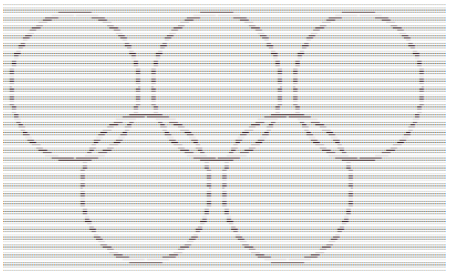
1. Izdelava komponente sistema

Cilj laboratorijske vaje je visokonivojska sinteza namenske komponente za prikaz grafike, ki je opisana s funkcijo v jeziku C.

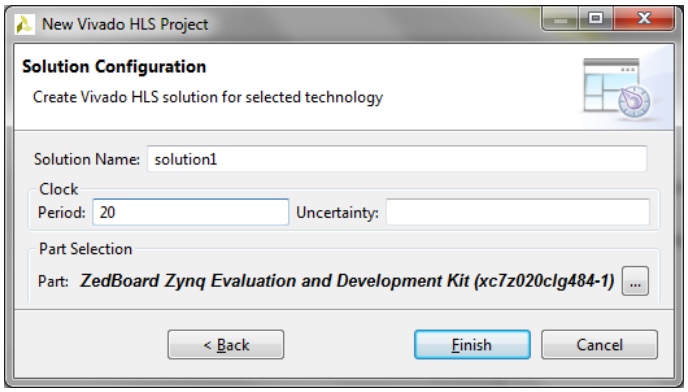
1.1 Izdelava projekta

- Zaženi program Vivado HLS in izberi **Create New Project**, ki odpre pomočnika (wizzard) za izdelavo projekta. Določi ime projekta, npr. krog in lokacijo na disku
- V naslednjem oknu določi ime glavne funkcije (krog) in dodaj datoteki krog.cpp in krog.h. Funkcija sprejme 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 28 in 30 točk, vrne vrednost 1, sicer pa vrednost 0.



```
#include "krog.h"

char krog(int x, int y)
{
  int a = 32, b; // koordinate središča
  int tmp;       // leva stran enačbe krožnice
  char i;
  char rgb = 0;  // izhod

  loop0: for (i=0; i<5; i++) {
   if (i%2==0) b = 32;
   else b = 72;

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

   // če se tocka nahaja znotraj krožnice, postavi rgb na 1
   if ((tmp < 30*30) && (tmp > 28*28)) rgb = 1;

   a += 32;
  }
```

- V tretjem oknu dodaj testno datoteko krog_test.cpp, nato pa določi periodo ure (20 ns) in izberi vrsto vezja (izbrali bomo razvojno ploščo ZedBoard.



1.2 Simulacija in sinteza vezja

- Za izvedbo simulacije funkcije izberi Project > Run C Simulation.
- Naredi sintezo vezja (Solution, Run C Synthesis) in preglej rezultate sinteze: zmogljivost vezja merjeno v zakasnitvah (latenca, interval ponovitve), zasedenost vezja in priključke vmesnika.
- Odpri analizo vezja (Solution, Open Analysis Perspective) in preglej časovni potek izvajanja operacij po posameznih ciklih.

Timing (ns)

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	20.00	16.99	2.50

Latency (clock cycles)

Summary

Latency		Interval		
min	max	min	max	Type
11	11	12	12	none

Detail

Instance

Loop

	Latency			Initiation Interval			
Loop Name	min	max	Iteration Latency	achieved	target	Trip Count	Pipelined
- loop0	10	10	2	-	-	5	no

Seznam priključkov vmesnika:

RTL Ports	Dir	Bits	Protocol
ap_clk	in	1	ap_ctrl_hs
ap_rst	in	1	ap_ctrl_hs
ap_start	in	1	ap_ctrl_hs
ap_done	out	1	ap_ctrl_hs
ap_idle	out	1	ap_ctrl_hs
ap_ready	out	1	ap_ctrl_hs
ap_return	out	8	ap_ctrl_hs
x	in	32	ap_none
y	in	32	ap_none

Časovni potek izvajanja operacij:

	Operation\Control Step	C0	C1	C2
1	y_read(read)			
2	x_read(read)			
3-19	⊞ loop0			

Časovni potek razdeljen po strojnih virih:

	Resource\Control Step	C0	C1	C2
1	⊟ I/O Ports			
2	y	read		
3	x	read		
4	ap_return			
5	⊟ Expressions			
6	i_phi_fu_67		phi_mux	
7	rgb_phi_fu_78		phi_mux	
8	a_phi_fu_56		phi_mux	
9	i_1_fu_96		+	
10	a_1_fu_136		+	
11	tmp_4_fu_125		-	
12	tmp_2_fu_114		-	
13	b_cast_cast_fu_102		select	
14	tmp_5_fu_130		*	
15	tmp_3_fu_119		*	
16	exitcond_fu_90		icmp	
17	tmp_fu_142			+
18	rgb_1_fu_164			select
19	or_cond_fu_158			&
20	tmp_9_fu_152			icmp
21	tmp_8_fu_146			icmp

1.3 Nadgradnja in optimizacija

- dodaj med vhode funkcije zbirko petih vrednosti char color[5], ki naj določa barvo posameznega kroga (izhodni signal rgb). Naredi sintezo vezje in poglej kakšne priključke je naredil program za dostop do zbirke vrednosti.

- prestavi zbirko tako, da bo notranja spremenljivka z vnaprej določenimi vrednostmi:
  char color[5] = {0x03, 0x3C, 0x15, 0x0C, 0x30};

- zamenjaj celoštevilske vrednosti vhodnih in izhodnih spremenljivk:
  - x in y naj bosta 10-bitni spremenljivki tipa ap_uint<10>
  - izhodna vrednost (rgb) pa 6-bitna ap_uint<6>
  - zamenjaj tudi podatkovni tip zbirke color
- naredi sintezo vezja in zapiši zasedenost virov

izvedba	latenca / interval	DSP48E	FF	LUT
int, char				
ap_int<>				

- dodaj še dva 10-bitna vhodna signala dx in dy s katerima bi premaknil vse krožnice za dx, dy

- dodaj direktivo, ki določa vmesnik brez krmilnih signalov: HLS INTERFACE: ap_ctrl_none
- dodaj direktivo za razvijanje zanke: HLS UNROLL
- dodaj direktivo za izvedbo s cevljenjem: HLS PIPELINE

izvedba	latenca / interval	DSP48E	FF	LUT
INTERFACE: none				
UNROLL				
PIPELINE				

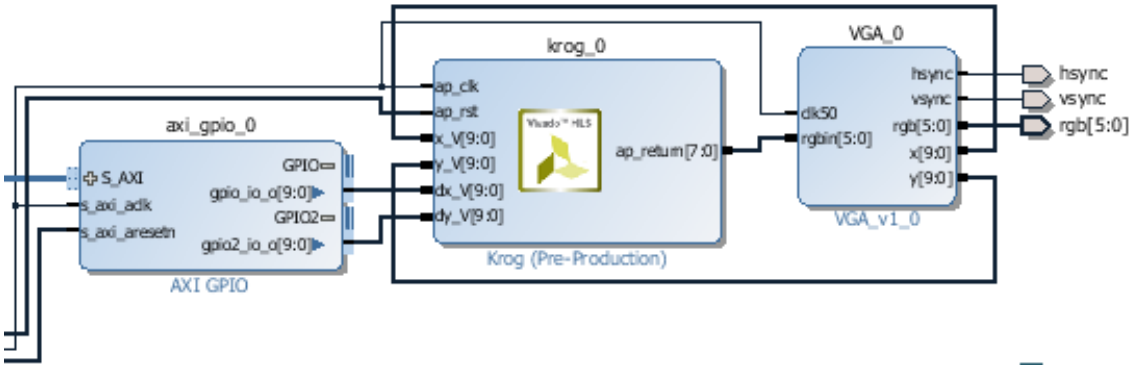
- izvozi rezultat sinteze (Export RTL) v obliki komponente IP. Vse datoteke komponente se nahajajo v podmapi: solution1\impl\ip

1.4 Integracija v sistem in preizkus

Sintetizirano komponento IP, ki se nahaja v datoteki solution1\impl\ip\xilinx_com_hls_krog_1_0.zip bomo odpakirali in prenesli v projekt, kjer so datoteke za prikaz grafike na razvojni plošči Zedboard.

- Prenesi in odpakiraj datoteke komponente IP v mapo grafika\IP\krog
- Zaženi program Vivado, odpri projekt grafika in blokovni načrt (**Open Block Design**)
- Dodaj v katalog IP mapo z novo komponento (**IP Settings, Repository Manager**) in dodaj simbol komponente v blokovni načrt (**Add IP,** krog)
- Poveži signale:

ap_clk	FCLK_CLK0 (komponenta Zynq_7)
ap_rst	peripheral_reset (Processor System Reset)
x_V	x (VGA)
y_V	y (VGA)
dx_V	gpio_io_o (AXI GPIO, odpri prvi izhod GPIO+)
dy_V	gpio2_io_o (AXI GPIO, odpri drugi izhod GPIO2+)
ap_return	rgbin (VGA)



- Preveri blokovno shemo (**Tools > Validate Design**)
- Naredi sintezo in implementacijo vezja (**Run Implementation**), ki traja nekaj minut
- Zadnji korak prevajanja je izdelava programske datoteke (**Generate Bitstream**)
- Programsko datoteko izvozimo (**File > Export > Hardware, Include Bitstream**) in odpremo SDK (**File > Launch SDK**)

1.5 Izdelava aplikacije

- Pripravili bomo novo aplikacijo: **File > New > Application Project** z imenom lab1, ki ga vnesemo v polje: Project Name, v naslednjem oknu pa potrdimo izbiro testne aplikacije z imenom Hello World.
  - V oknu Project Explorer odpri lab1 > src > helloworld.c
- Priključi ZedBoard in iz menija izberi **Xilinx Tools > Program FPGA**.
- V oknu Project Explorer klikni na aplikacijo (npr. lab1) in jo poženi na razvojni plošči **Run > Run As > Launch on Hardware**.

- Sedaj bomo nadgradili osnovni program in preizkusili komunikacijo s periferijo. Najprej dodajmo knjižnico za neposreden dostop do registrov periferije (xil_io.h) in pobrišimo deklaracijo funkcije print:

```
#include <stdio.h>
#include "platform.h"
#include "xil_io.h"// neposreden dostop do registrov

//void print(char *str);
```

- v funkcji main() dodajmo kodo za vpis podatkov v dva registra enote AXI GPIO:

```
Xil_Out16(XPAR_AXI_GPIO_0_BASEADDR, 300);
Xil_Out16(XPAR_AXI_GPIO_0_BASEADDR + 8, 200);
```