5. Visokonivojska sinteza

Pri vaji bomo:

- spoznali orodje **Vivado HLS**, ki izvaja visokonivojsko sintezo vezja iz jezika C++ v VHDL ali Verilog
- naredili komponento vezja za prikaz olimpijskih krogov
- preizkusili delovanje na sistemu

5.1 Projekt Vivado HLS

- Zaženi program Vivado HLS in izberi **Create New Project**. Določi ime projekta, npr. okrogi in lokacijo na disku. Prenesi v svojo mapo in odpakiraj izvorne datoteke v jeziku C++: cpp_okrogi.zip
- V naslednjem oknu določi ime glavne funkcije (**okrogi**) in dodaj datoteki *okrogi.cpp* in *okrogi.h*. Funkcija sprejme 32-bitni **din,** ki vsebuje koordinato točke (x, y) in izračuna, ali se nahaja na eni izmed petih krožnic ali ne. Če se nahaja na krožnici oz. bolj natančno na kolobarju med krožnicama s polmerom 14 in 16, vrne izhod z določeno barvo točk, sicer pa je izhodna vrednost enaka vhodni.
- V tretjem oknu dodaj testno datoteko *okrogi_test.cpp*, nato pa določi periodo ure (20 ns) in izberi vrsto vezja (izbrali bomo razvojno ploščo ZedBoard).

```
int okrogi(int din)
{
  ...
  x = (din >> 8) & 0xFF;  // dekodiranje vhoda din, format: 0x80 yy xx rgb
  y = (din >> 16) & 0xFF;

  loop0: for (i=0; i<5; i++) { // zanka za računanje 5 kolobarjev
   if (i%2==0) b = 28; else b = 16;         // menjava y koordinat

   tmp = (x-a)*(x-a) + (y-b)*(y-b);

   // če se točka nahaja znotraj kolobarja (14 < r < 16), postavi rgb na color[i]
   if ((tmp < 16*16) && (tmp > 14*14)) rgb = color[i];

   a += 18;
  }

  if (rgb==0) dout = din;  // sestavi izhod
  else dout = (din & 0xFFFFFF00) | rgb;

  return dout;
}
```

5.2 Simulacija in sinteza vezja

- Za izvedbo simulacije funkcije izberi **Project > Run C Simulation**. Ko se prevede in izvede testna datoteka, se prikaže izhodna slika v tekstovni obliki.
- Dodaj v funkcijo okrogi() še dva vhodna parametra: celoštevilska **dx** in **dy,** ki naj povzročita premik izrisanih krogov (dodaj njuno vrednost k notranji spremenljivki **a** oz. **b**). Preizkusi delovanje premika na simulaciji.
- Naredi sintezo vezja (**Solution, Run C Synthesis**) in preglej rezultate sinteze: zmogljivost vezja merjeno v zakasnitvah (latenca, interval ponovitve), zasedenost vezja (število blokov DSP48E, registrov in vpoglednih tabel) ter zapiši podatke v tabelo:

latenca	interval	DSP48E	registri (FF)	tabele(LUT)

- Oglej si še ostale podatke po sintezi, npr. število in vrsto priključkov vezja. Odpri analizo vezja (**Solution, Open Analysis Perspective**) in časovni potek izvajanja operacij po posameznih ciklih.

5.3 Optimizacija sintetiziranega vezja

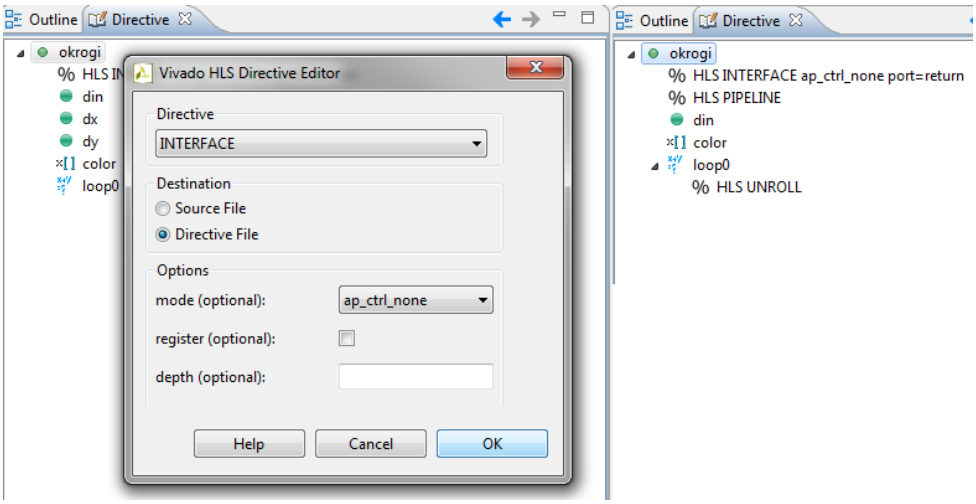
- V Vivadu HLS vplivamo na rezultat sinteze vezja z direktivami, ki jih dodamo funkcijam, strukturam in spremenljivkam. Na velikost vezja zelo vpliva tudi pravilna izbira podatkovnih tipov spremenljivk.

  Najprej bomo funkciji dodali direktivo, ki pove, da na priključkih vezja ne potrebujemo krmilnih signalov. V zavihku **Directive** naredi desni klik na ime funkcije **okrogi,** izberi **Insert Directive...** in HLS INTERFACE: ap_ctrl_none (glej sliko).

- Spremeni podatkovni tip parametrov **dx** in **dy** v 8-bitno celo število (**i8**), ki je definiran v *okrogi.h,* ponovi sintezo ter preglej in zapiši rezultate v tabelo. Kaj se spremeni?

```
#define i8 ap_uint<8>
int okrogi(int din, i8 dx, i8 dy);
```

latenca	interval	DSP48E	registri (FF)	tabele(LUT)



- Dodaj zanki **loop0** direktivo HLS UNROLL, ki določa, da naj jo program razvije in naredi sintezo. Nazadnje pa dodaj fukciji **okrogi** še direktivo HLS PIPELINE. Zapiši rezultate sintez v tabelo:

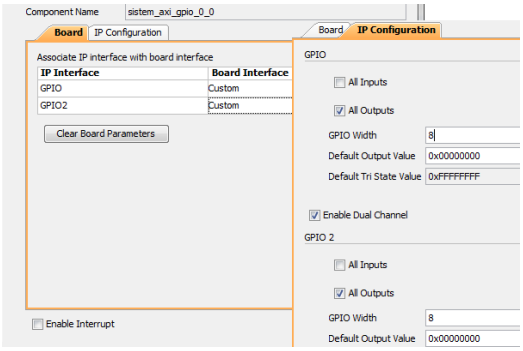
latenca	interval	DSP48E	registri (FF)	tabele(LUT)

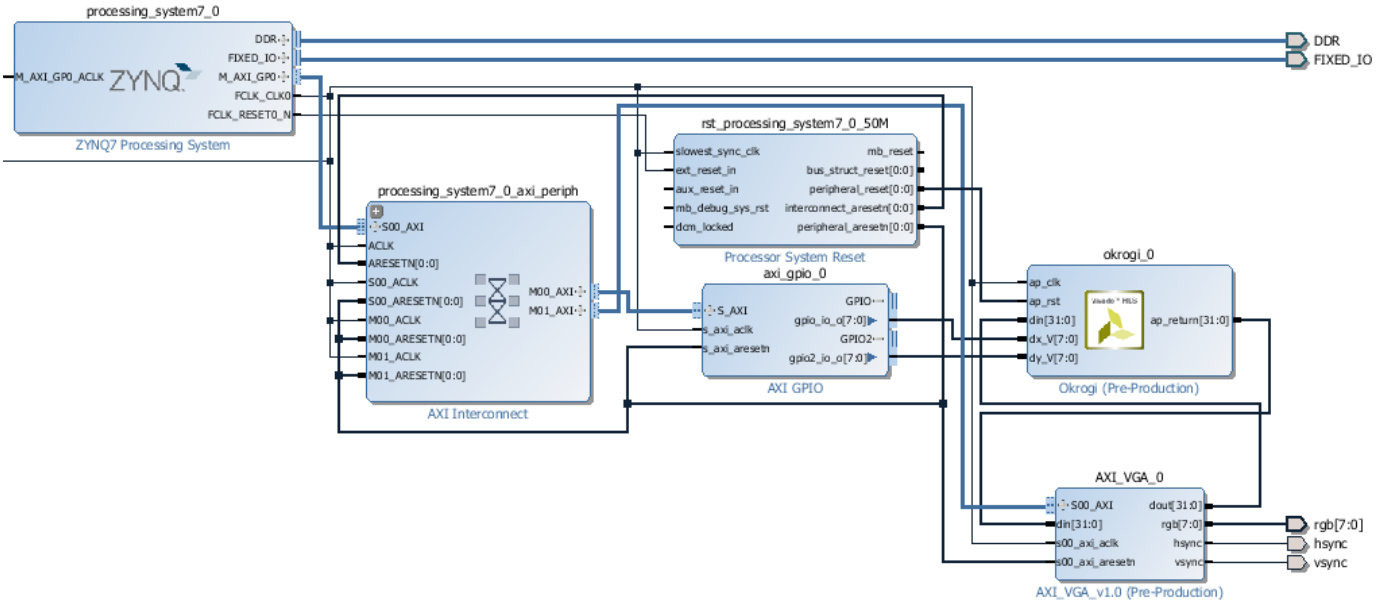
latenca	interval	DSP48E	registri (FF)	tabele(LUT)

- izvozi rezultat sinteze (**Solution, Export RTL**) v obliki komponente IP. Vse datoteke komponente se nahajajo v podmapi: solution1\impl\ip

5.4 Integracija v sistem in preizkus

- odpri digiralni sistem za prikazovanje slike iz prejšnje vaje in dopolni blokovni diagram. Dodaj v katalog IP mapo z novo komponento (**IP Settings, Repository Manager**) in dodaj simbol komponente v blokovni načrt (**Add IP,** okrog) ter poveži signale:
    - ap_clk na uro FCLK_CLK0
    - ap_rst na peripheral_reset izhod komponente **Processor System Reset**
    - din na dout komponente **AXI_VGA**
    - ap_return na din komponente **AXI_VGA**
    - dx in dy na izhode **AXI GPIO**, kjer izbriši povezava na tipke in LED ter nastavi komponento: Interface: Custom, IP Configuration, All Outputs, Width 8 tako, da bo imela dva 8-bitna izhoda (glej sliko).





- Preveri blokovno shemo (**Tools > Validate Design**). Naredi sintezo in implementacijo vezja (**Run Implementation**) in programsko datoteko (**Generate Bitstream**), ki jo izvozi (**File > Export > Hardware, Include Bitstream**) in odpri SDK (**File > Launch SDK**)
- Preizkusi delovanje na razvojni plošči s testno aplikacijo, ki premika olimpijske kroge z nastavljanjem registrov GPIO.

```
#include <xil_io.h>
#include <sleep.h>

#define AXIRGB 0x43c00000
#define AXIPIX 0x43c00004
#define GPIO   0x41200000

int main()
{ int i;

  Xil_Out8(AXIRGB, 50);
  Xil_Out8(GPIO+8, 100);

  for (i=0; i<850; i++) {
          Xil_Out8(GPIO, i);
          usleep(10000);
  }
    return 0;
}
```