EmbHardw - Sobel

Sébastien Deriaz 18.12.2021

Performances

le système final, avec l'utilisation de toutes les optimisations possibles, nécessite 423 ms pour traiter une image (2.36 fps ou 108 clk/pixels).
Les optimisations les plus utiles sont, dans l'ordre :

1. L'utilisation de l'optimiseur (-O1, -O2, -O3)
2. Unrolling
3. Inlining

Structure

Grayscale

Trois méthodes sont utilisées :

- Méthode précise (C non-optimisé et VHDL)
- Méthode approximée (C semi-optimisé)
- Méthode très approximée (C optimisé)

Méthode précise

Cette méthode est celle donnée par les coefficients pour la conversion RGB -> grayscale. Elle est utilisée dans la verison non-optimisée du code ainsi que dans la méthode avec les custom instructions (en combinatoire)

```
// RGB -> R, G, B (all to 8 bits)
R = (rgb & 0xF800) >> 8; //-> RRRR'R000
G = (rgb & 0x07E0) >> 3; //-> GGGG'GG00
B = (rgb & 0x001F) << 3; //-> BBBB'B000
gray =  (R * 30 + G * 59 + B * 11) / 100
```

Méthode approximée

Cette méthode permet d'éviter le / 100 en faisant une approximation par >>8 (et en modifiant les coefficients)

```
gray = ((rgb >> 8) & 0xF8) * 77;            // red
       gray += ((rgb >> 3) & 0xFC) * 151;   // green
       gray += (rgb & 0x1F) * 224;          // blue
       gray = gray >> 8;
```

Méthode très approximée

Les multiplications et divisions sont approximées par des masques et/ou des shifts. L'erreur commise est importante mais le système reste fonctionnel

```
gray = (rgb >> 2) & 0x3E00;           // red
        gray += (rgb << 4) & 0x7E00;  // green
        gray += (rgb << 8) & 0x1F00;  // blue
        gray = gray >> 8;
```

Optimisation supplémentaire

Il est possible de transformer deux boucles for sur `x` et `y` en une seule boucle for sur `z`. Ceci a été utilisé pour optimiser la fonction `grayscale` ainsi que la fonction `sobel_complete`

Sobel

Les optimisations seront présentée dans l'ordre dans lequel elles ont été appliquées :

Unrolling

Les boucles de 3 éléments utilisées pour les masques sont supprimées. Ce qui donne 2*9 lignes de code (pour x et y)

Inlining + sobel_complete

Les fonctions sont supprimées et une seule fonction `sobel_complete` est utilisée pour effectuer x, y et le threshold

1. Combinaison des boucles for sur `x` et `y` en une seule boucle `z`
2. Suppression des lignes où `gx_array=0` et `gy_array=0`
3. Suppression des tableaux `gx_array` et `gy_array` (valeurs hard-codées)
4. Suppression des tableaux de résultats pour `x` et `y` et remplacement par des valeurs simples (car il n'y a plus besoin de passer les valeurs à `sobel_threshold`)

Custom instructions

Les 8 pixels (celui du centre est ignoré car multiplié par 0) sont transmis d'un coup dans `dataa` et `datab`. L'instruction custom effectue le masque sur x, sur y et le threshold. Le resultat est directement remis dans le tableau de sortie via le retour de la fonction