2022 Digital IC Design

Final Project

1. Introduction

In digital TV technology, videos or images will be compressed before transmission to transmit a huge amount of image data efficiently. The compressed data are then transmitted via satellite, terrestrial radio, or cable. After the Digital TV at the end-user side receives the compressed image data, it will decompress them with the built-in digital IC. Finally, the decompressed images are interpolated to provide different resolutions according to the specifications of the monitors.

In the final project, you are asked to implement the steps of compression, decompression, and interpolation in digital TV technology. The compression(encoding) and decompression(decoding) process adopt the LZ77 method, which has already been implemented in homework 3. For the interpolation process, you can adopt the edge-based line average interpolation, which is implemented in homework 4, or other interpolation method that can reach higher quality.

The score of the final project is judged according to the correctness of the encoder/decoder and the quality of the interpolated images. Three testing data are provided in this project, you have to pass the functional simulation of all of them to get the score of each part.

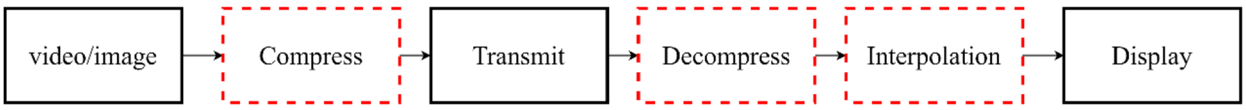

Fig. 1. The process from the transmission to display in digital TV technology.

2. Design Specifications

In this project, the testbench is divided into two modules. One is for encoding and the other one is for decoding and interpolation, which correspond to the transmission side and receive side in digital TV technology. Fig. 2 and Fig. 3 show the block overview of them. The testbench for encoding will verify the correctness of the LZ77 encoder module (*LZ77_Encoder*), while the testbench for decoding and interpolation will verify the correctness of the LZ77 decoder module (*LZ77_Decoder*) and produce interpolated image with the interpolation module

(*ELA*). The LZ77 encoder and decoder must follow the specifications of LZ77 algorithm, and the size of search buffer and look-ahead buffer are 30 and 25, respectively. The interpolation algorithm in *ELA* module can be modified. However, the quality of interpolated images can't be worse than that obtained from the edge-based line average interpolation. Otherwise, you won't get the score of the interpolation part. The method for quality judgment will be introduced in the scoring section.

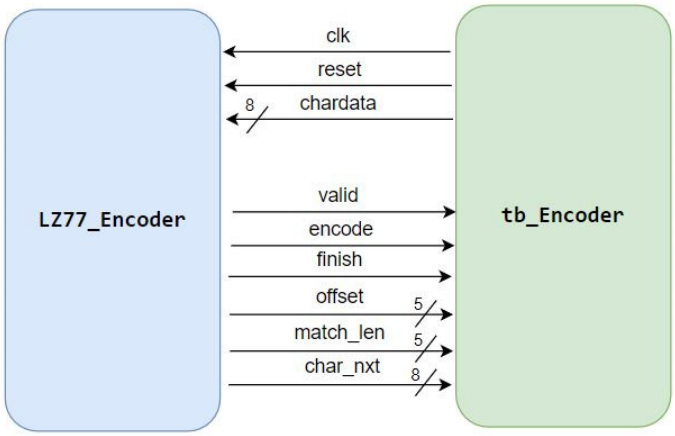

Fig. 2. Overview of the encoder side.

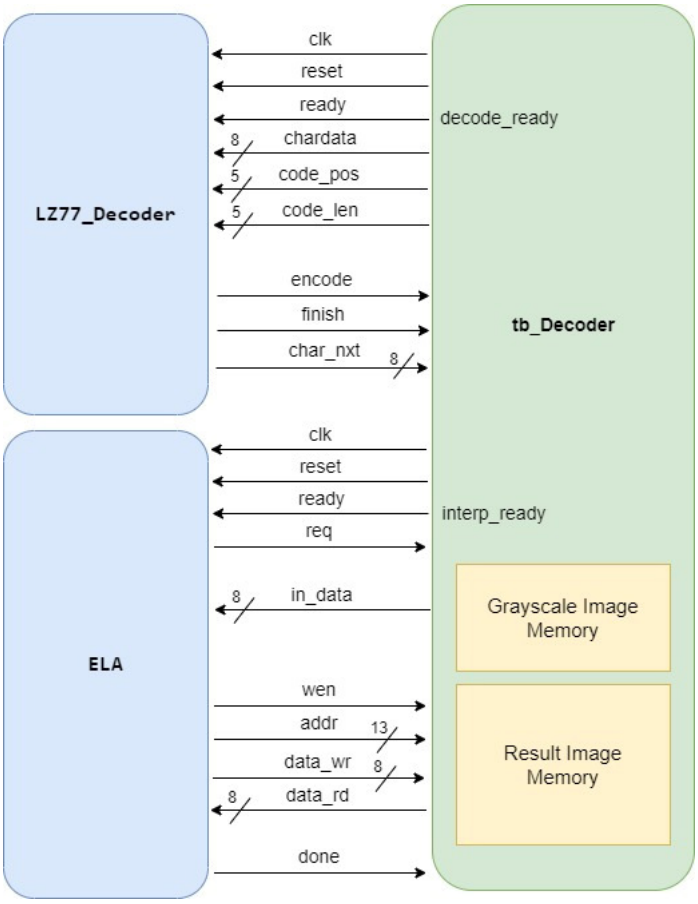

Fig. 3. Overview of the decoder side.

2.1 I/O Interface

Signal Name	I/O	Width	Description
clk	I	1	This circuit is a synchronous design triggered at the positive edge of *clk*.
reset	I	1	Active-high asynchronous reset signal.
chardata	I	8	Character to be encoded.
valid	O	1	When output encoding result, set *valid* signal to high. The testbench will check the output result when *valid* signal is high.
encode	O	1	When encoding, set *encode* signal to high.
offset	O	5	The offset between the position of the first character in the matched string to the head of the search buffer.
match_len	O	5	The length of matched string.
char_nxt	O	8	The next character behind last matched character in look-ahead buffer.
finish	O	1	When the encoding process finishes, set *finish* signal to high.

Table 1. LZ77 Encoder I/O

Signal Name	I/O	Width	Description
clk	I	1	This circuit is a synchronous design triggered at the positive edge of *clk*.
reset	I	1	Active-high asynchronous reset signal.
ready	I	1	When *ready* signal is high, the input code (*code_pos*, *code_len*, *chardata)* is valid.
code_pos	I	5	The begining position of the matched string in the search buffer.
code_len	I	5	The length of matched string.
chardata	I	8	The next character behind matched string.
encode	O	1	When decoding, set *encode* signal to low.
char_nxt	O	8	Decoded character.
finish	O	1	When the decoding process finishes, set *finish* signal to high.

Table 2. LZ77 Decoder I/O

Name	I/O	Width	Description
clk	I	1	System clock signal. This system is synchronized with the positive edge of the clock.
rst	I	1	Active-high asynchronous reset signal.
ready	I	1	When *ready* signal is high, the ELA module can request pixel data from the grayscale image memory.
req	O	1	The request signal for a row of pixels.
in_data	I	8	The 8-bit input pixel data. When req signal is high, the testbench will send a row of image pixels from Grayscale Image Memory, one pixel per cycle.
wen	O	1	Result Image Memory write enable signal. When this signal is low (read), the data with address *addr* in Result Image Memory is sent by *data_rd*. When this signal is high (write), the data sent by data_wr is written to the address *addr* in Result Image Memory.
addr	O	13	Result Image Memory read/write address. This signal indicates which address of the Result Image Memory will be read or written.
data_wr	O	8	Result Image Memory write data signal, which represents an 8 bits unsigned integer. The interpolation result of the ELA circuit is written to Result Image Memory through this signal.
data_rd	I	8	Result Image Memory read data signal, which represents an 8 bits unsigned integer. This signal is used to send the Result Image Memory data to ELA circuit.
done	O	1	After the interpolation of the whole image is completed and the result is completely output, setting this signal to high and the testbench will start checking if the result is correct.

Table 3. Interpolation Module I/O

2.2 File Description

File Name	Description
LZ77_Encoder.v	The module of LZ77_Encoder.
LZ77_Decoder.v	The module of LZ77_Decoder.
ELA.v	The module for line interpolation.
tb_Encoder.sv	Testbench for LZ77 encoder. The content is **not allowed** to be modified.
tb_Decoder.sv	Testbench for LZ77 decoder and the line interpolation. The content is **not allowed** to be modified.
calPSNR.py	Python code for calculating PSNR of the interpolated results.

Table 4. File description.

2.3 Functional Description

The function of the LZ77 encoder is the same as that in homework 3, with the only difference being the size of the search buffer and the look-ahead buffer. In this project, the size of the search buffer and the look-ahead buffer are 30 and 25, respectively. For the detail of the algorithm of the LZ77 encoder, you can examine the pdf file of homework 3.

The function of the LZ77 decoder is also similar to that in homework 3. The size of the search buffer and the look-ahead buffer are set as 30 and 25, and a new signal is added in the LZ77 decoder module. The new signal is *ready*, which represents the validity of the input code. Only when the *ready* signal is high, the LZ77 decoder has to output decoding results according to the *code_pos*, *code_len*, and *char_nxt* signals. Fig. 4 shows the timing diagram of the decoding process. For the detail of the algorithm of the LZ77 decoder, you can examine the pdf file of homework 3.

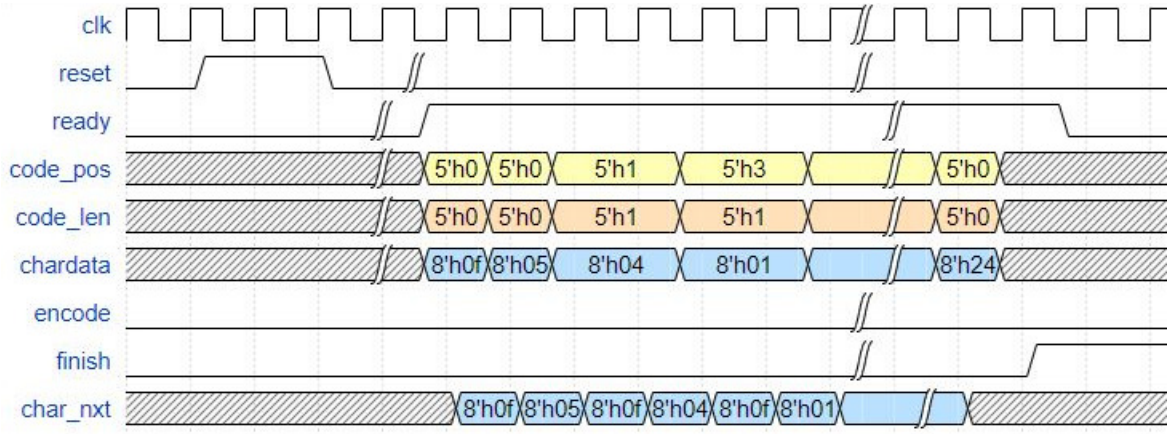

Fig. 4. Timing diagram of the decoding process.

For the interpolation module, only the way to read and write the memory is specified. The interpolation algorithm is unrestricted. However, the quality of interpolated images can't be worse than that obtained from the edge-based line average interpolation. Otherwise, you won't get the score of the interpolation part. The given test patterns in this homework are three 128x32 images, to which you need to design a circuit to perform line interpolation. Each row in the given test pattern is odd field, and you have to calculate the even row by line interpolation method as shown in Fig. 5.

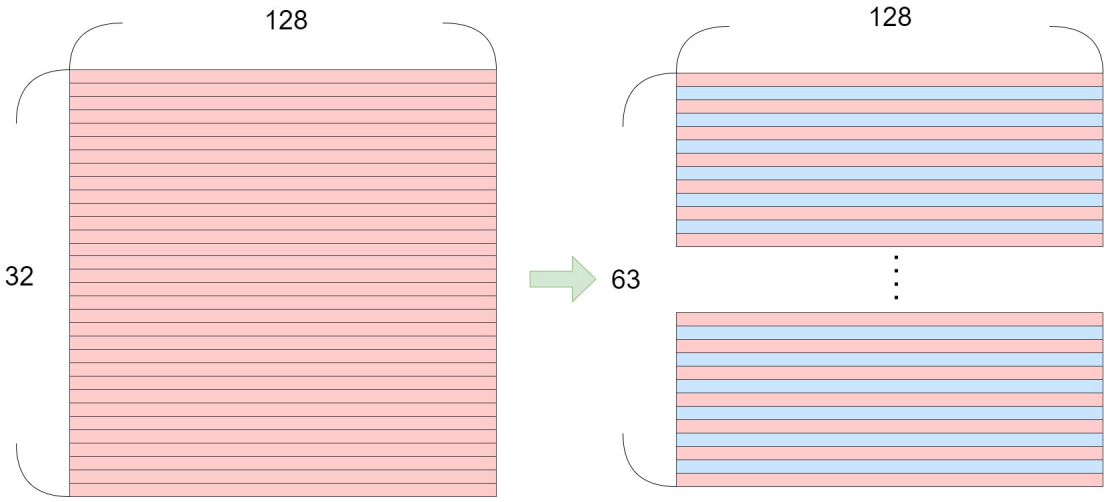

Fig. 5. The line interpolation process. Red rows are odd field and blue rows are even field.

The input order is from the first row to the last row and each row will only send one time. After the *ready* signal is pulled up, you can set the *req* signal as high to get one row of image data. The pixel data will be sent into the ELA circuit by *in_data* signal at the negative edge of each cycle. Please receive *in_data* at the positive edge of the clock cycle to avoid data unstable. After sending 128 pixels of a row, the testbench will stop sending data until the next high-level *req* signal. The timing diagram of reading grayscale memory is shown in Fig. 6.

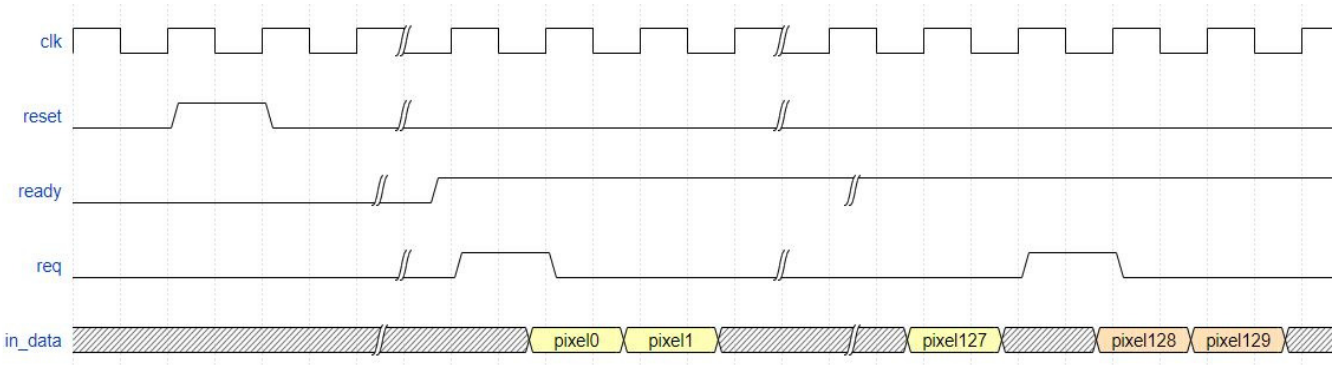

Fig. 6. Timing diagram while reading Grayscale Image Memory.

The Result Image Memory is an 8064x8 bit memory, which is used to store the result image of interpolation. The correspondence between each pixel of the image and memory arrangement is shown in Fig. 7. Remember to set the *done* signal to high after writing all the result pixels to memory, then the testbench will write out the result as raw image and terminate the simulation.

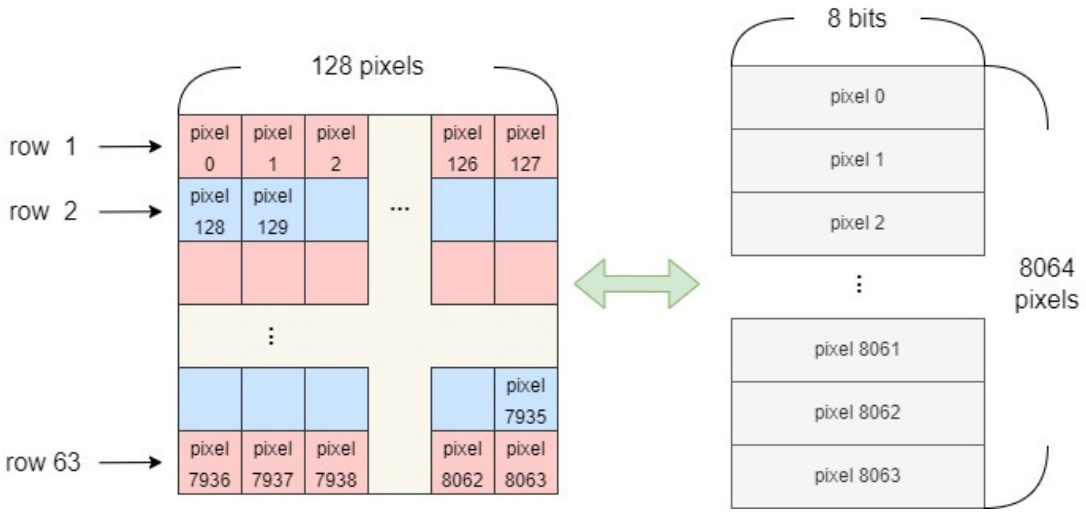

Fig. 7. Arrangement mapping of result image (left) and Result Image Memory (right).

The timing diagrams of reading/writing Result Image Memory are shown in Fig. 8 and Fig. 9. For the detail of specifications of the Result Image Memory, you can examine the pdf file of homework 4.

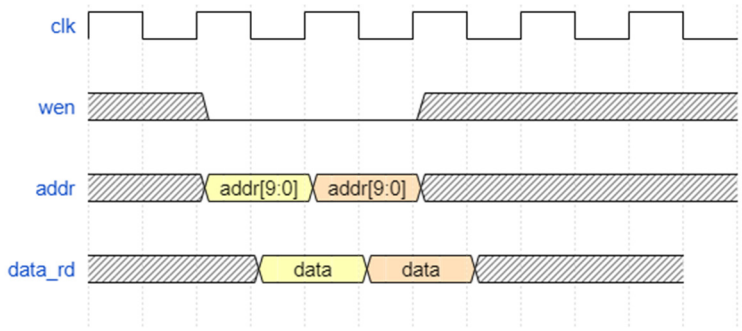

Fig. 8. Timing diagram of reading Result Image Memory.

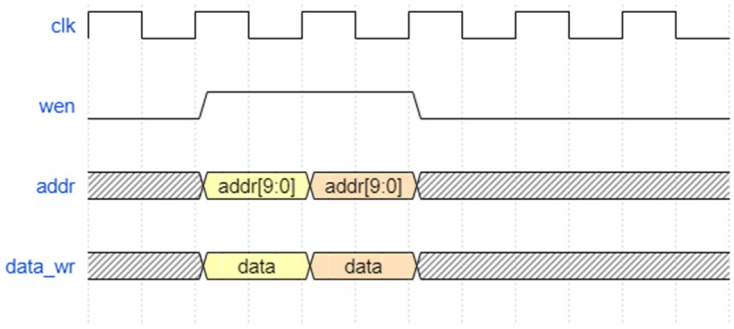

Fig. 9. Timing diagram of writing Result Image Memory.

3. Scoring

In this project, you only need to pass the functional simulation. Three testing data are provided in this project, you have to pass the functional simulation of all of them to get the score of each part. You can change the pattern path in the testbench (tb_Encoder.sv/ tb_Decoder.sv) to simulate with different image data.

```
`define PAT         "./img0/testdata_encoder.dat"
// `define PAT        "./img1/testdata_encoder.dat"
// `define PAT        "./img2/testdata_encoder.dat"
```

```
`define PAT             "./img0/testdata_decoder.dat"
`define DECODE_OUT      "./img0/decode_result.raw"
`define INTERP_OUT      "./img0/interp_result.raw"
```

3.1 LZ77 Encoder (25%)

All of the result should be generated correctly, and you will get the following message in ModelSim simulation.

```
# cycle 20632, expect(17,03,6) , get(17,03,6) >> Pass
# cycle 2065b, expect(0d,05,b) , get(0d,05,b) >> Pass
# cycle 20684, expect(0f,03,9) , get(0f,03,9) >> Pass
# cycle 206ab, expect(0b,03,6) , get(0b,03,6) >> Pass
# cycle 206d0, expect(1d,01,5) , get(1d,01,5) >> Pass
# cycle 206f5, expect(01,03,6) , get(01,03,6) >> Pass
# cycle 2071b, expect(1b,02,9) , get(1b,02,9) >> 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 ---------
# ------------------------------------------------
```

Fig. 10. Functional simulation result of the LZ77 encoder.

3.2 LZ77 Decoder (25%)

All of the result should be generated correctly, and you will get the following message in ModelSim simulation.

```
#   == Decoding string "6566"
# cycle 01ffd, expect 6, get 6 >> Pass
# cycle 01ffe, expect 5, get 5 >> Pass
# cycle 01fff, expect 6, get 6 >> Pass
# cycle 02000, expect 6, get 6 >> Pass
#   == Decoding string "6b9"
# cycle 02001, expect 6, get 6 >> Pass
# cycle 02002, expect b, get b >> Pass
# cycle 02003, expect 9, get 9 >> Pass
#   == Decoding string "1"
# cycle 02004, expect 1, get 1 >> Pass
# ------------------------------------------------------------
# -------------- Decoding finished, ALL PASS --------------
# ------------------------------------------------------------
```

Fig. 10. Functional simulation result of the LZ77 decoder.

3.3 Line Interpolation (30%)

After the *done* signal of *ELA* module is pulled up, testbench will write out the image data in Result Image Memory. You can get the score of this part as long as the quality of the three result images are no worse than that obtained by adopting edge-based line average interpolation. The method for quality judgment will be introduced in the next section.

```
# ------------------------------------------------------
# ----- Interpolation finished, result is written out -----
# ------------------------------------------------------
```

3.4 Performance (20%)

The quality of result images is judged with their PSNR. The definitions of *PSNR* are as following equations. I represents the golden image, which has full information of even rows and odd rows. K represents the interpolated image obtained from the interpolation algorithm.

$$MSE = \frac{1}{mn}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}[I(i,j) - K(i,j)]^2$$

$$MAX_I = 2^B - 1$$

$$PSNR = 20 \times log_{10}(\frac{MAX_I}{\sqrt{MSE}})$$

The *PSNR* of your interpolated results must equal to or higher than the baseline to get the score of line interpolation and performance. The baseline of each testing image is listed in Table 5. The *PSNR* is rounded to the second decimal point.

Testing image	Baseline *PSNR*
img0	23.80
img1	24.51
img2	27.88

Table 5. Baseline of each testing image.

The software for *PSNR* calculation has been provided. You can get the results by simply executing the python code. Before executing, you may need to install the required modules. The installation commands are also listed in the following.

```
execution:
    python calPSNR.py

installation:
    pip install numpy
    pip install Pillow
```

4. Submission

4.1 Submitted files

You should classify your files into two directories as the table below and compress them to .zip format. The naming rule is PROJECT_studentID_name.zip. If your file is not named according to the naming rule or file hierarchy is wrong, you will lose 5 points.

RTL category	
*.v	All of your Verilog RTL code
Documentary category	
*.pdf	The report file of your design (in pdf).

4.2 Report file

Please follow the spec of report. If your report does not meet the spec, you may lose part of score. You are asked to describe how the circuit is designed as detailed as possible, especially the interpolation algorithm. Please fill the field of *PSNR* of each interpolated result according to the output of the python code.

```
PSNR of image 0: 23.797002896730227
PSNR of image 1: 24.509962850189638
PSNR of image 2: 27.875880720521792
```

4.3 Note

In this homework, you are allowed to modify the defined *CYCLE* and *End_CYCLE* in testbench file. *CYCLE* decides the clock width in your design, and *End_CYCLE* decides the maximum cycles your circuit take to complete simulation. Please do not modify any other content of testbench.

Please submit your .zip file to folder for FINAL Project in moodle.
Deadline: 2022/6/22 16:59

If you have any problem, please contact TA by email.
p78101548@gs.ncku.edu.tw