2. Matrična transformacija

Sintetizirali bomo komponento vezja za geometrijsko transformacijo koordinat z matriko realnih vrednosti.

2.1 Izdelava projekta

- Zaženi program Vivado HLS in izberi **Create New Project**, ki odpre pomočnika (wizzard) za izdelavo projekta. Določi ime projekta, npr. transform in lokacijo na disku
- V naslednjem oknu določi ime glavne funkcije (npr. mm), dodaj datoteko mm.h in naredi novo datoteko mm.cpp. Funkcija naj sprejme koordinato točke (x, y) in izračuna transformirano koordinato (fx, fy) s pomočno matrike m:

$$\begin{bmatrix} fx \\ fy \end{bmatrix} = \begin{bmatrix} m[0] & m[1] \\ m[2] & m[3] \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix}$$

- Uporabi deklaracijo funkcije mm() in podatkovnih tipov iz datoteke mm.h:

```
typedef int xy_t;
typedef float m_t;
void mm (xy_t x, xy_t y, xy_t *fx, xy_t *fy, m_t m[4]);
```

- Napiši telo funkcije, ki izračuna matrično transformacijo in preveri delovanje s testno strukturo mm_test.cpp.

2.2 Sinteza vezja in optimizacija

- Dodaj direktivo, ki določa vmesnik brez krmilnih signalov: HLS INTERFACE: ap_ctrl_none

- Naredi sintezo vezja (**Solution**, **Run C Synthesis**) in preglej rezultate sinteze: zmogljivost vezja merjeno v zakasnitvah (latenca, interval ponovitve), zasedenost vezja in priključke vmesnika.

- Zamenjaj podatkovni tip **xy_t** v **ap_int<8>** in tip koeficientov matrike iz **float** na **ap_fixed<8, 2>** (8-bitna realna števila s fiksno decimalko, dve celoštevilski mesti) in ponovno naredi sintezo vezja. Preizkusi različne nastavitve podatkovnega tipa, preveri natančnost transformacije s simulacijo in zapiši rezultat sinteze v tabelo:

izvedba	latenca / interval	DSP48E	FF	LUT
float, int				
ap_fixed<8, 2>				
ap_fixed< , >				

- V vezju je za matriko narejen pomnilniški vmesnik, ki zahteva sekvenčni dostop do podatkov. Ker imamo le 4 koeficiente, lahko razdelimo zbirko na 4 diskretne vhodne vrednosti (registre).
  Naredi novo rešitev **Project > New Solution** in dodaj na spremenljivko **m** direktivo ARRAY_RESHAPE, type: complete.

- Naredi novo rešitev in dodaj še na funkcijo mm direktivo PIPELINE ter primerjaj rezultate sinteze: **Project > Compare Reports**

izvedba	latenca / interval	DSP48E	FF	LUT
ARRAY_RESHAPE				
PIPELINE				

2.3 Obdelava podatkov iz pomnilnika

Naredili bomo sintezo funkcije, ki izvaja geometrijsko transformacijo točk iz vhodnega pomnilnika p[size*size], ki naj vsebuje 1024 točk (size=32). V funkciji naj bo zanka, ki za vsako koordinato (x,y) iz pomnilnika izračuna transformirano koordinato (fx, fy), nato pa prenese podatek iz pomnilnika **p** v pomnilnik **q**. Če je transformirana koordinata izven območja (0,0)-[size,size], naj nastavi pomnilnik **q** na vrednost '2'.

- Odstrani iz projekta testno strukturo in jo zamenjaj s transform_test.cpp, nato pa v nastavitvah določi novo glavno funkcijo: Project > Project Settings, Synthesis, Top Function: transform

```
void transform (dat_t p[size*size], dat_t q[size*size], m_t m[4])
```

- V funkciji naredi dvojno zanko za prehod čez koordinate (x,y). V vsaki iteraciji zanke s klicanjem funkcije mm() izračunaj transformirani koordinati fx in fy. Pomnilniške naslove določi s pomikanjem podatkov, kar je hitrejše kot z množenjem. Npr. naslov koordidate (x,y) v pomnilniku velikosti 32x32 izračunamo kot 32*y+x ali pa (y<<5)+x. Prenos podatkov: `q[(y<<5)+x] = p[(fy<<5)+fx];`

- Naredi sintezo in zapiši koliko ciklov je potrebnih za celoten algoritem (zanka se ponovi 1024-krat).

- Dodaj direktivo PIPELINE, ki jo priredi zanki in naredi sintezo. Ugotovi ali je pomembno kateri zanki dodaš to direktivo. Kaj se zgodi, če odstraniš kakšno direktivo s funkcije mm() ?

- Spremeni opis funkcije, tako da bo namesto dvojne zanke le ena zanka in boš izračunal koordinate na podlagi indeksa zanke.

```
x = i & 0x1f;
y = i >> 5;
```

izvedba	latenca / interval	DSP48E	FF	LUT