2022 Digital IC Design
Final Project

NAME	王梓帆
Student ID	N26114976

Functional Simulation Result of LZ77 Encoder

Testing Pattern 0	Pass	Testing Pattern 1	Pass	Testing Pattern 2	Pass

```
# cycle 2073e, expect(00,00,1) , get(00,00,1) >> Pass
# cycle 2075f, expect(00,00,$) , get(00,00,$) >> Pass
# ------------------------------------------------
# --------- Encoding finished, ALL PASS ---------
# ------------------------------------------------
# ** Note: $finish    : C:/Final_Project/tb_Encoder.sv(285)
#    Time: 664800 ns  Iteration: 1  Instance: /testfixture_encoder
```

```
# cycle 1a162, expect(01,01,e) , get(01,01,e) >> Pass
# cycle 1a186, expect(15,02,$) , get(15,02,$) >> Pass
# ------------------------------------------------
# --------- Encoding finished, ALL PASS ---------
# ------------------------------------------------
# ** Note: $finish    : C:/Final_Project/tb_Encoder.sv(285)
#    Time: 534435 ns  Iteration: 1  Instance: /testfixture_encoder
```

```
# cycle 1c880, expect(0b,02,0) , get(0b,02,0) >> Pass
# cycle 1c8a3, expect(00,00,$) , get(00,00,$) >> Pass
# ------------------------------------------------
# --------- Encoding finished, ALL PASS ---------
# ------------------------------------------------
# ** Note: $finish    : C:/Final_Project/tb_Encoder.sv(285)
#    Time: 584500 ns  Iteration: 1  Instance: /testfixture_encoder
```

Functional Simulation Result of LZ77 Decoder

Testing Pattern 0	Pass	Testing Pattern 1	Pass	Testing Pattern 2	Pass

```
#   == Decoding string "1"
# cycle 02004, expect 1, get 1 >> Pass
# ------------------------------------------------------
# -------------- Decoding finished, ALL PASS -----------
# ------------------------------------------------------
# ------------------------------------------------------
# ----- Interpolation finished, result is written out -----
# ------------------------------------------------------
# ** Note: $finish    : C:/Final_Project/tb_Decoder.sv(384)
#    Time: 160685 ns  Iteration: 0  Instance: /testfixture_decoder
```

```
# cycle 02003, expect 5, get 5 >> Pass
# cycle 02004, expect f, get f >> Pass
# ------------------------------------------------------------
# -------------- Decoding finished, ALL PASS --------------
# ------------------------------------------------------------
# ------------------------------------------------------------
# ----- Interpolation finished, result is written out -----
# ------------------------------------------------------------
# ** Note: $finish    : C:/Final_Project/tb_Decoder.sv(384)
#    Time: 160685 ns  Iteration: 0  Instance: /testfixture_decoder
```

```
# cycle 02003, expect 7, get 7 >> Pass
# cycle 02004, expect 0, get 0 >> Pass
# ------------------------------------------------------------
# -------------- Decoding finished, ALL PASS --------------
# ------------------------------------------------------------
# ------------------------------------------------------------
# ----- Interpolation finished, result is written out -----
# ------------------------------------------------------------
# ** Note: $finish    : C:/Final_Project/tb_Decoder.sv(384)
#    Time: 160685 ns  Iteration: 0  Instance: /testfixture_decoder
```

Quality of Interpolated Results

Testing Pattern 0	24.54	Testing Pattern 1	24.56	Testing Pattern 2	27.67

```
C:\N26114976_Final>python calPSNR.py
PSNR of image 0: 24.540343856679172
PSNR of image 1: 24.55962861736303
PSNR of image 2: 27.67269071532993
```

Description of your design

這次的期末專案，我個人認為最困難的地方在於提升 PSNR，我發現每張照片都會有非常極端的狀況，像是這次的第一張照片，我發現每一格都使用正上(b)與正下（e）的平均，則 PSNR 居然提升了將近 1(dB)，然而這樣的做法，會使得其他的兩張照片的 PSNR 大幅提升，所以我個人想到的做法，比較是針對這三張圖片，希望可以再大幅提升第一張照片 PSNR 的同時，其他兩張照片所降低的值可以更小。

這邊我的作法是將需要用到 a, c, d, f 插值的部分切得更細，而其他的狀況，則全部使用正上（b）與正下（e）的平均，以 a 與 f 為例：

```
if(((a + b + c) / 3) - a < 3 || ((a + b + c) / 3) - a > -3)
    if(((a + b + c) / 3) - a < 2 || ((a + b + c) / 3) - a > -2)
        min = {1'b0, a * 5/8} + {1'b0, f *3/8};
    else
        min = {1'b0, a * 17/32} + {1'b0, f *15/32};
else if(((d + e + f) / 3) - f < 3 || ((d + e + f) / 3) - f > -3)
    if(((d + e + f) / 3) - f < 2 || ((d + e + f) / 3) - f > -2)
        min = {1'b0, a * 3/8} + {1'b0, f *5/8};
    else
        min = {1'b0, a * 15/32} + {1'b0, f *17/32};
else
    // min = ({1'b0, a} + {1'b0, f}) >> 1;
    min = ({1'b0, b} + {1'b0, e}) >> 1;
```

原先要做左上（a）跟右下（f）的部分，我使用了特徵強化的作法，如果 a, b, c 的平均，很接近 a 的話，就使用不同 a 跟 f 的比例，去做插值；如果不靠近 a 跟 f，就使用正上（b）與正下（e）的平均，我們可以發現到，這個做法雖然其 PSNR 在第三種圖片上有小幅度的下降，但在另外兩張照片都有所提升，甚至在第一張照片，提升了將近 1(dB) 的 PSNR 值。最後比起 Basic 的數值，總共提升了 0.68(dB)。

Scoring = Pattern 0 PSNR + Pattern 1 PSNR + Pattern 2 PSNR = 76.77(dB)

The higher, the better.