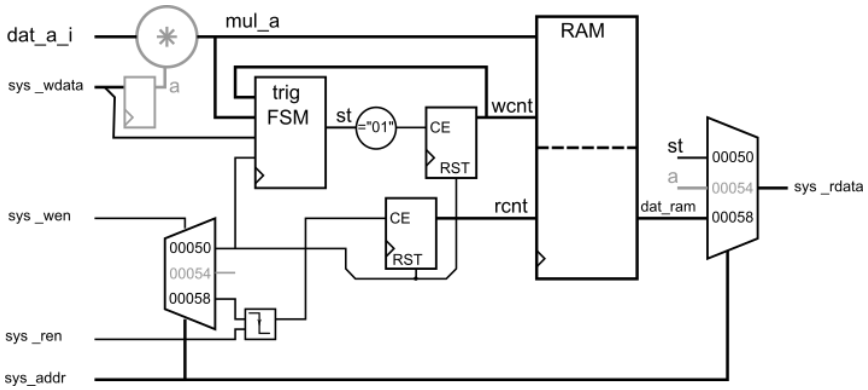
5. in 6. vaja: vzorčenje signalov



5.1 Procesna komponenta s pomnilnikom

V projektu redpitaya-proc bomo dodali pomnilnik za vzorčenje signalov. Shranjevanje signalov naj se izvede, ko procesor nastavi stanje z vpisom vrednosti 1 na naslov 50.

1. V komponenti **proc** deklariraj pomnilnik velikosti 256 x 16 bitov in 16-bitni predznačen vektor **dat_ram**. Deklariraj še dvobitni signal za stanje (**st**) in 8-bitna naslovna števca (**wcnt**, **rcnt**) ter dodaj v vezje opis pomnilnika, ki shranjuje 16-bitne vrednosti iz množilnika in jih oddaja na **dat_ram**.

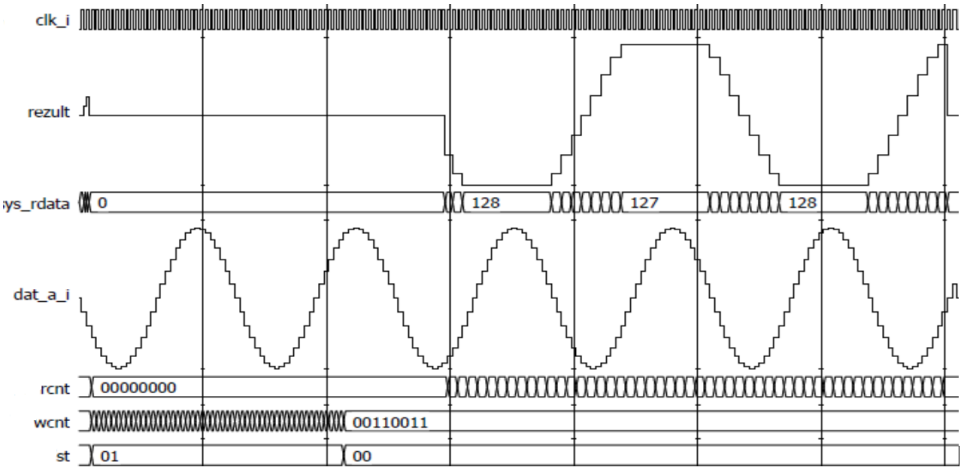
```
pram: process(clk_i)
begin
if rising_edge(clk_i) then
 if st=1 then
  ram(to_integer(wcnt))<=mul_a(19 downto 4);
 end if;
 dat_ram <= ram(to_integer(rcnt));
end if;
end process;
```

2. V sinhroni proces dodaj stavek, ki ob vpisu iz vodila na naslov 00050 nastavi stanje **st** (spodnja 2 bita) in resetira oba naslovna števca. Naslovni števec **wcnt** naj se povečuje ob uri in pogoju **st**="01".

3. Shranjene vrednosti bomo iz pomnilnika zaporedno prenašali v procesor. Uporabimo en prost naslov (npr. 00058) na katerega naj bo vezan notranji signal **dat_ram**. Števec za branje se poveča po vsakem branju iz naslova 00058 (pogoj je zadnja fronta signala **sys_ren**).

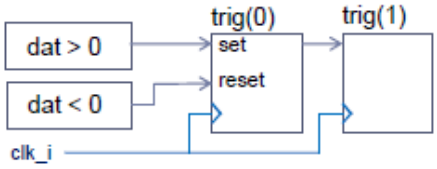
4. Preizkusi delovanje s simulacijo. V testni strukturi nastavi **st** na 1 (naslov 00000050), počakaj 50 ciklov, **st** na 0, počakaj še nekaj ciklov, nato pa v zanki beri pomnilnik, tako da se spreminja signal **ren_i**, npr.:

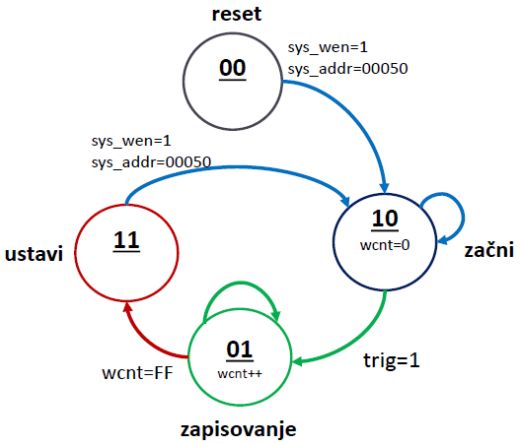
```
for i in 0 to 50 loop
  addr_i  <= x"00000058"; ren_i <= '1'; wait for T;
  ren_i <= '0'; wait for T;
end loop;
```



6.1 Osciloskop

Dodali bomo logiko za proženje vzorčenja signalov, kot ga ima osciloskop. Shranjevanje v pomnilnik naj se sproži ob prehodu signala čez vrednost 0, kar ustreza naraščajoči fronti ob nastavljenem nivoju proženja 0 V.





1. Deklariraj 2-bitni signal **trig**, ki bo predstavljal pomikalni register za zaznavanje fronte. Najnižji bit naj se sinhrono z uro postavi na 1, ko je podatkovni signal večji 0, kadar je manjši od 0 pa naj se postavi nazaj na 0. Najvišji bit dobi ob uri vrednost najnižjega, kot prikazuje shema.
2. Opiši prožilni avtomat, ki spreminja stanja (**st**) ob uri in pogojih prikazanih na diagramu stanj. Komponento za vzorčenje signalov z vpisom na prvi naslov aktiviramo, nato pa naj po izpolnjenem pogoju za proženje zapisuje podatke. Ko je pomnilnik poln (**wcnt** = 255), naj gre v stanje, kjer čaka ponovno aktivacijo.
3. Preveri delovanje prožilnega avtomata na simulaciji s testno strukturo. Pri branju podatkov iz naslova 00050, naj predstavljata spodnja dva bita stanje avtomata. Z branjem iz tega naslova, bo procesor lahko preveril, ali se je proženje izvedlo.

6.2 Preizkus na Red Pitayi

Delovanje komponente preizkusi na Red Pitayi z nastavljanjem registrov in s programom v jeziku C.

4. Odpri program Bitwise SSH Client in se poveži z Red Pitayo na naslovu: **192.168.1.15**. Z orodjem SFTP prenesi nastavitveno datoteko **red_pitaya_top.bit**, ki se nahaja v `project\redpitaya.runs\impl_1` nato pa v terminalu nastavi datoteko za spletno aplikacijo:

   ```
   ./nastavi.sh red_pitaya_top.bit
   ```

5. V brskalniku napiši naslov **192.168.1.15** in odpri aplikacijo Oscilloscope & Signal Generator. Nastavi generiranje signala na OUT1. V terminalu krmili ustrezne registre (branje ID in nastavljanje registra za skaliranje signala):

   ```
   monitor 0x40300050
   monitor 0x40300054 10
   ```

6. Aktiviraj vzorčenje z vpisom na naslov `0x40300050`, nato pa preberi iz tega naslova in preveri ali se je proženje izvedlo. Za branje podatkov uporabi aplikacijo vzorci.c, ki jo prevedi in izvedi v terminalu:

   ```
   gcc vzorci.c -o vzorci
   ./vzorci
   ```

7. Izhodne podatke lahko zapišemo v datoteko in uporabimo za prikaz grafikona: `vzorci > graf.csv` (uporabi npr. https://www.csvplot.com/)

Razmisli

- Kako bi logiki zaznavanja proženja dodal histerezo, da bi bila manj občutljiva na motnje v signalu?