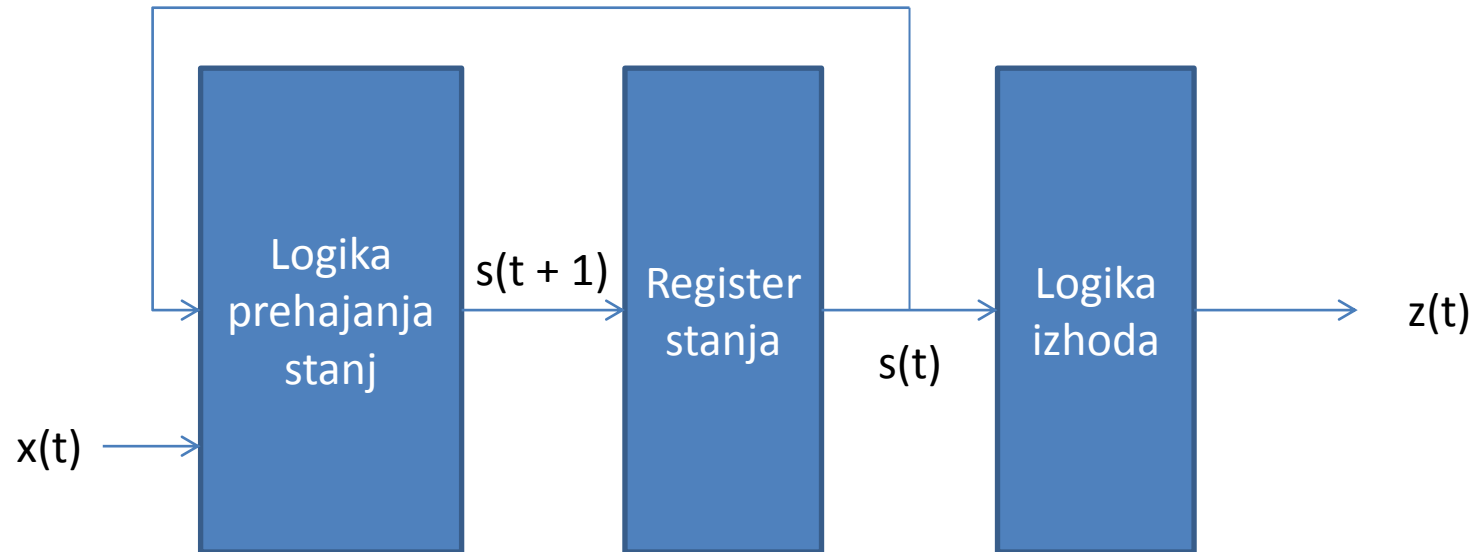


# Končni avtomati

Digitalno načrtovanje 2013/14

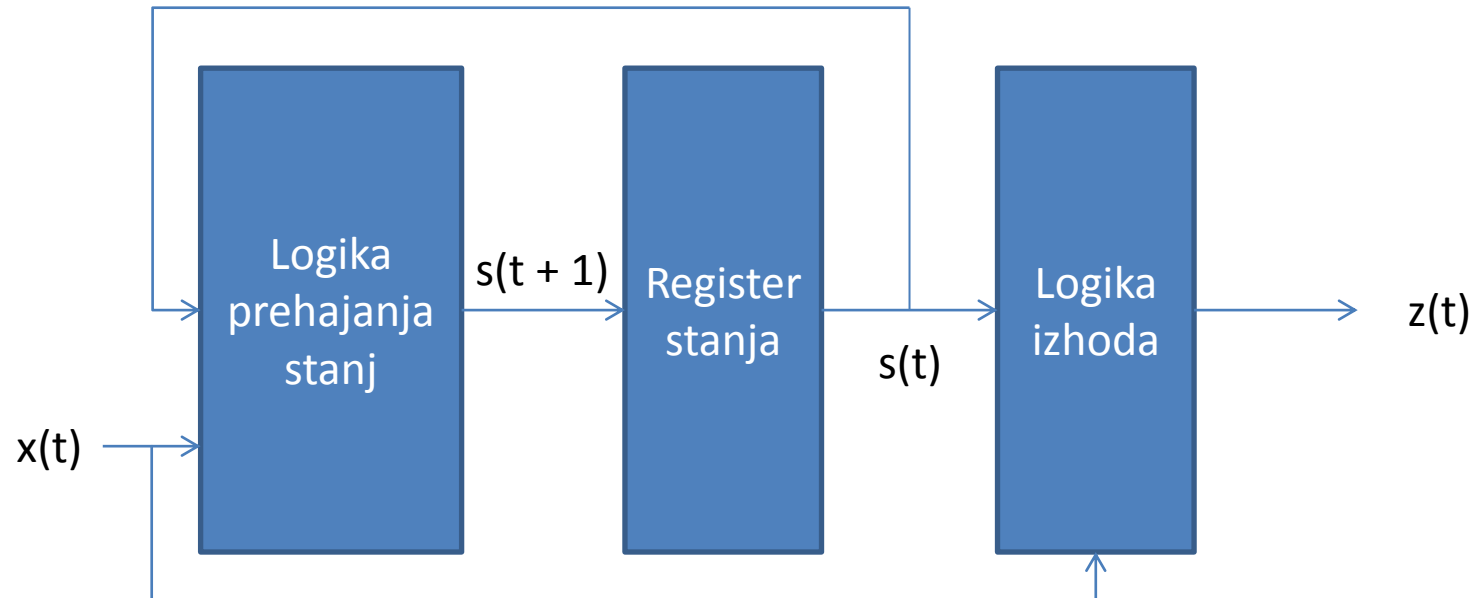
# Moorov avtomat

- Naslednje stanje je odvisno od vhoda in stanja
- Izhod je odvisen od stanja



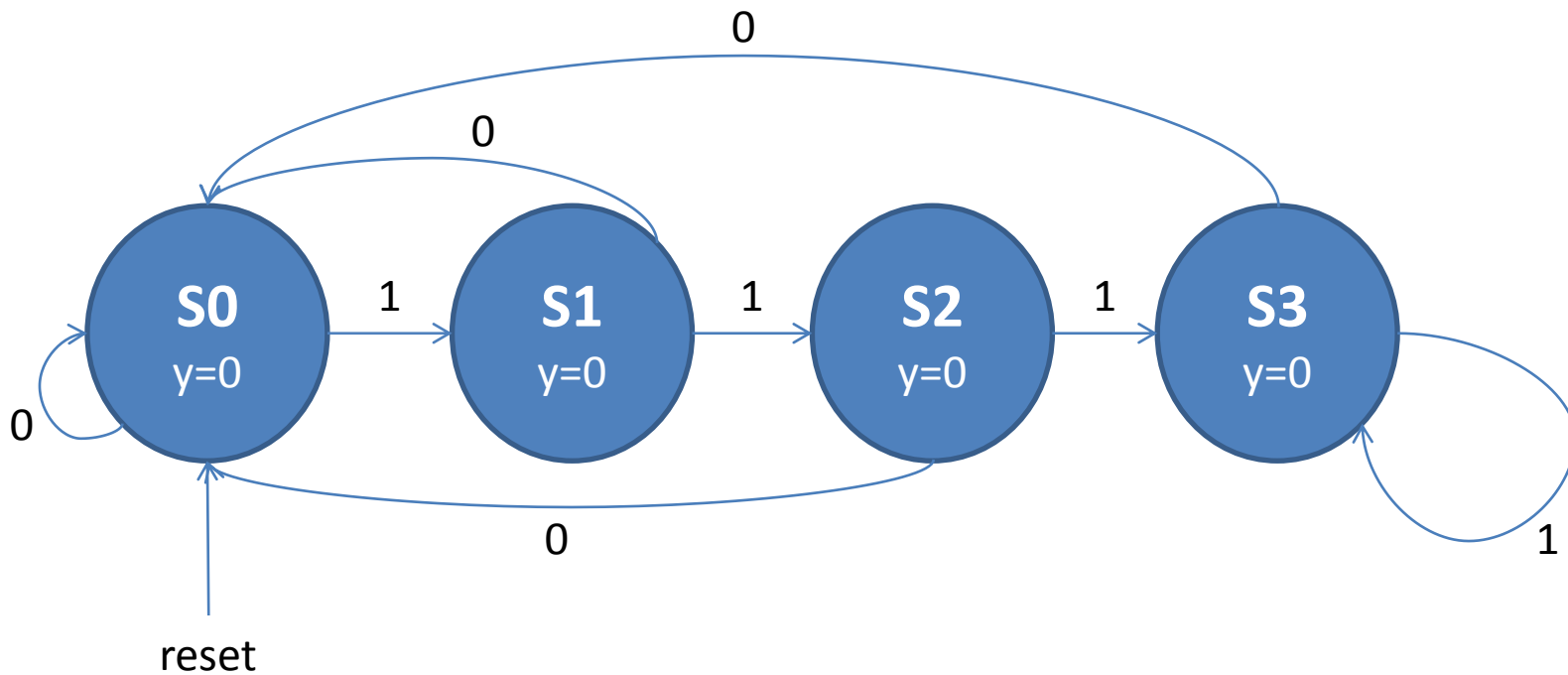
# Mealyev avtomat

- Naslednje stanje je odvisno od vhoda in stanja
- Izhod je odvisen od stanja in vhoda



# Primer

- Moorov avtomat za razpoznavo zaporedja 111
  - ob pravilno razpoznanem zaporedju je na izhod **y** enica



# Realizacija končnega avtomata

- V VHDL opišemo
  - možna stanja
  - delovanje registra stanj
  - logiko prehajanja stanj
  - logiko za izhod

# Primer

```
architecture Behavioral of example_automata is
    type state_type is ( s0 , s1 , s2 , s3 ); -- možna stanja
    signal state , next_state : state_type;
begin
    SYNC_PROC: process ( clk_i ) -- delovanje registra stanj
    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;
end;
```

# 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;
```



# Primer

```
OUTPUT_DECODE: process ( state ) -- logika za izhod
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';
```

```
...
```

# Naloga 1

- Realizirajte avtomat, ki ob negativni fronti vhodnega signala generira pulz v dolžini 1 urine periode

