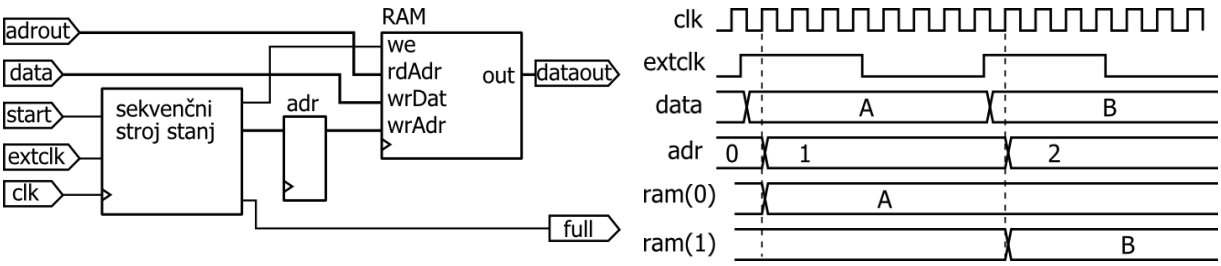
V2. Zajem logičnih signalov

Naredili bomo sekvenčno vezje za zajem in shranjevanje logičnih signalov v pomnilnik. Vezje začne ob signalu **start** zajemati podatke iz vodila **data** in jih shranjuje v pomnilnik. Ko je pomnilnik poln, postavi zastavico **full.** Podatke beremo tako, da nastavljamo naslov **adrout**.



Vhodni podatki se spreminjajo z zunanjo uro **extclk**. V sekvenčnem stroju stanj naj bo logika, ki zazna fronto zunanje ure in določi časovni trenutek prenašanja podatka v pomnilnik.

Opis vezja

V prvem delu opisa vezja definirajmo celoštevilski parameter AW, ki določa velikost notranjega pomnilnika (deklarirali bomo 2^{AW} pomnilnih celic):

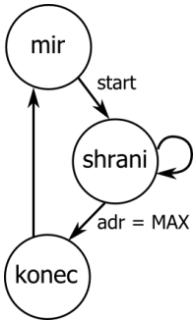
```
generic (AW: integer := 4);
port ( adrout: in unsigned(AW-1 downto 0);
       data : in std_logic_vector (7 downto 0);
       start, extclk, clk : in std_logic;
       dataout: out std_logic_vector (7 downto 0);
       full : out std_logic);
```

V opisu arhitekture deklariramo zbirko za pomnilnik, signale za naslov (**adr**) in stanja:

```
architecture Behavioral of zajem is
   type mem is array(0 to 2**AW - 1) of std_logic_vector(7 downto 0);
   signal ram: mem;
   signal adr: unsigned(AW-1 downto 0);
   type stanja is (mir, ...
```

Delovanje vezja krmili sekvenčni stroj s tremi stanji:

- **mir**, mirovno stanje v katerem čakamo na signal **start**,
- **shrani**, ob zaznani fronti signala **extclk** shranimo podatek v **ram** in povečamo naslov,
- **konec**, signalizira poln pomnilnik (**full**) in se vrne v mirovno stanje



Zapis podatka **data** v pomnilnik na naslovu **adr** opisuje stavek:

```
ram(to_integer(adr)) <= data;
```

Naredi opis vezja, preveri pravilnost opisa s sintezo vezja in na RTL shemi ter s simulacijo !