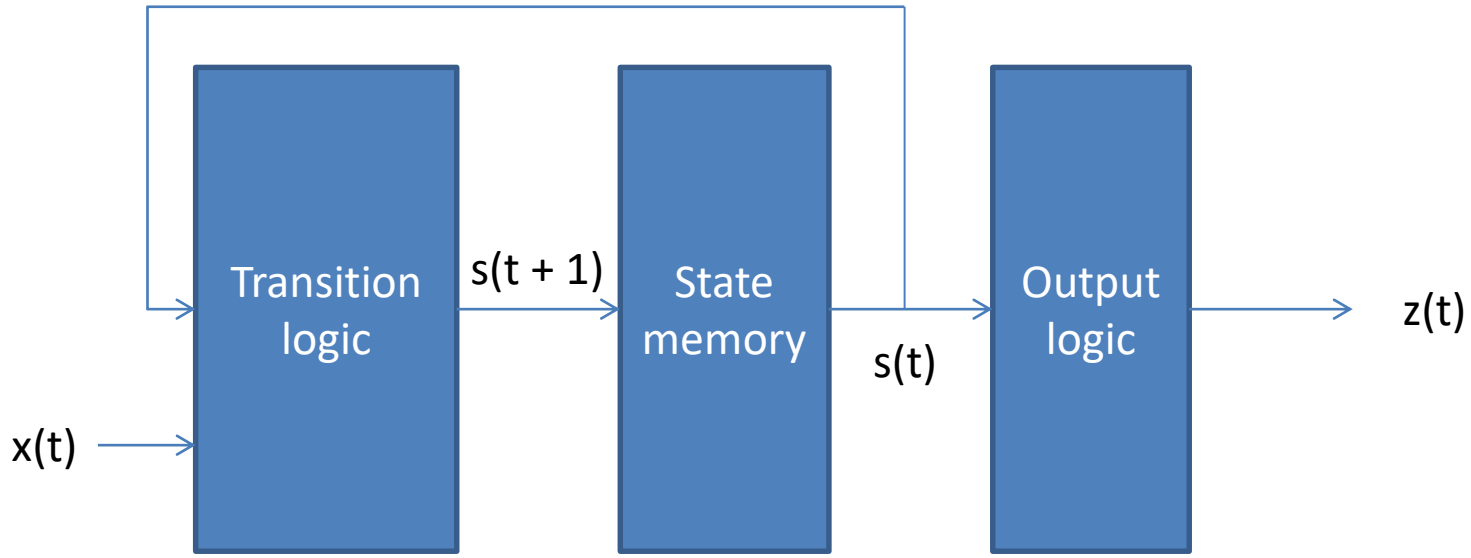
Finite state machine

Digital design

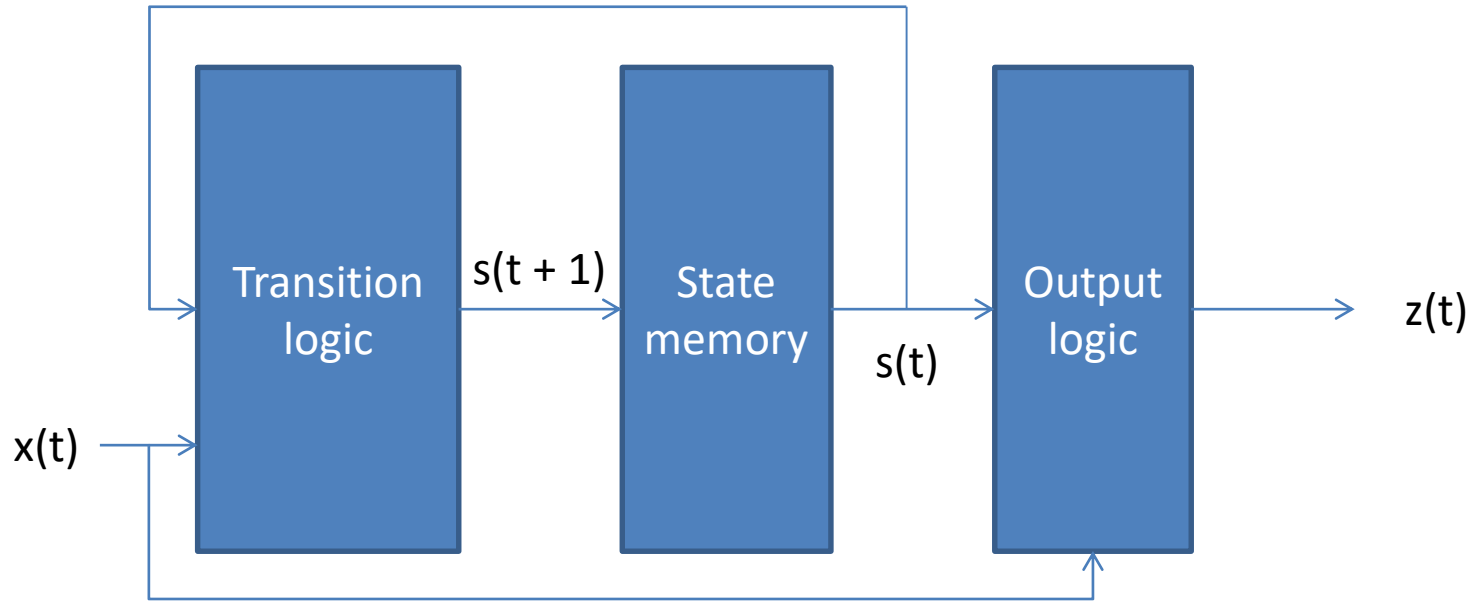
Moore automata

- Next state depends on the state and input
- Izhod je odvisen od stanja



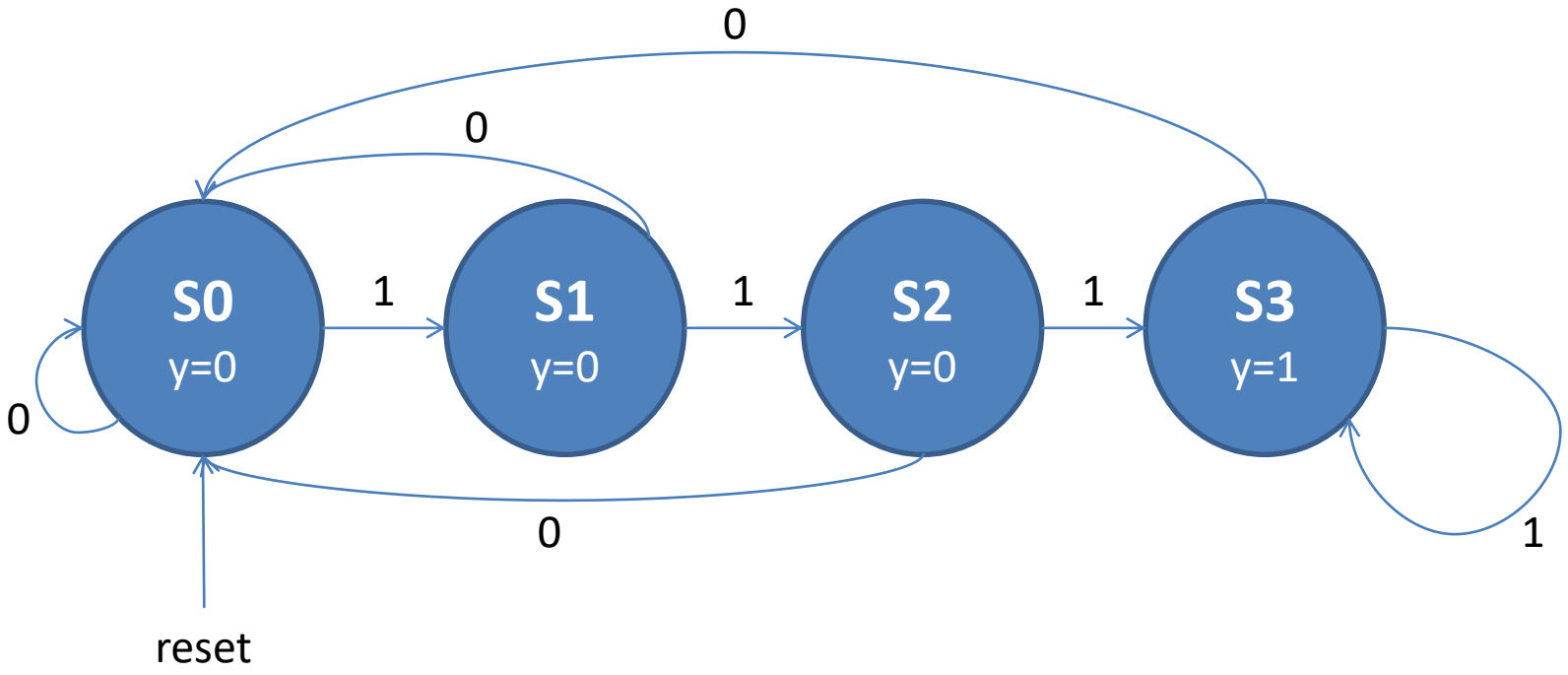
Mealy automata

- Next state depends on the state and input
- The output depends on the state and input



Example

- Moore automata to recognize 111
  - Output y=1 when found



Coding a finite state machine

- Describe
  - The states
  - State memory
  - Transition logic
  - Output logic

Example

```
architecture Behavioral of example_automata is
        type state_type is ( s0 , s1 , s2 , s3   ); -- possible states
        signal state , next_state : state_type;
begin
        SYNC_PROC: process ( clk_i ) -- state memory
        begin
                if (clk_i'event and clk_i = '1') then
                        if ( rst_i = ’1’) then
                                state <= s0;
                        else
                                state <= next_state;
                        end if;
                end if;
        end process;
```

Primer

```
NEXT_STATE_DECODE: process (state , x )
begin

        next_state <= state;
        case ( state ) is
                when s0 =>
                        if x = '0' then          next_state <= s0;
                        else                     next_state <= s1;
                        end if;
                when s1 =>
                        if x = '0' then          next_state <= s0;
                        else                     next_state <= s2;
                        end if;
```

Primer

```
        when s2 =>
                if x = '0' then          next_state <= s0;
                else                     next_state <= s3;
                end if;
        when s3 =>
                if x = '0' then          next_state <= s0;
                else                     next_state <= s3;
                end if;
        when others =>                   next_state <= s0;
    end case;
end process;
```

Example

```
OUTPUT_DECODE: process ( state ) – output logic
        begin
                y <= '0' ;
                case ( state ) is
                        when s0 =>          y <= '0' ;
                        when s1 =>          y <= '0' ;
                        when s2 =>          y <= '0' ;
                        when s3 =>          y <= '1' ;
                        when others =>      y <= '0';
                end case;
        end process;
```

Mealyev avtomat

- Izhod je odvisen od stanja in vhoda

```
OUTPUT_DECODE: process ( state, vhod1,vhod2,... ) -- logika za izhod
        begin
                y <= '0' ;
                case ( state ) is
                        when s0 =>
                                        if vhod1= '0'
                                                y <= '0' ;
                                        else
                                                y <= '1;
...
```

Exercise

- Create a negative edge automata