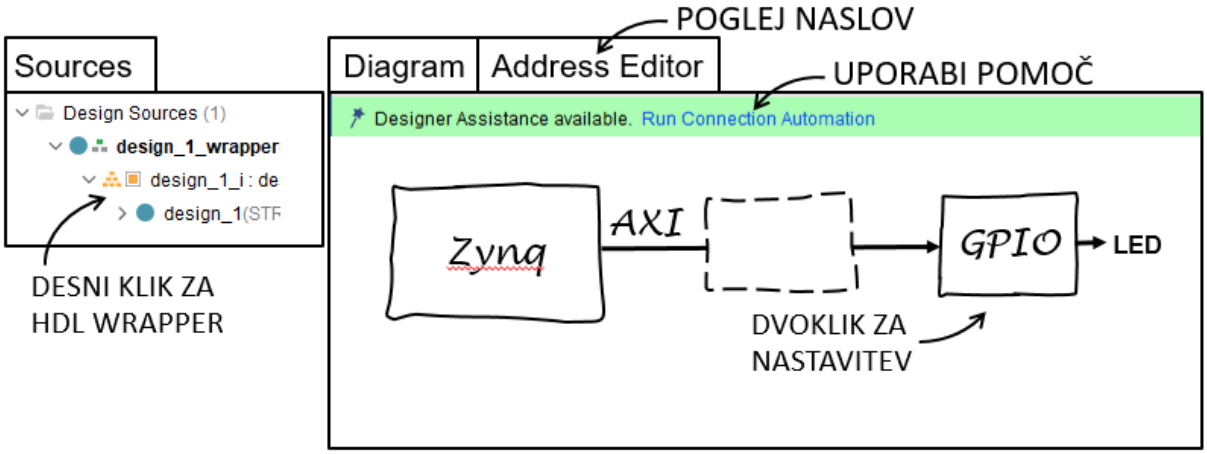
1. in 2. vaja: vezje z vmesnikom AXI

Naredili bomo računsko enoto z vmesnikom AXI kot komponento IP. Vezje naj računa povprečje vseh vrednosti zapisanih v register **r0**.

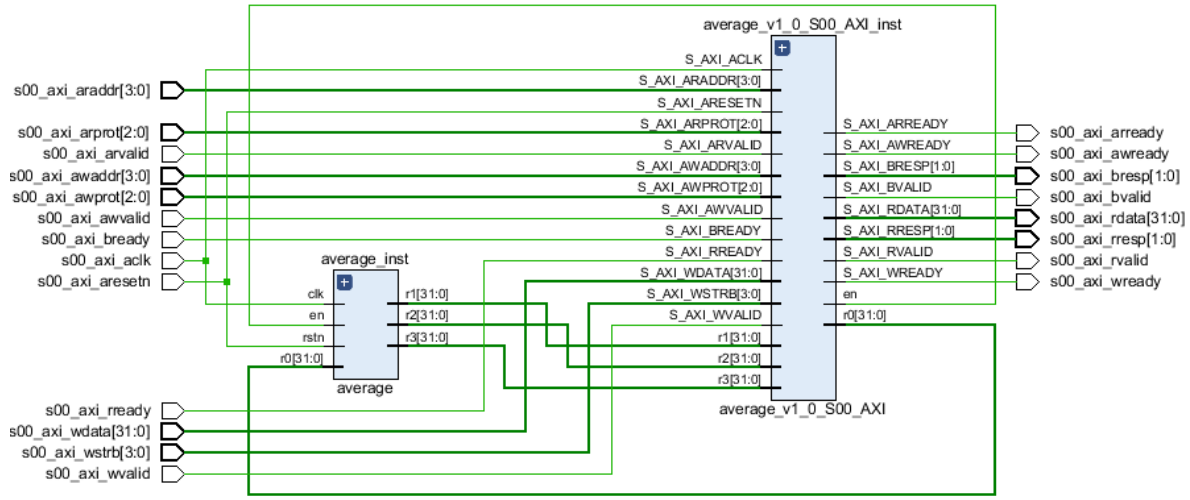
1.1 Vivado: blokovni diagram https://lniv.fe.uni-lj.si/xilinx/vivado-blok.htm

Izdelaj projekt v orodju Vivado, izberi ploščo z vezjem Zynq in sestavi blokovni diagram sistema s splošnim vmesnikom **axi_gpio**.



1.2 Računska enota

Odpri vzorec projekta z opisom vmesnika AXI-Lite (**average_v1_0_S00_AXI**), komponente za računanje povprečja (**average**) in vezja, ki vključuje in povezuje obe komponenti.



1. Preglej VHDL opis vmesnika. Poišči, kam je vezan izhod **r0** in dodaj stavke, ki ob vpisu nove vrednosti naredijo impulz na izhodni signal **en**.
2. Odstrani registre **slv_reg1..3** in naredi povezavo na vhode **r1..r3**, da bo procesor bral vrednosti iz teh vhodov.
3. Naredi opis komponente za izračun povprečja: deklariraj 32-bitno vsoto in 16-bitni signal za število podatkov. Ob vsakem **en**='1' naj se **r0** prišteje k vsoti in poveča število sprejetih podatkov. Izhod **r1** predstavlja vsoto, **r2** število podatkov, **r3** pa povprečje (kvocient).
4. Izvedi sintezo, preglej shemo (*Elaborated Design*) in poročilo o uporabi FPGA (LUT, FF).

2.1 Simulacija računske enote

Simulacijo naredi s testno strukturo **average_tb**, ki posnema pisanje in branje vodila AXI-Lite. Sekvenco signalov nastavljamo s procedurama **write32** in **read32**. Ob klicu procedure za pisanje vnesemo naslov in podatek, ob branju pa le naslov perifernega registra. Naslovi registrov **r0**..**r3** so: 0, 4, 8 in 12.

```
stim_proc: process
   procedure write32(adr: integer; data: integer) is  -- postopek pisanja na AXI-lite
   begin
     wait until rising_edge(s_axi_aclk);
        s_axi_awaddr <= std_logic_vector(to_unsigned(adr, 4));  -- nastavi naslov in podatek
        s_axi_wdata <= std_logic_vector(to_unsigned(data, 32));
     ...
   end write32;

   begin
      wait for 3*period; -- cakaj na reset
      write32(0, 2);     -- vpis v register r0
      write32(0, 3);
      ...
```

1. Simuliraj vezje računske enote in preglej signale na časovnem diagramu in prebrane vrednosti v konzoli. Popravi testno strukturo, da bo izvajala simulacijo z izbranimi podatki.

2.2 Pakiranje komponente IP

Računsko enoto bomo shranili kot komponento IP, ki jo lahko dodamo v blokovni diagram. Opis IP komponente vključuje datoteko XML in pripadajoče datoteke z modelom in grafično podobo vezja.

2. V Vivadu izvedi korake pakiranja računske enote kot komponente IP: Tools, *Create and Package New IP,* nato: *Package your current project*. Po pregledu korakov, zaključi z *Review and Package.* Uredi nastavitve, da bo Vivado naredil arhiv in izvedi *Package IP*.
3. Prestavi se v projektno mapo, poišči in oddaj v spletno učilnico datoteko z arhivom opisa komponente (xilinx.com_user_average_v1_0_1.0.zip).

2.3 Blokovni diagram in prevajanje sistema

Na projekt z blokovnim diagramom bomo poleg splošnega vmesnika dodali še računsko enoto.

4. Odpri prvi projekt z blokovnim diagramom in v nastavitvah projekta (*Settings*) dodaj mapo z opisom nove komponente (*Project Settings, IP, Repository*). Dodaj komponento na diagram in jo poveži (uporabi *Design Assistance*). Preglej naslove dodeljene novi komponenti.
5. Prevedi digitalni sistem. Preglej poročilo o uporabi in zakasnitvah vezja.
6. Preizkusi delovanje na Red Pitayi. S programom Bitvise SSH prenesi prevedeno datoteko (*.bit), ki se nahaja v podmapi .runs/impl1 na razvojno ploščo in izvajaj ukaze v terminalu:

`cat ime.bit > /dev/xdevcfg`	prenos datoteke v FPGA (programiranje)
`monitor 0x43C00000 20`	vpis podatka v register r0
`monitor 0x43C0000C`	branje iz registra r3

Razmisli

- Zakaj je potreben korak pakiranja komponente IP?
  Kako se določi periferni naslov?
- Kakšne so omejitve vezja za računanje povprečja in kaj bi lahko spremenili?
  Kaj je potrebno narediti, da bi vezje delovalo pri višji frekvenci?