Primer sekvenčnega vezja

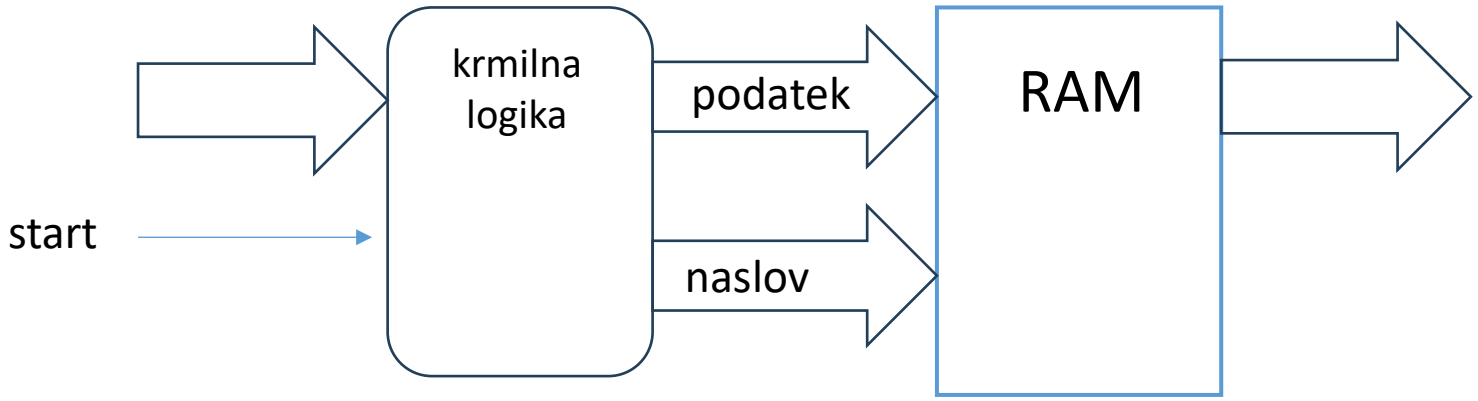
Digitalni elektronski sistemi 2024/25

Andrej Trost

UNIVERZA V LJUBLJANI | FE Fakulteta za elektrotehniko

Zajem bloka podatkov

- vezje za zajemanje podatkov
  - npr. vhodni del logičnega analizatorja ali digitalnega osciloskopa
- podatke na 8-bitnem vodilu ob fronti ure shranjujemo v pomnilnik
  - začetek določa zunanji signal in prožilni pogoj
  - konec, ko napolnimo pomnilnik
- podatke zaporedoma prenesemo iz pomnilnika (FIFO vmesnik)



Kaj imamo na vhodu?

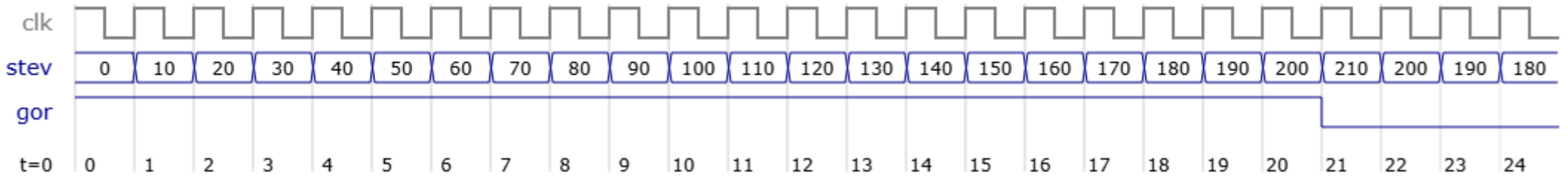
- signal start, ki je aktiven eno ali več period ure
- 8-bitne vrednosti, ki se spreminjajo s časom
- Primer:
  - na vhodu je zaporedje vrednosti 0,10,20...
  - za test naredimo števec navzgor in navzdol

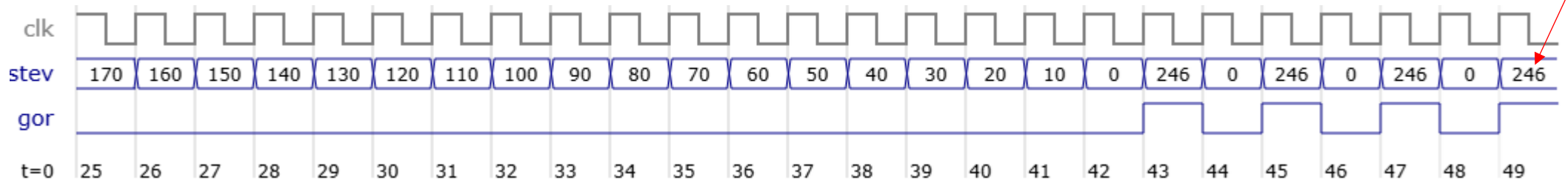
```
entity test
 stev: u8
 gor: u1=1
begin
 if gor=1 then
   stev <= stev + 10
 else
   stev <= stev - 10
 end
end
```

Števec za generiranje podatkov

- števec naj sam menja smer štetja

```
if gor=1 then
  stev <= stev + 10
  if stev>=200 then gor<=0 end
else
  stev <= stev - 10
  if stev=0 then gor<=1 end
end
```





8-bitni stev
0 - 10 = 246

Popravljen števec za generiranje podatkov

```
if gor=1 then
  stev <= stev + 10
  if stev>=200 then gor<=0 end
else
  stev <= stev - 10
  if stev<=10 then gor<=1 end
end
```

Rešitev:

- zamenjaj smer ob prejšnji vrednosti števca (=10)
- ali pa zamenjaj ob trenutni vrednosti naslednjega stanja

```
stev <= n_stev
if n_stev=0 then
```

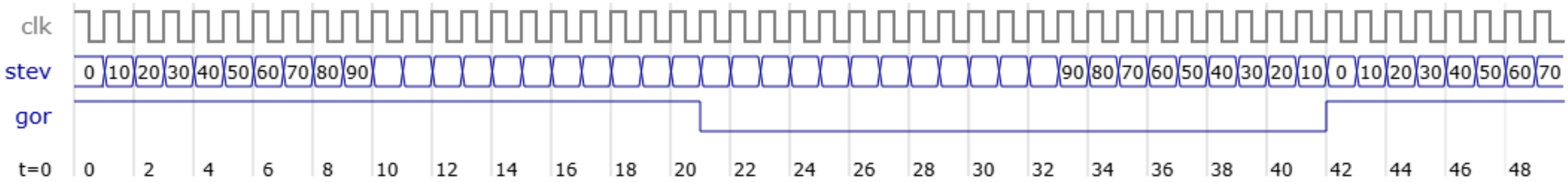
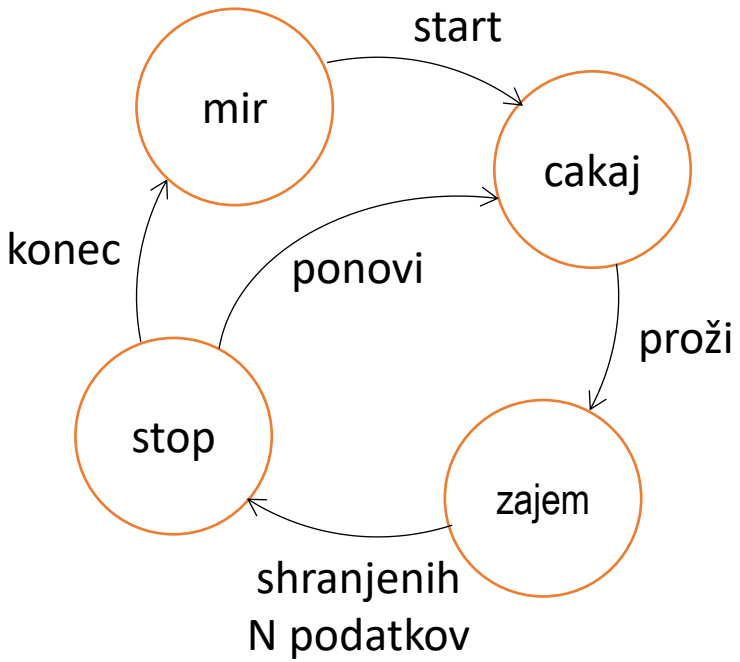


Diagram stanj krmilne logike

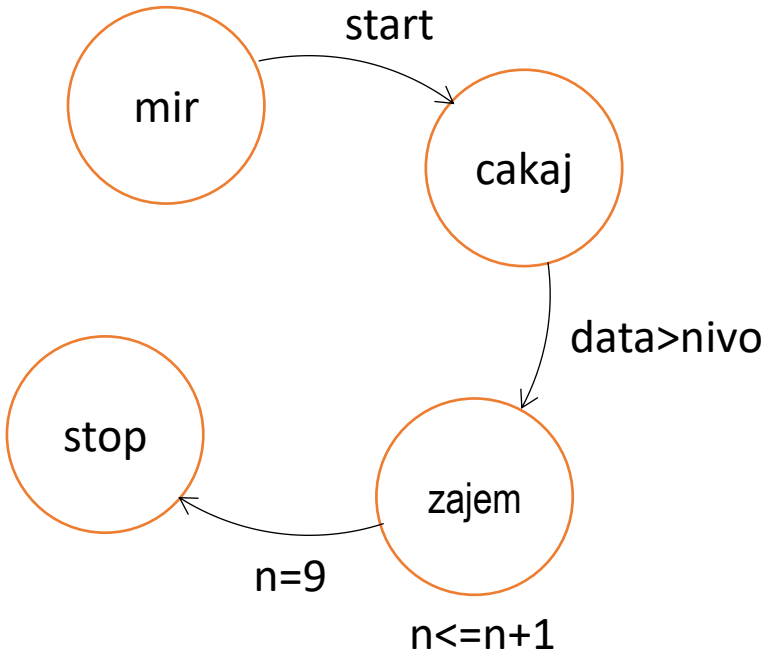


- začni ob signalu start
- čakaj na pogoj za proženje
  - npr. podatek > nivo
- zajem N podatkov
- stop, čakaj da se podatki prenesejo ven iz pomnilnika
  - podatke je potrebno prenesti za prikaz, preden lahko ponovimo zajem
- zaključi (konec) ali ponovi zajem

Načrtovanje sekvenčnega stroja

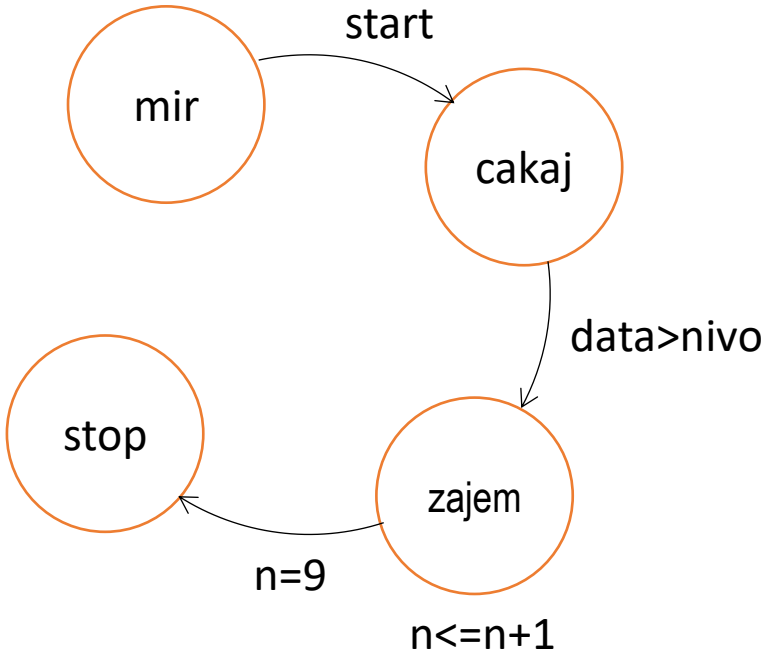
```
if gor=1 then
  stev <= stev + 10
  if stev>=200 then gor<=0 end
else
  stev <= stev - 10
  if stev<=10 then gor<=1 end
end
data=stev
```

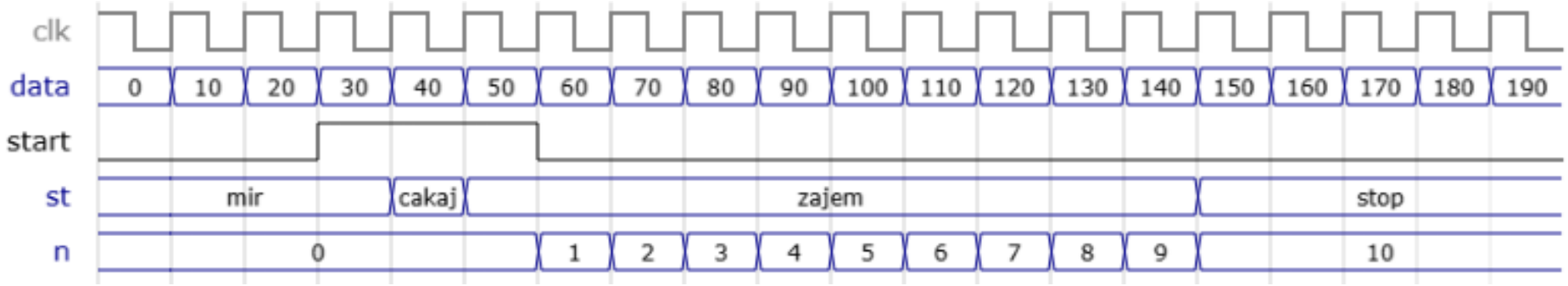
```
 st: (mir, cakaj, zajem, stop)
 nivo: u8 = 20
 n: u4
begin

 if st=mir then
    if start=1 then st<=cakaj end
 elsif st=cakaj then
    if data>nivo then st<=zajem end
 elsif st=zajem then
    n <= n+1
    if n=9 then
       st <= stop
    end
 end
end
```



Simulacija sekvenčnega stroja

```
 nivo: u8 = 20
begin
 if st=mir then
    if start=1 then st<=cakaj end
 elsif st=cakaj then
    if data>nivo then st<=zajem end
 elsif st=zajem then
    n <= n+1
    if n=9 then
       st <= stop
    end
 end
end
```

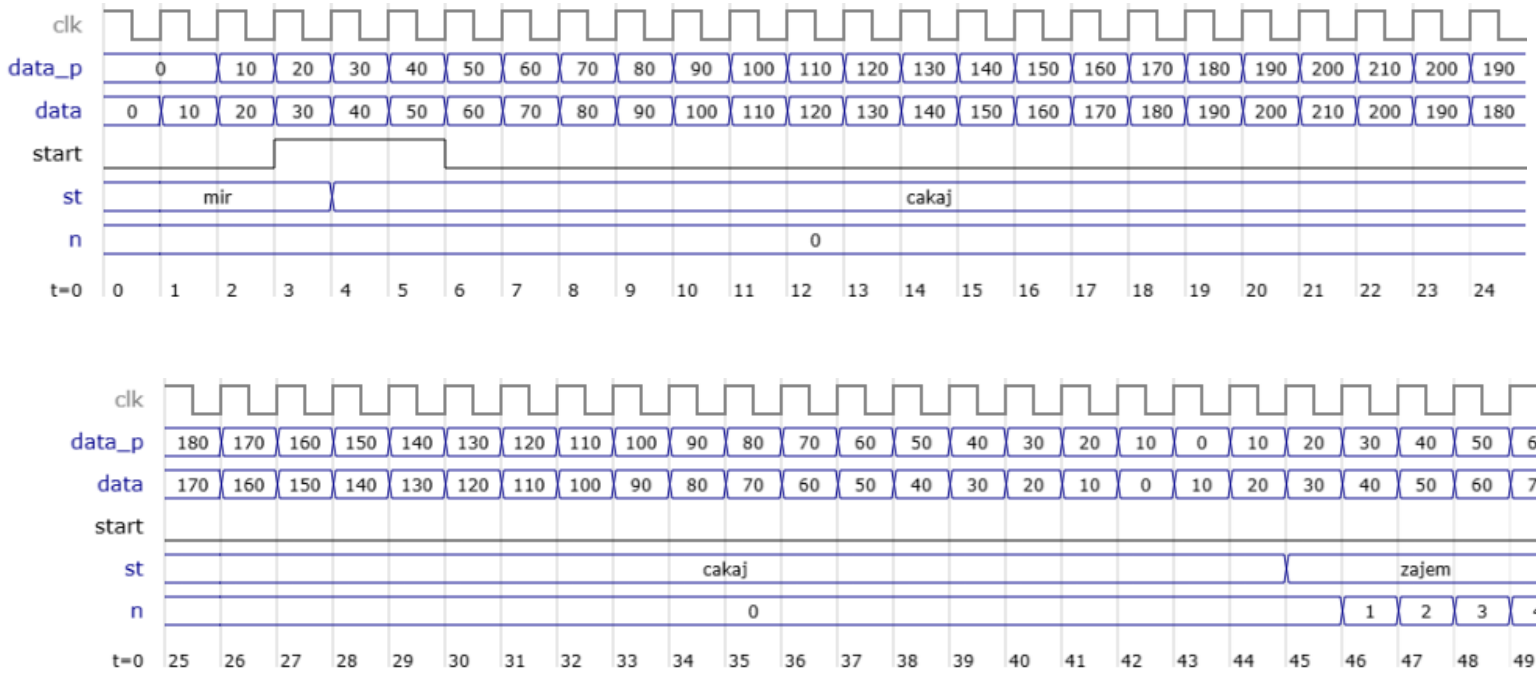




Izboljšave

- kriterij za proženje naj bo prehod čez nivo
  - npr. predhodni podatek < nivo, trenutni podatek >= nivo
- rešitev: shranimo prejšnjo vrednost podatka

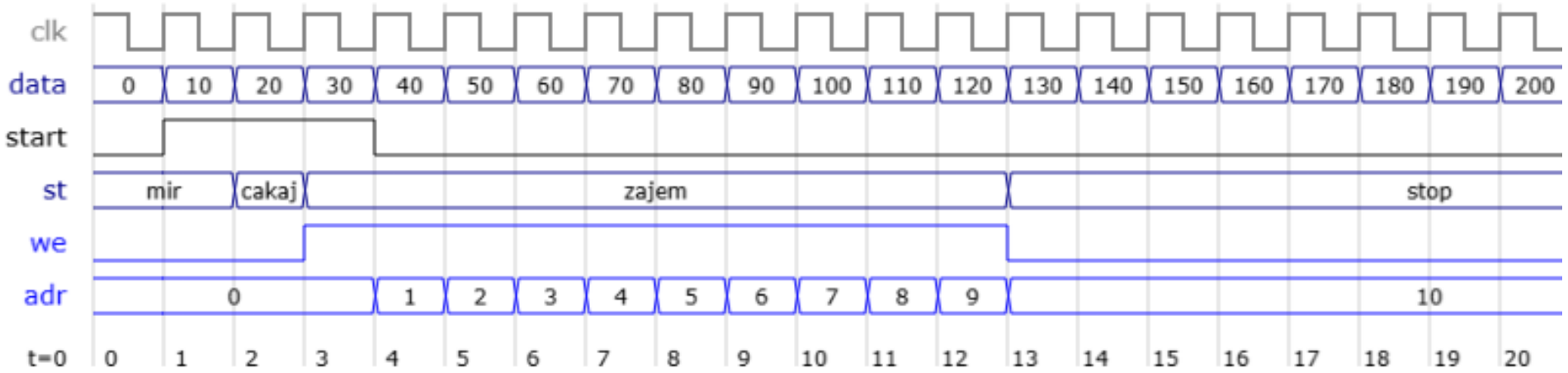
```
 nivo: u8 = 20
begin
 ...
 data_p <= data
 data = stev
 if st=mir then
    if start=1 then st<=cakaj end
 elsif st=cakaj then
    if data>=nivo and data_p<nivo then
       st<=zajem
    end
 elsif st=zajem then
   ...
```



Shranjevanje v pomnilnik

- aktiviramo signal za pisanje (we) in določamo naslov

```
 start: in u1
 adr: out u4
 we: out u1
begin
 ...
 adr=n
 we=1 when st=zajem else 0
```



Testna struktura - VHDL

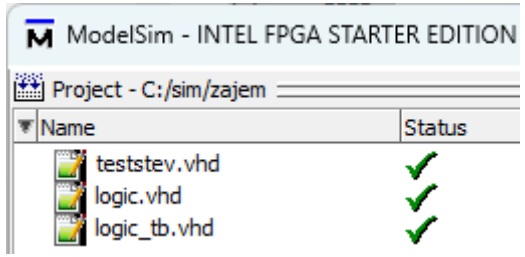
- povežemo testni števec in krmilno logiko

```
entity logic_tb is
end logic_tb;

architecture sim of logic_tb is
 signal clk : std_logic := '1';
 signal data : unsigned(7 downto 0);
 signal start : std_logic := '0';
 signal adr : unsigned(3 downto 0);
 signal we : std_logic;
 constant T : time := 10 ns;
begin

u1: entity work.teststev port map(
          clk => clk,
          data => data
);

u2: entity work.logic port map(
          clk => clk,
          start => start,
          data => data,
          adr => adr,
          we => we
);
```

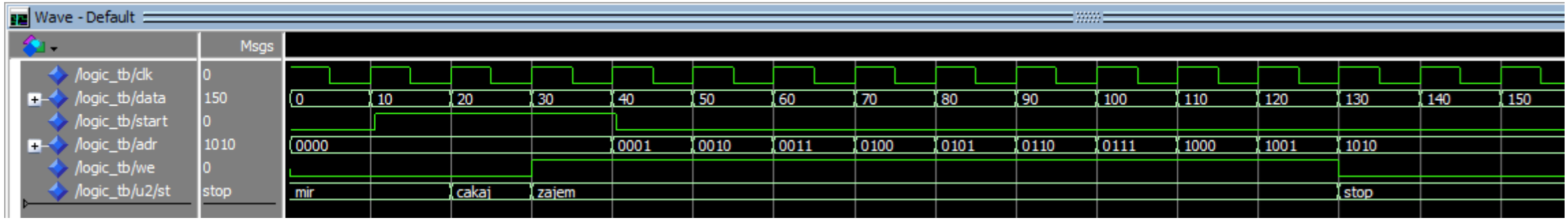
```
-- Clock generator
clk_gen: process
begin
 clk <= '1';  wait for T/2;
 clk <= '0';  wait for T/2;
end process;

stim_proc: process
begin
 wait for T/20;
 start <= '0';
 wait for T;
 start <= '1';
 wait for 3*T;
 start <= '0';
 wait;
end process;
end sim;
```

Testna struktura - simulacija

- ModelSim




```
stim_proc: process
begin
 wait for T/20;
 start <= '0';
 wait for T;
 start <= '1';
 wait for 3*T;
 start <= '0';
 wait;
end process;
end sim;
```

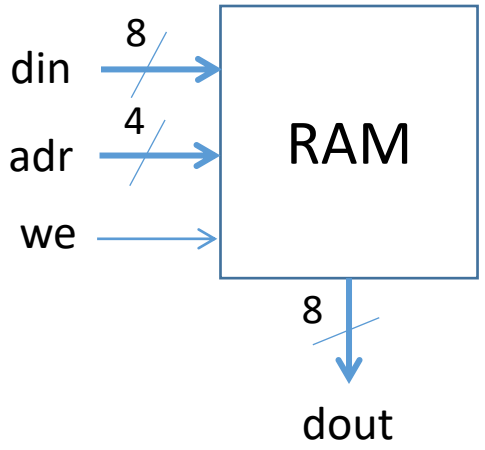


Testna struktura s pomnilnikom

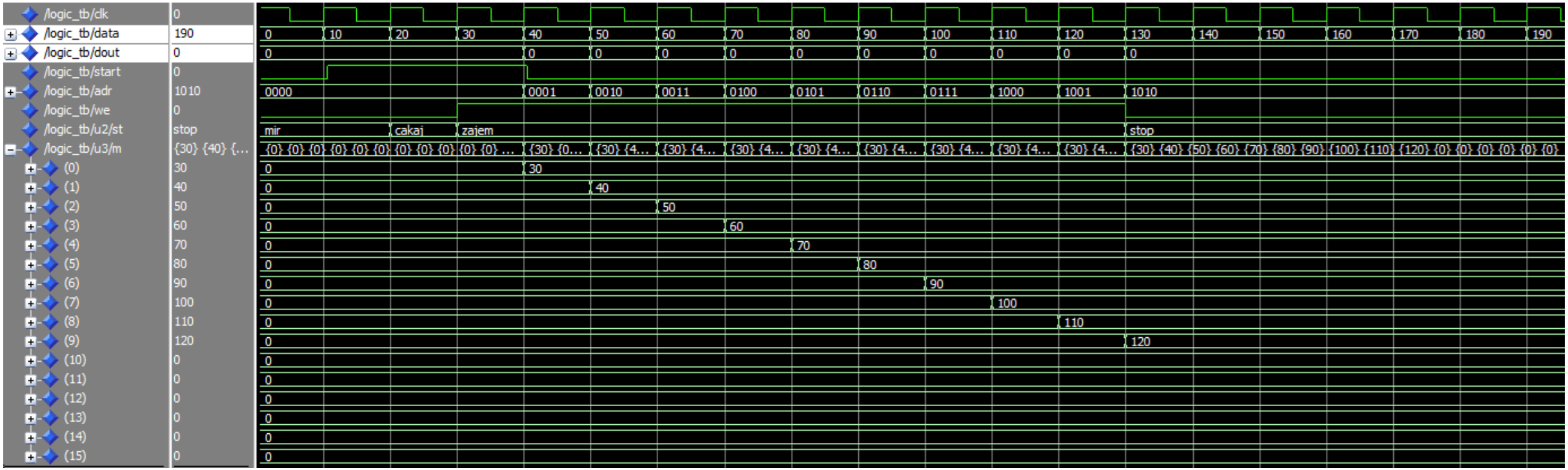
- dodamo še model sinhronega pomnilnika

```
architecture sim of logic_tb is
begin

u1: entity work.teststev port map(
          clk => clk,
          data => data
);

u2: entity work.logic port map(
          clk => clk,
          start => start,
          data => data,
          adr => adr,
          we => we
);

u3: entity work.ram port map(
         clk => clk,
         din => data,
         adr => adr,
         we => we,
         dout => dout
);
```

```
entity ram
  din: in u8
  adr: in u4
  we: in u1
  dout: out u8
  m: 16u8
begin
 dout = m(adr)
 if we=1 then
  m(adr) <= din
 end
end
```



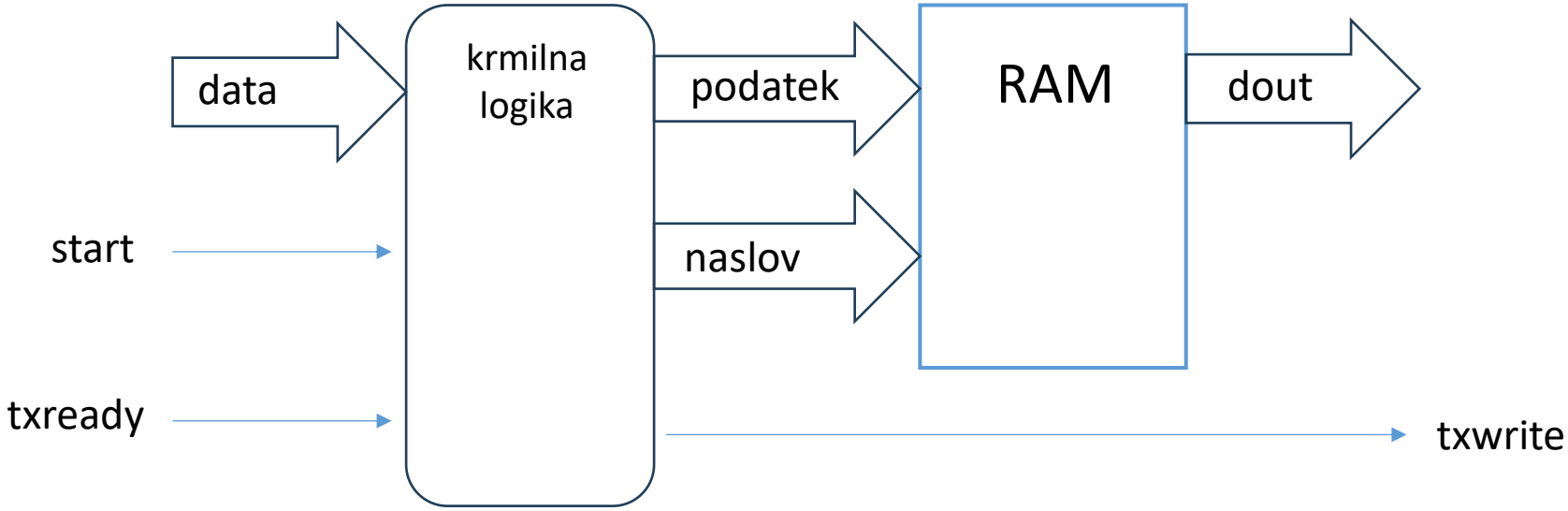
Testna struktura s pomnilnikom



- Kako dosežemo, da se shrani tudi podatek ob proženju (sedaj se začne z naslednjim podatkom)?

Prenos podatkov iz pomnilnika

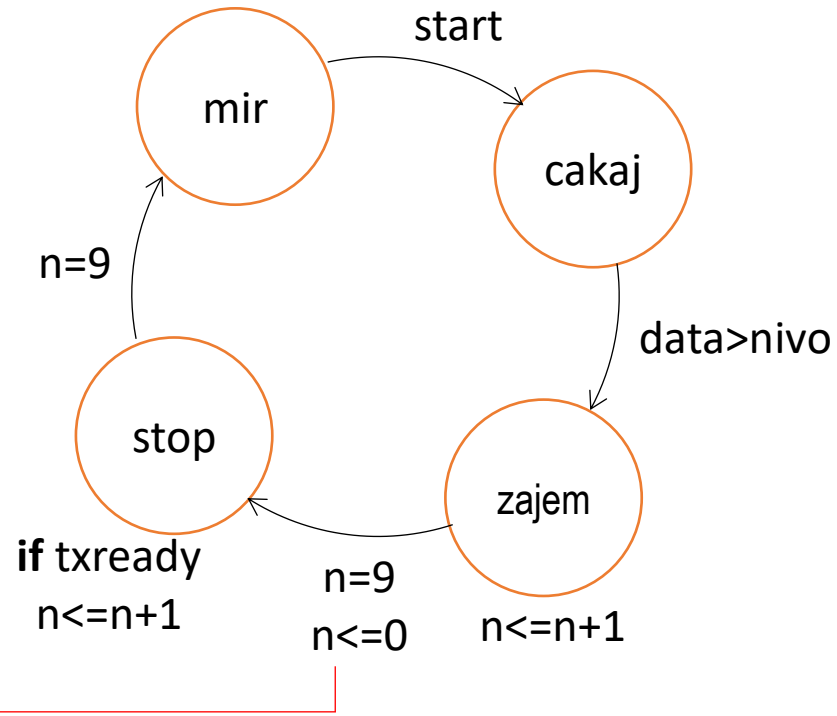
- podatke prenašamo po vrsti, FIFO = Frist In First Out
- dodatni signali
  - vhod txready=1 pomeni, da lahko oddamo podatek
  - podatkovni izhod dout in txwrite=1 ob prenosu podatka



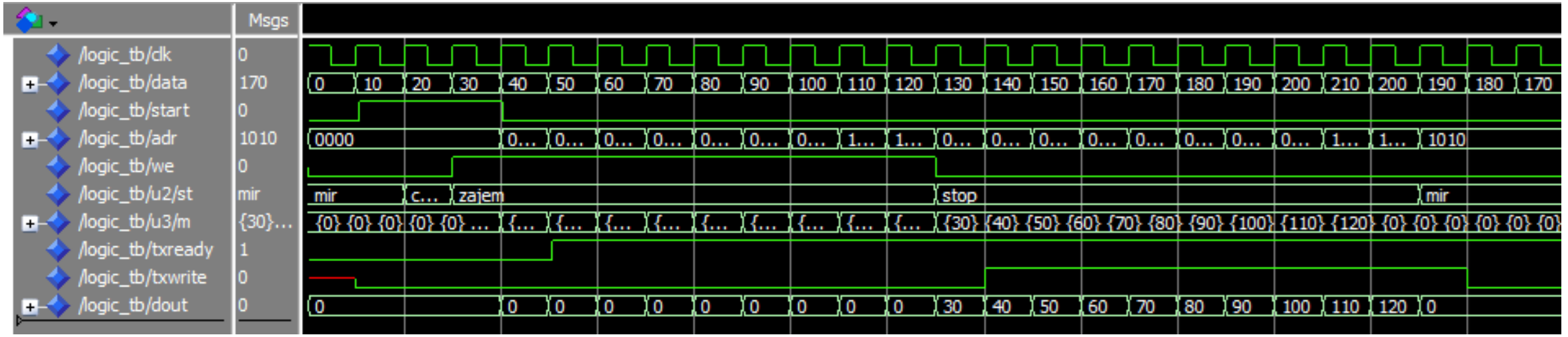
Dopolnimo opis sekvenčnega stroja

```
txwrite <= 0
...
elsif st=zajem then
   n <= n+1
   if n=9 then
      st <= stop
      n <= 0
   end
elsif st=stop then
  if txready=1 then
    txwrite <= 1
    n <= n + 1
    if n=9 then st <= mir end
  end
end
```

- pred prenosom postavimo naslov na 0

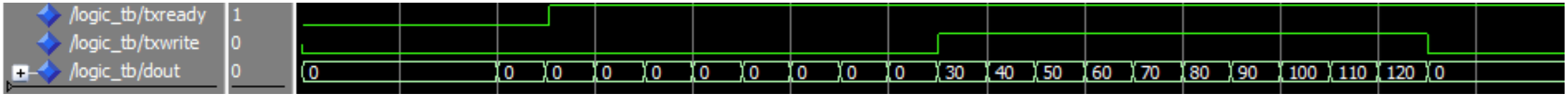


Simulacija prenosa podatkov



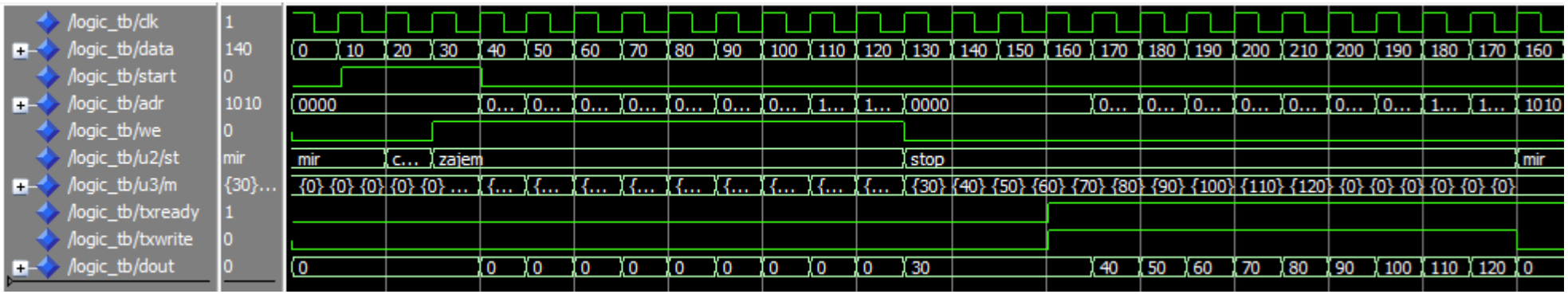
- txwrite mora biti en cikel prej, naredimo kombinacijsko logiko

```
txwrite <= '1' when st = stop and txready='1' else '0';
```

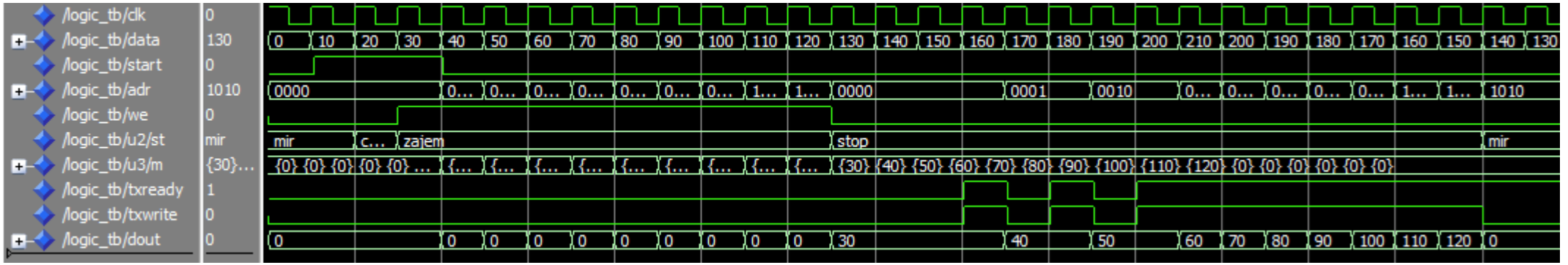


Test: FIFO ni pripravljen na sprejem

- v stop čakamo na txready



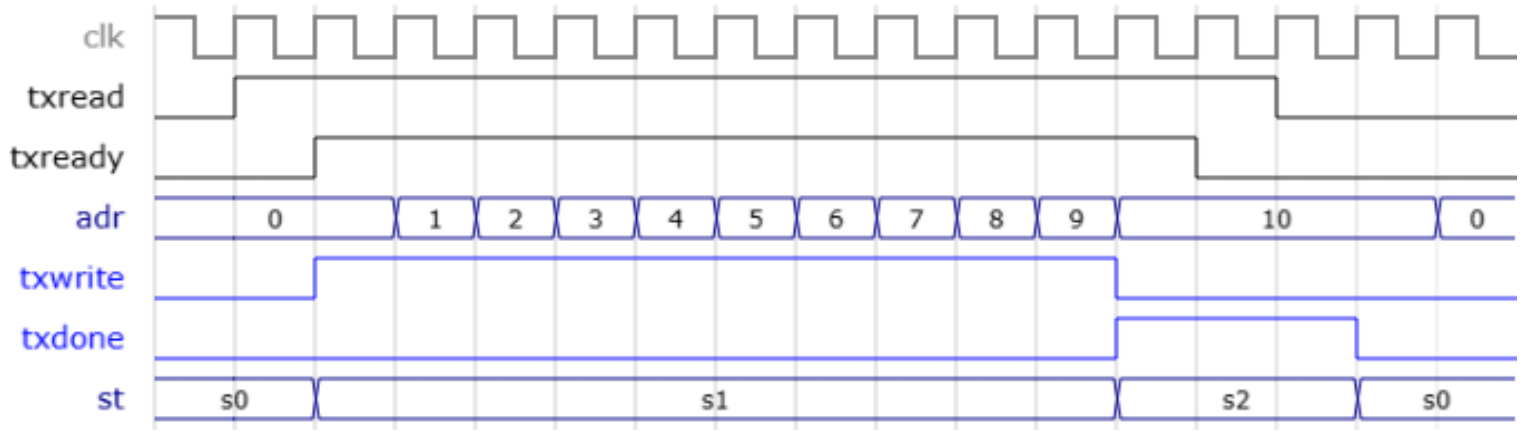
- signal txready uravnava hitrost prenosa v FIFO



*Prenos podatkov z drugo uro

```
entity sync
 txread: in u1;
 txready: in u1;
 adr: out u4;
 txwrite: out u1;
 txdone: out u1;
 st: (s0,s1,s2)
begin
 if st=s0 then
   adr <= 0
   if txread=1 then st <= s1 end
  elsif st=s1 then
   if txready=1 then
     adr <= adr + 1
     if adr=9 then
       st<=s2
       txdone<=1
     end
   end
  else
   if txread=0 then
     txdone<=0
     st<=s0
   end
  end
  txwrite = '1' when st=s1 and txready='1' else '0';
```

- Primer: FIFO ima počasnejšo uro za sprejem podatkov
  - potrebno je zagotoviti sinhronizacijo krmilnih signalov
  - vhod txread=1 za začetek prenosa
    izhod txdone=1 sporoči zaključek prenosa



*Simulacija

- generator ure clk in počasnejše ure clk2
- naslov pomnilnika določata obe sekvenčni vezji
  - adr **<=** wadr **when** txread**=**'0' **else** txadr**;**
- RAM poskrbi za ločitev delov vezja, ki uporabljajo različne ure