# Real-time FPGA-based Acoustic Imaging

B09901018 劉則暐
B09901105 謝博揚
B09901156 張哲銘

***Abstract*—A real-time FPGA-based Acoustic Imaging Camera is presented in this report. Since current available devices requires large and expensive equipment to implement such signal processing, we use a low-complexity algorithm for hardware development. The system consists of 16 MEMS microphones and performs calculation on Altera Cyclone IV DE2-115. The intensity image is then shown on VGA with a resolution of 80×60 pixels at 30 frames per second.**

## I. Introduction

Acoustic Imaging is to visualize the origin and intensity of audible sound waves (ranging from 20 to 20k Hz) from a certain plane. There are lots of applications of acoustic imaging camera, such as in industry, it can help people detect where the noise come from, then find what causes machine to malfunction; it can also be used for planning of placement, and optimization of noise barriers in civil engineering.

### *A. Motivation*

Our final project is aiming at making low-cost acoustic imaging camera, since an acoustic imaging camera usually costs up to tens of thousands of dollars in online shopping due to requirement large computations or memory throughput. Besides, we also make it real-time visualization to provide better and immediate feedback. A real-time version is imperative for applications, where the sound-source only appears occasionally or at different locations during the measurement.

### *B. Outline*

The outline of this report is as follows. Section II gives a brief introduction to Delay-and-sum Beamforming algorithm. Section III describes the system architecture. Section IV describes how the Top module controls the state and achieve such task. Section V is the file strucutre of our design. Section VI is the screenshot from Quartus II, and we put our final thoughts at the end of the report.

## II. Delay and Sum Beamforming Algorithm

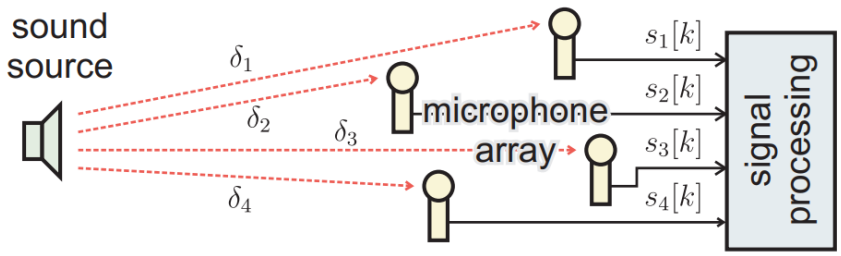



Fig. 1. Delay-and-sum Beamforming Algorithm diagram. Reference: [1].

The key idea is to delay the signals from each microphone in the way that all sound waves originating from a point of interest can interfere constructively, and all the other sound waves interfere destructively, as shown in Fig.1 . First, we compute the delay from sound sources to microphones, where $\boldsymbol{m_i}$ denotes the position of microphone, $\boldsymbol{p}$ denotes the position of sound source, $v$ is the sound of speed, and $f_s$ is the sampling frequency.

$$\delta_i(\boldsymbol{p}) = \frac{f_s}{v}|\boldsymbol{p} - \boldsymbol{m}_i| \tag{1}$$

Then, the delayed version of sound waves for each microphone is summed together. $N$ stands for the number of microphones, in our case, $N$ is equal to 16.

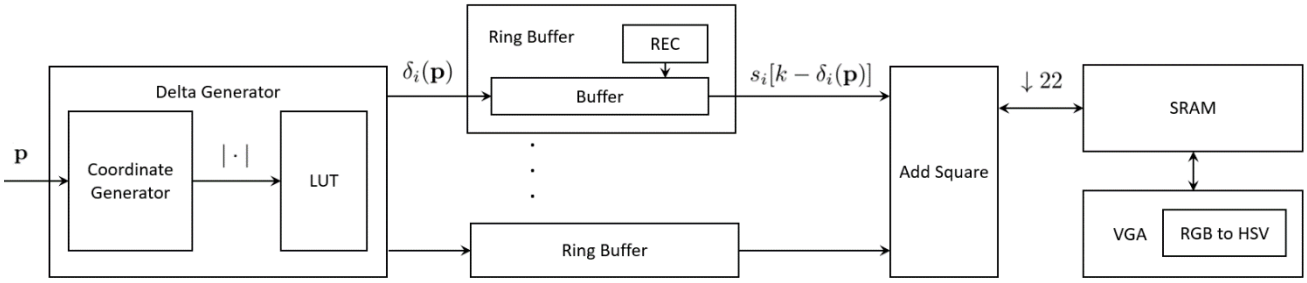



Fig. 2. The block diagram of our design (Some details omitted). The sound data is provided by 16x microphones, and the output is VGA screen with output frequency at 30Hz.

$$s(\boldsymbol{p})[k] = \sum_{i=1}^{N} s_i[k - \delta_i(\boldsymbol{p})] \tag{2}$$

Finally, the intensity of the sound wave from a particular point in space can be computed by squaring the delayed sound waves. We also average $L$ sampling points' intensity to smoothen the intensity values over time.

$$I(\boldsymbol{p})[k] = \frac{1}{L}\sum_{l=0}^{L-1} |s(\boldsymbol{p})[k-l]|^2 \tag{3}$$

## III. System Architecture

In order to realize real-time processing of delay-and-sum beamforming on FPGAs, several components are described in the following sections. The block diagram of our system is shown in Fig. 2.

### *A. Delta Generator*

As previously mentioned, large amount of pixel points and microphones results in numerous amount of delay calculations. The delays $\delta_i(\boldsymbol{p})$ are possible to be pre-calculated and store in memory, however, it is not possible to get all $\delta_i(\boldsymbol{p})$ when receiving a pixel point $\boldsymbol{p}$, since SRAM in DE2-115 only supports single address accessing. Thus, we decided to store the look-up table (LUT) in register. Due to the limited amount of logic elements on FPGA, we proposed several space optimization methods (e.g., an image of 80×60 pixels requires 76800 delay values for 16 microphones).

Before implementing on FPGA board, we used Python to simulate the delay values for all microphones and pixel point. The result is shown in Fig. 3 (a)(b). In our design, the microphones are labeled in a 2D array, [0][0] to [3][3] as in common order. Not surprisingly, all microphones exhibit similar pattern in general, though we only presented 2 delay patterns of contingent microphones. Note that microphone[0][0] exhibits its pattern in the right and lower part on the original map, which is created virtually as shown in Fig. 3 (c). The delay pattern for a microphone can be viewed as "capturing" part of the original map, and different microphones can be viewed as contributing different coordinate shift on a larger map. Let the center of the sound source be located at (0,0) on the original map. For each microphone, the corresponding coordinate can be viewed as shifting the origin to $-\boldsymbol{m}_i$. Thus for any pixel point $\boldsymbol{p}$, the *i*-th microphone has its delay value located at

$$\boldsymbol{p}_i = \boldsymbol{p} + \boldsymbol{m}_i \tag{4}$$

Therefore, we only need to store the original map instead of 16 similar maps. These are done by the module "Coordinate Generator" inside this module. Furthermore, since sound is transmitted to different orientation equally, the delta pattern presents a circular symmetry. We can further reduce the space usage by mapping all points to first quadrant by taking the absolute value of each component in the coordinate. Because the pattern is

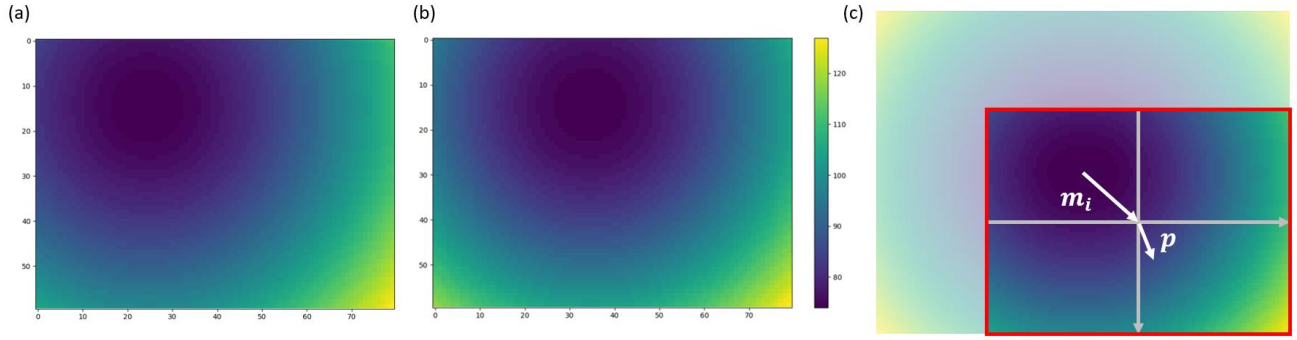



Fig. 3. The Python simulation (matplotlib) of pixel width 1cm, pixel shape 80×60 and distance 50 cm. (a) The delta pattern of the microphone[0][0]. (b) The delta pattern of the microphone[0][1]. (c) The original map. The red box shows the pattern of microphone[0][0], which can be viewed as shifting the coordinate by a distance $-\boldsymbol{m}_i$.

extremely sensitive to various of conditions (e.g., distance of pixel plane to microphone plane, pixel width, number of pixels…), we automate the generation of this Verilog code by Python.

In [1], 2 CORDICs are used in cascade to calculate the Euclidean distances of 3D vector $\boldsymbol{r} = |\boldsymbol{p} - \boldsymbol{m}_i|$. Though it can effectively reduce memory usage, however, it does not utilize the pattern similarity and does many repetitive computation.

### *B. Ring Buffer*

There is a sub-module called "Recorder" inside "Ring buffer" module. "Recorder" receives 24-bit microphone data through I2S interface with 50kHz sampling frequency and the data will be stored in the ring buffer in sequence. Depends on the delay calculated by "delta generator", the ring buffer output the corresponding 24bit data. The delay is range from 74 to 127 sample, the larger the delay the older the data is chosen.

For the reason that we need 16 microphones, there are 16 ring buffer in our architecture. For each ring buffer, the buffer size is $(\delta_{max} - \delta_{min} + L)$×24 bits. Notice that we don't have to store $\delta_{max}$ data because sound is a consecutive signal. Besides storing $(\delta_{max} - \delta_{min})$ data, we have to store $L$ more data because we will average $L$ frame results to obtain a smoothen sound pattern.

### *C. Storing data in SRAM*

After we get the 24-bit sound data from each microphone's ring buffer, we will sum up all data given by all ring buffers. Then, we will square the summation to get the intensity. For a 24-bit signed data, the intensity is at most 56 bit. Even we divided $L$ before we store in SRAM, the intensity is still 52 bit. However, the SRAM can only access 16 bit per cycle (for single address accessing), so if we store the whole information into SRAM, it requires 3 cycles to read the information and lead to high latency, which is something that we want to avoid. Therefore, truncate the unnecessary bit is necessary. We use logic analyzer to analyze the range of the sound and truncate the sound intensity to 16 bit. By truncating unnecessary intensity bit, we can calculate more pixel in same amount of time without losing the quality of the intensity information.

### *D. VGA*

The last step is to render the sound intensity for each pixel through VGA. We need to transform 16-bit data to RGB , which is the input of VGA. A naive solution is linear transform the data to RGB. Nevertheless, linear transformation will cause the region of red, green, blue too big and these color will dominate the spectrum, as shown in Fig. 4 (a). To obtain rich colors, we first transform data to HSV then transform to RGB rather than directly transform to RGB. Note that the transformation requires numerous floating point calculations, so instead of using the transform directly, we estimate the transform that requires only integer operations. In HSV representation, each RGB is at least a nonzero baseline and we can also set V to 100% to get a bright color. The effect of the method proposed above can be clearly seen in Fig. 4 (b).

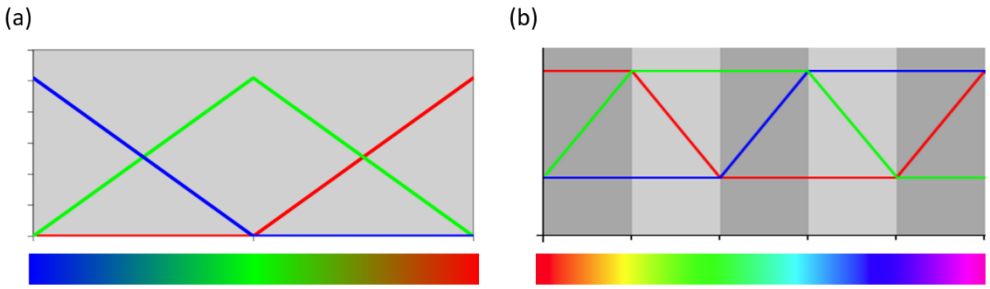
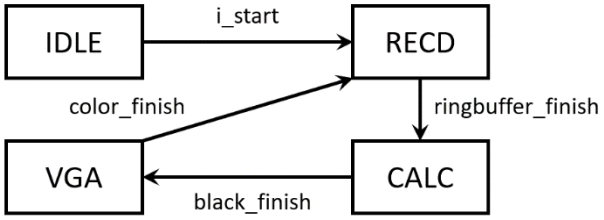



Fig. 4. (a) Linear RGB rendering method. Note that the linear interpolation does not effectively reflects the continuous spectrum of intensity. (b) HSV method. The intensity is first transform to corresponding H, and with pre-defined S(70) and V(100), we trarsform the color back to RGB.

Since the SRAM only supports single port read/write, while VGA display requires a full 60HZ cycle to read data (for displaying 640×480 pixels), we show the pattern in 1 60Hz cycle and show a black screen in another cycle, so that calculation can be done in the cycle that does not display the pattern.

## IV. Overall State Control




"Top" module has four states: IDLE, RECORD, CALCULATE, VGA. The module changes from state IDLE to state RECORD when the start button is pressed. At the RECORD state, the microphone array receives a piece of sound data. After the receiving data fill up the ring buffer, the module changes to state CALCULATE. At this state, the module run through all pixels $L$ times and generate a frame of sound intensity pattern. The module then changes to state VGA and display the image. Notice that the module needs 2 cycles to calculate the sound intensity for each pixel $\boldsymbol{p}$. First, the module read out the partially summed intensity depends on the pixel coordinate. Second, the module adds current intensity divided by $L$ to the read data and write back to the same address in SRAM. By alternatively read and write, the module can calculate the averaged sound intensity of all pixels without buffering intermediate data in register.

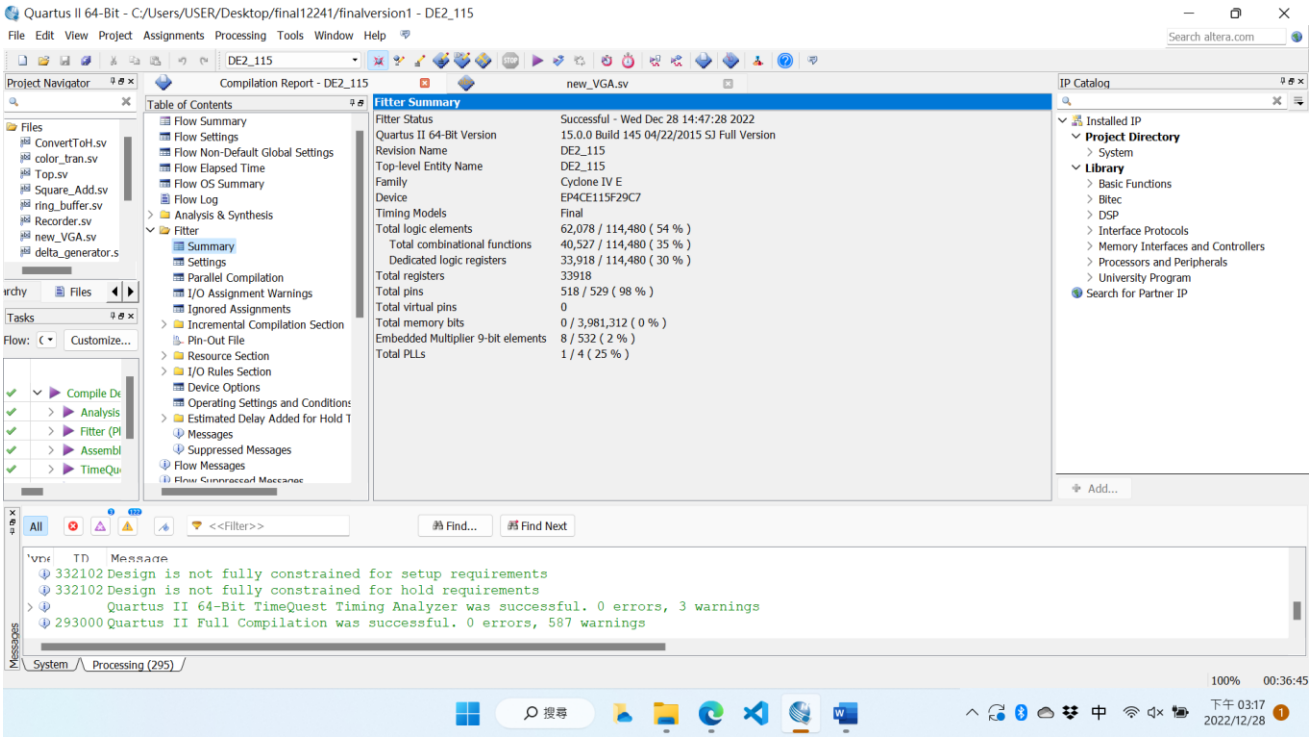
## V. File Structure

The file structure of our final project is listed as follows:

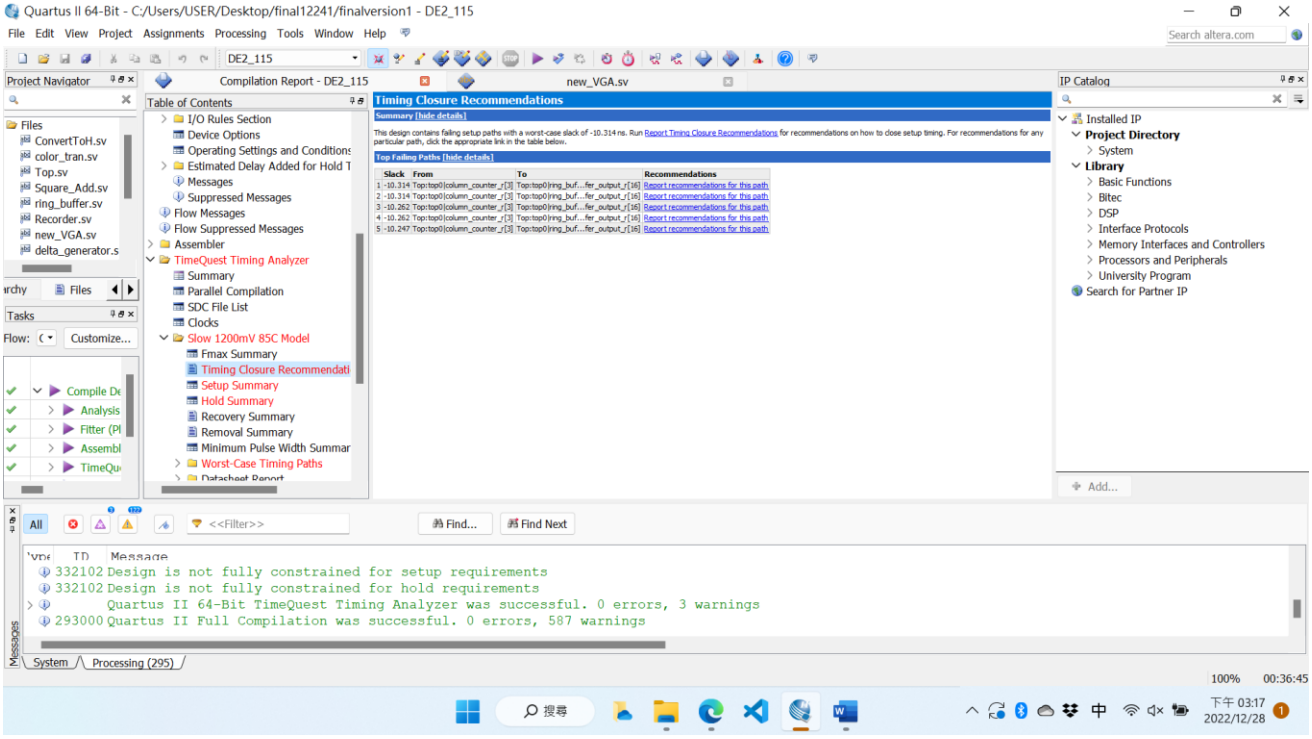
```
├──team07_final_report.pdf
└──src/
    ├──delta_generator.sv
    ├──Recorder.sv
    ├──ring_buffer.sv
    ├──Add_ Square.sv
    ├──VGA.sv
    ├──ConvertToH.sv
    ├──color_tran.sv
    ├──Top.sv
    ├──altpll.v
    ├──test_delta_new.py
    └──DE2_115/
        ├──DE2_115.sv
        ├──Debounce.sv
        └──SevenHexDecoder.sv
```

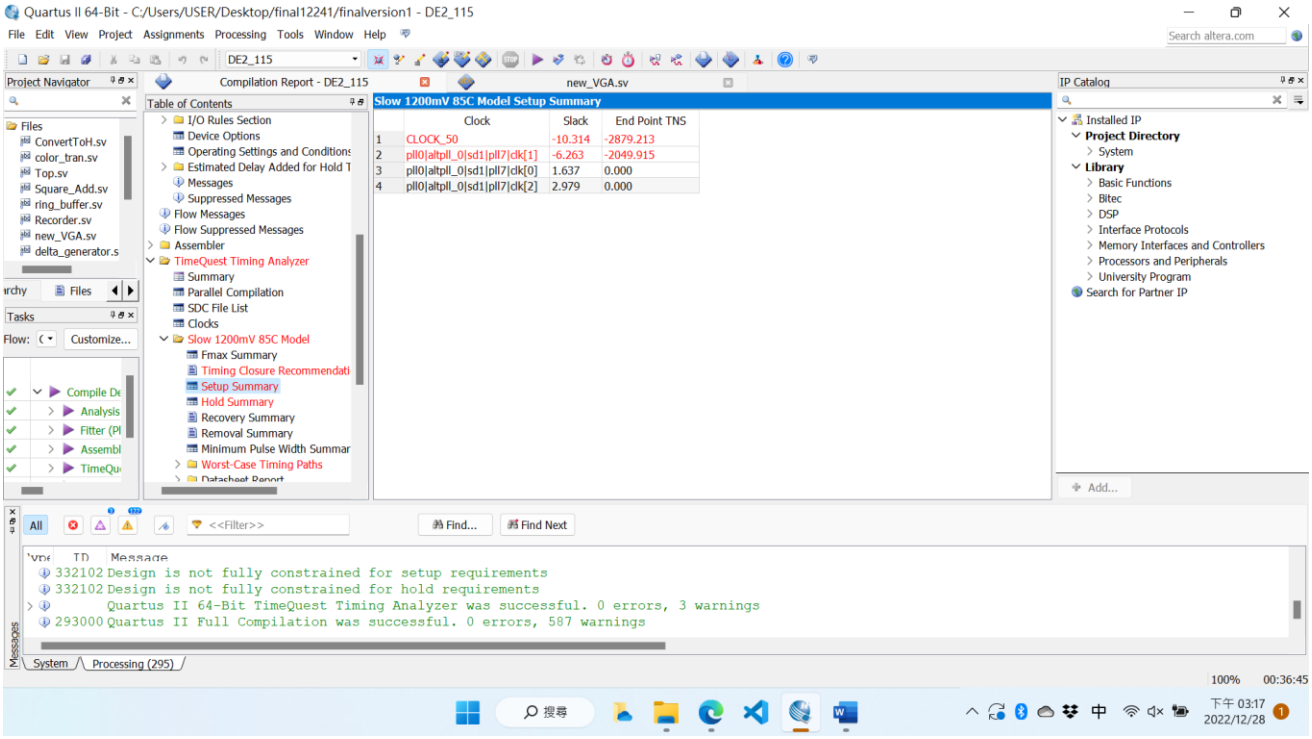
# VI. Screenshots

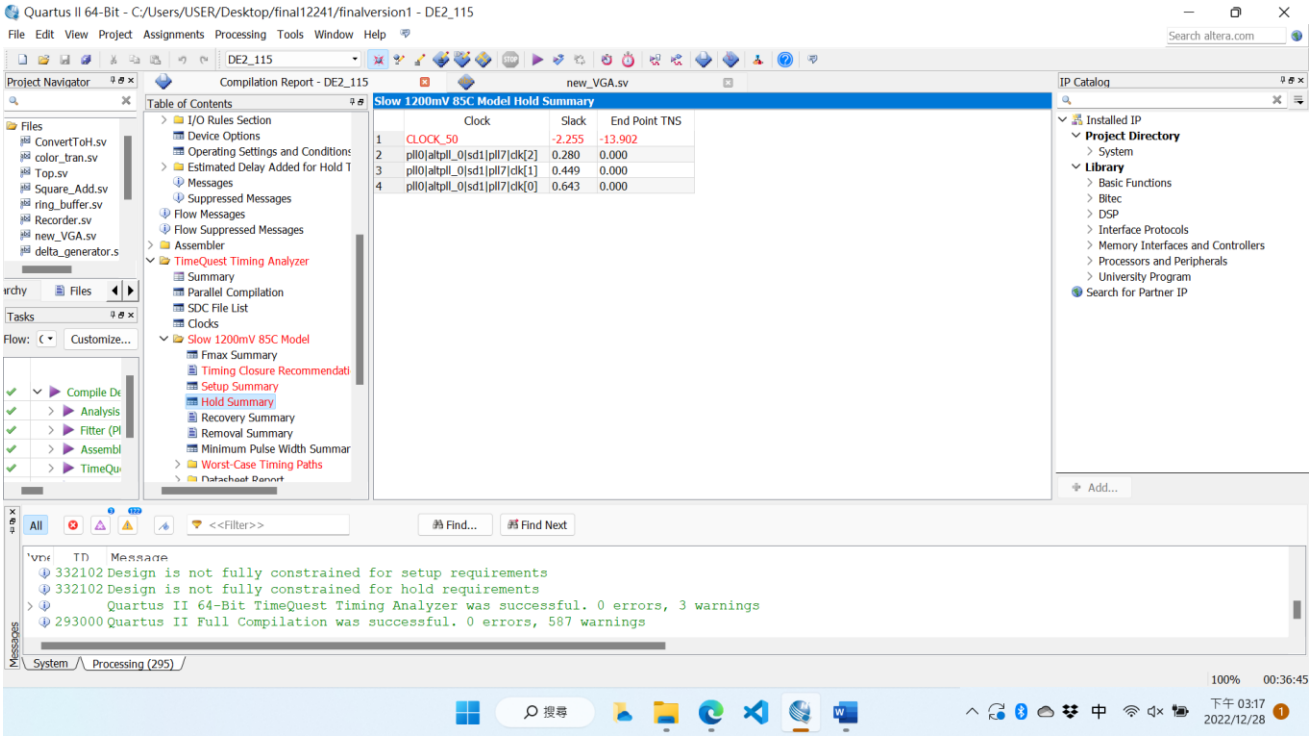
## A. *Fitter Summary*

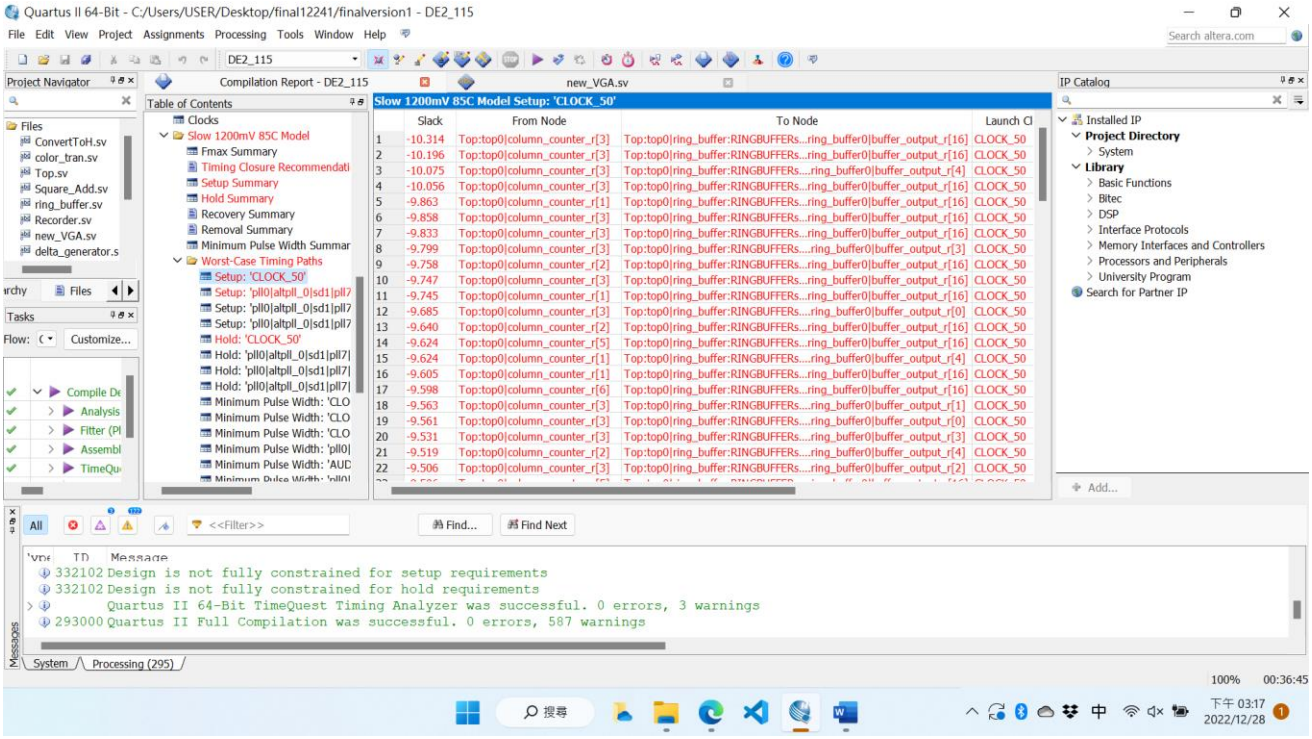


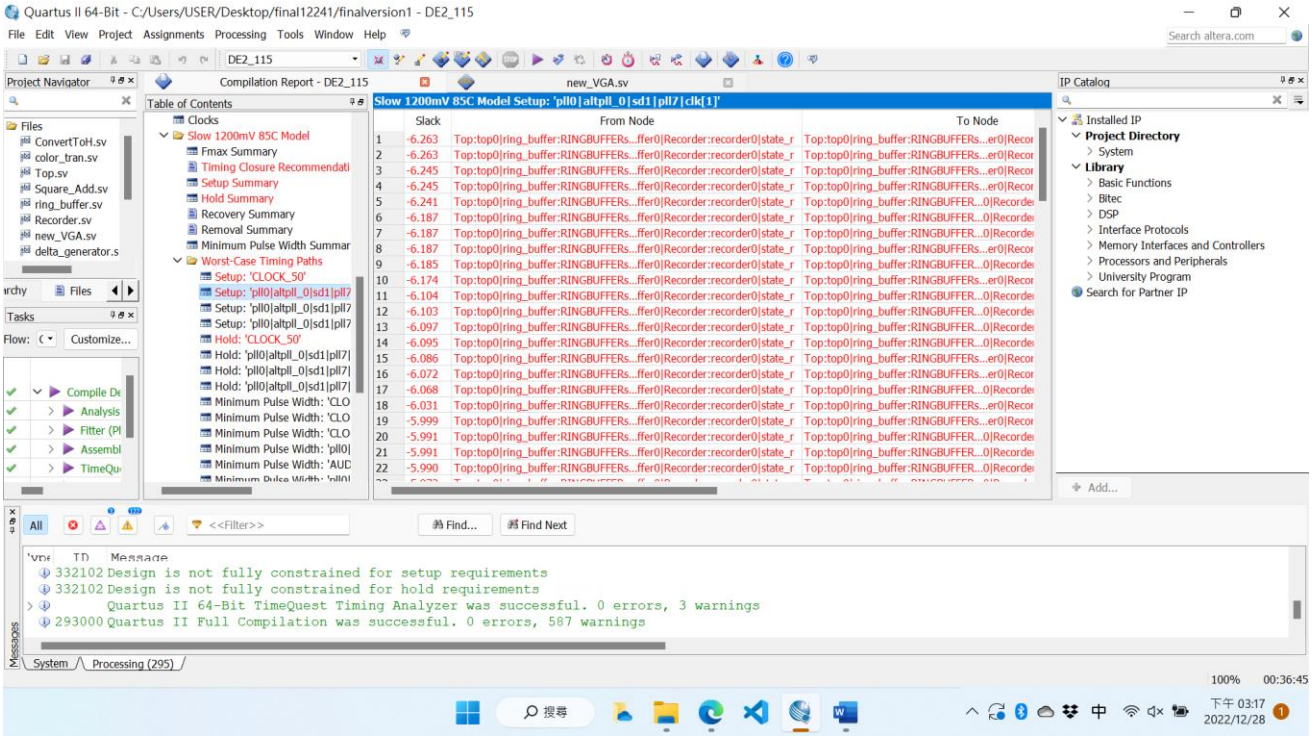

## B. *Timing Analyzer*

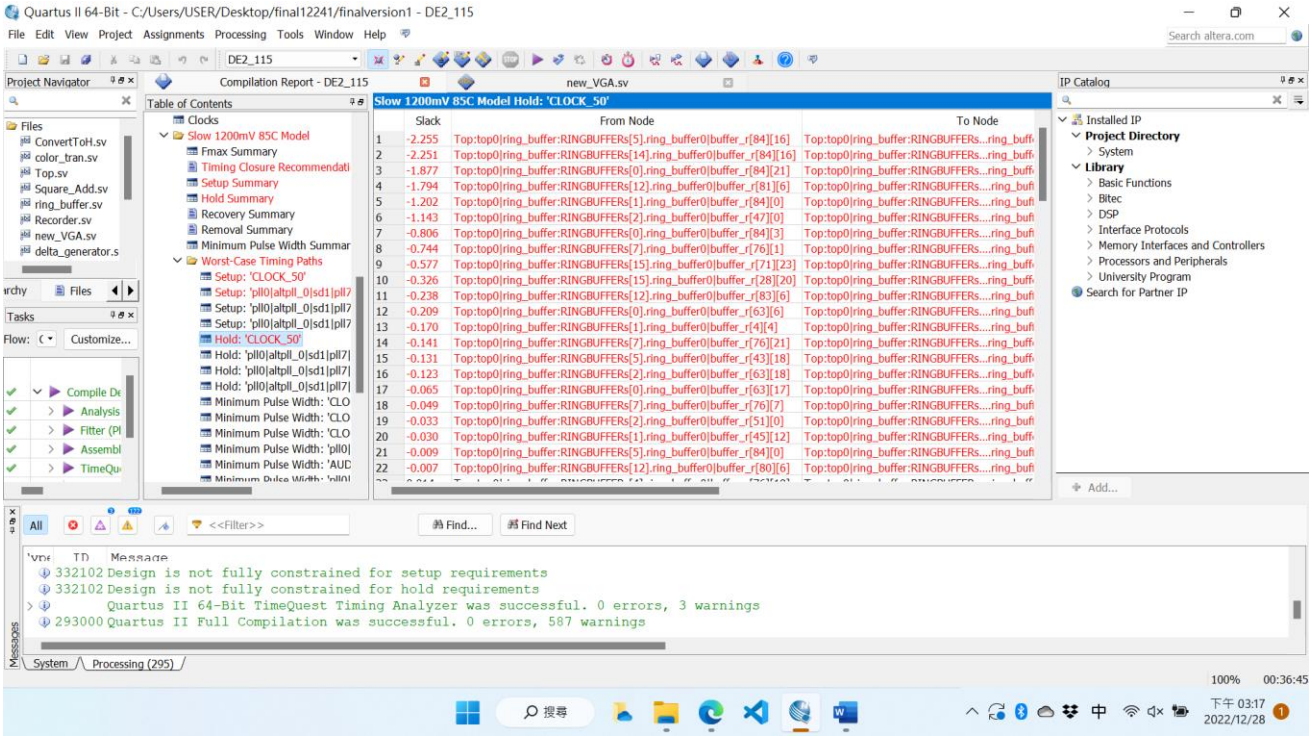


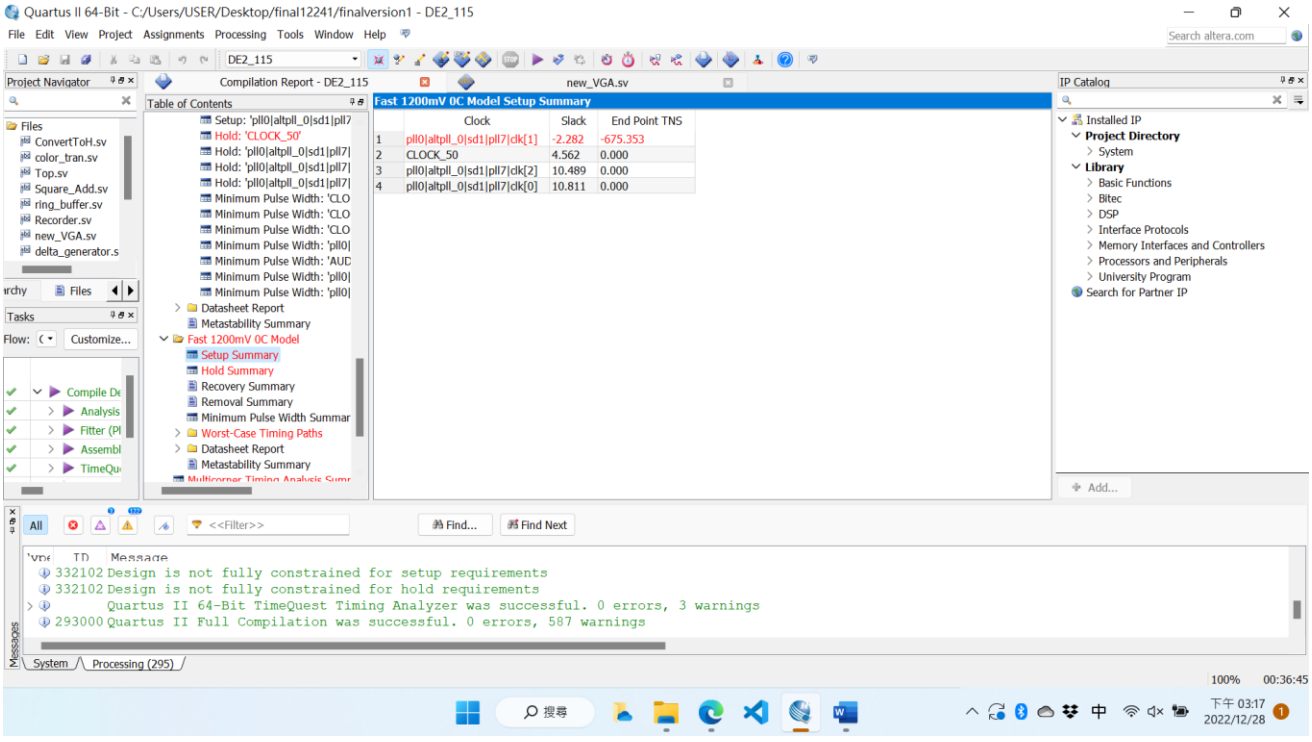

Quartus II 64-Bit - C:/Users/USER/Desktop/final12241/finalversion1 - DE2_115
Slow 1200mV 85C Model Setup Summary
Clock Slack End Point TNS
1 CLOCK_50 -10.314 -2879.213
2 pll0|altpll_0|sd1|pll7|clk[1] -6.263 -2049.915
3 pll0|altpll_0|sd1|pll7|clk[0] 1.637 0.000
4 pll0|altpll_0|sd1|pll7|clk[2] 2.979 0.000
332102 Design is not fully constrained for setup requirements
332102 Design is not fully constrained for hold requirements
Quartus II 64-Bit TimeQuest Timing Analyzer was successful. 0 errors, 3 warnings
293000 Quartus II Full Compilation was successful. 0 errors, 587 warnings

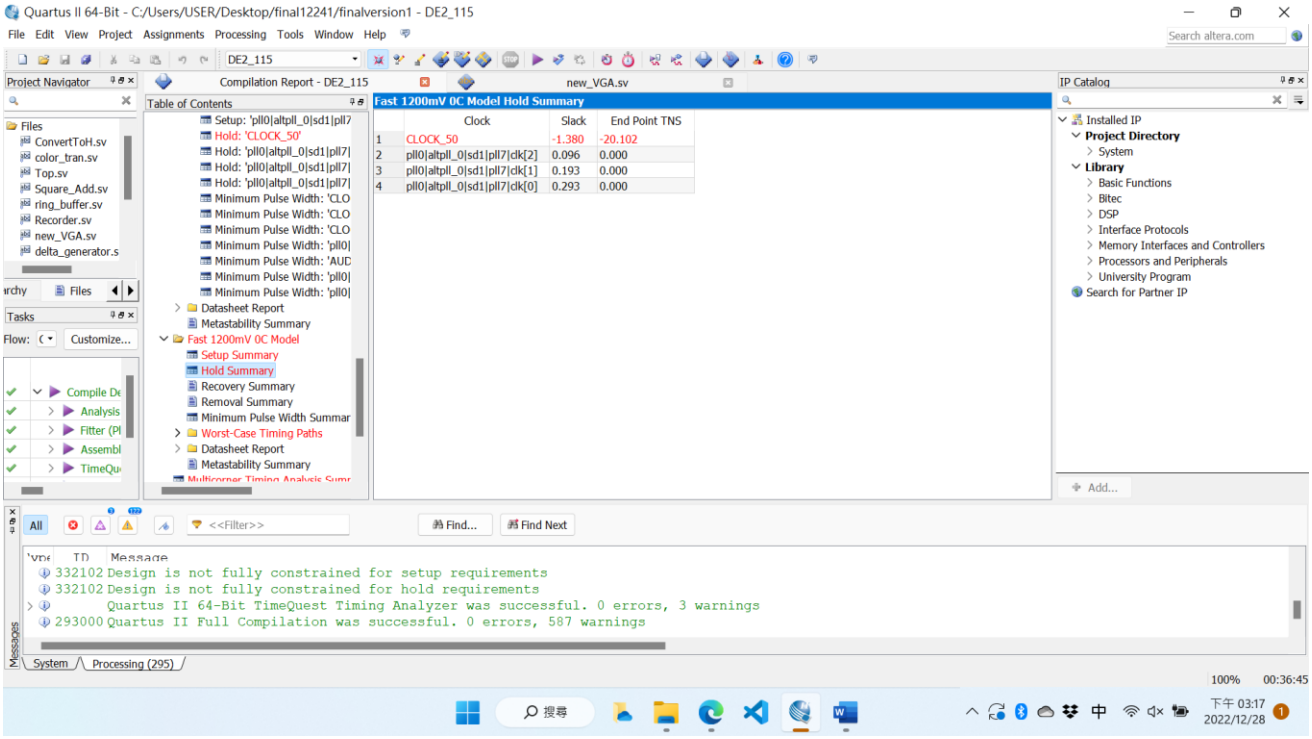

Quartus II 64-Bit - C:/Users/USER/Desktop/final12241/finalversion1 - DE2_115
Slow 1200mV 85C Model Hold Summary
Clock Slack End Point TNS
1 CLOCK_50 -2.255 -13.902
2 pll0|altpll_0|sd1|pll7|clk[2] 0.280 0.000
3 pll0|altpll_0|sd1|pll7|clk[1] 0.449 0.000
4 pll0|altpll_0|sd1|pll7|clk[0] 0.643 0.000
332102 Design is not fully constrained for setup requirements
332102 Design is not fully constrained for hold requirements
Quartus II 64-Bit TimeQuest Timing Analyzer was successful. 0 errors, 3 warnings
293000 Quartus II Full Compilation was successful. 0 errors, 587 warnings

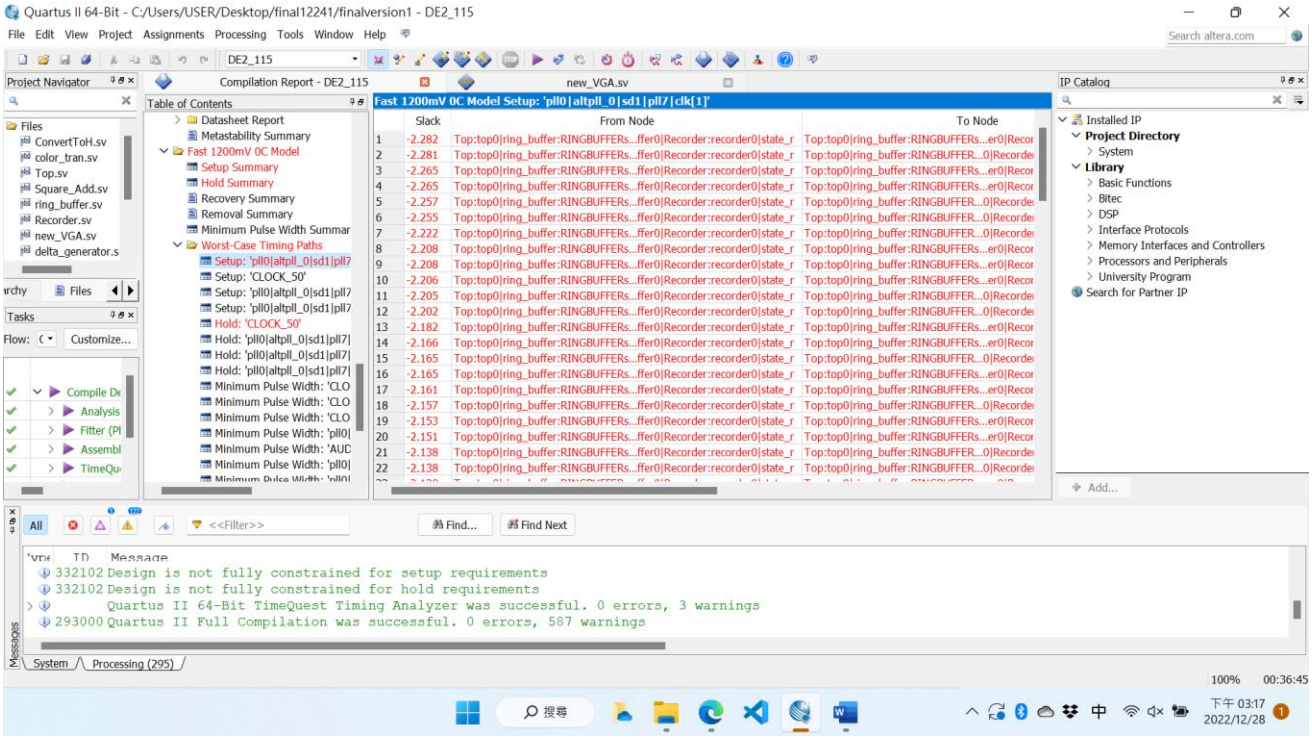

Quartus II 64-Bit - C:/Users/USER/Desktop/final12241/finalversion1 - DE2_115
Slow 1200mV 85C Model Setup: 'CLOCK_50'
Slack
From Node
To Node
Launch Cl
-10.314
-10.196
-10.075
-10.056
-9.863
-9.858
-9.833
-9.799
-9.758
-9.747
-9.745
-9.685
-9.640
-9.624
-9.624
-9.605
-9.598
-9.563
-9.561
-9.531
-9.519
-9.506
CLOCK_50
332102 Design is not fully constrained for setup requirements
332102 Design is not fully constrained for hold requirements
Quartus II 64-Bit TimeQuest Timing Analyzer was successful. 0 errors, 3 warnings
293000 Quartus II Full Compilation was successful. 0 errors, 587 warnings

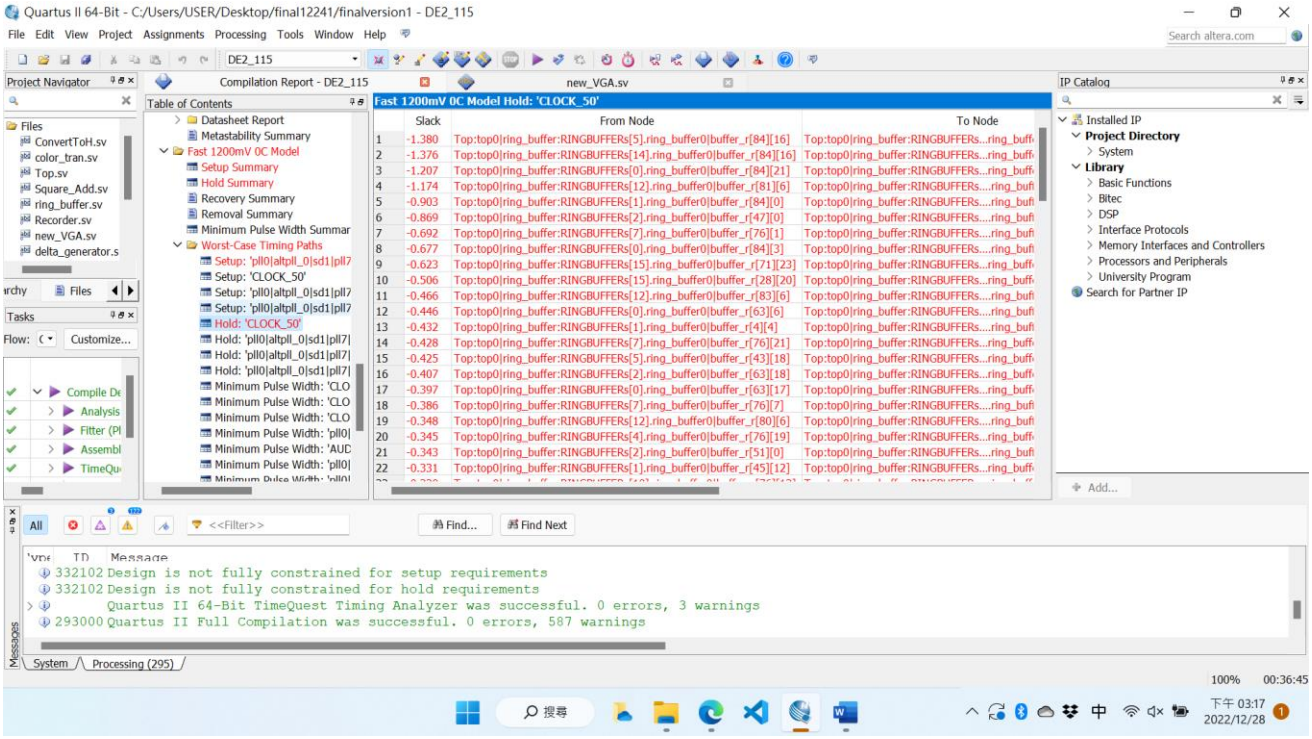

Quartus II 64-Bit - C:/Users/USER/Desktop/final12241/finalversion1 - DE2_115
Slow 1200mV 85C Model Setup: 'pll0|altpll_0|sd1|pll7|clk[1]'
Slack
From Node
To Node
-6.263
-6.245
-6.241
-6.187
-6.185
-6.174
-6.104
-6.103
-6.097
-6.095
-6.086
-6.072
-6.068
-6.031
-5.999
-5.991
-5.990
332102 Design is not fully constrained for setup requirements
332102 Design is not fully constrained for hold requirements
Quartus II 64-Bit TimeQuest Timing Analyzer was successful. 0 errors, 3 warnings
293000 Quartus II Full Compilation was successful. 0 errors, 587 warnings

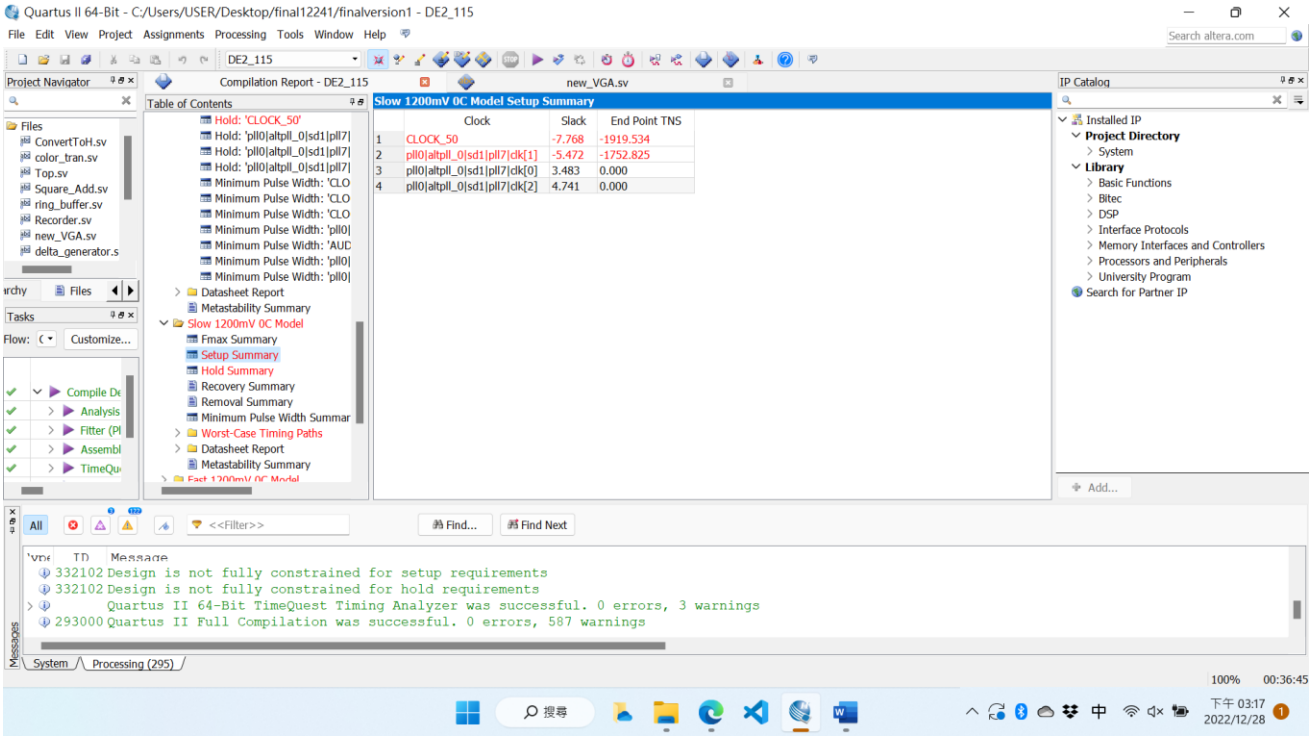

Quartus II 64-Bit - C:/Users/USER/Desktop/final12241/finalversion1 - DE2_115
Slow 1200mV 85C Model Hold: 'CLOCK_50'
Slack
From Node
To Node
-2.255
-2.251
-1.877
-1.794
-1.202
-1.143
-0.806
-0.744
-0.577
-0.326
-0.238
-0.209
-0.170
-0.141
-0.131
-0.123
-0.065
-0.049
-0.033
-0.030
-0.009
-0.007
332102 Design is not fully constrained for setup requirements
332102 Design is not fully constrained for hold requirements
Quartus II 64-Bit TimeQuest Timing Analyzer was successful. 0 errors, 3 warnings
293000 Quartus II Full Compilation was successful. 0 errors, 587 warnings

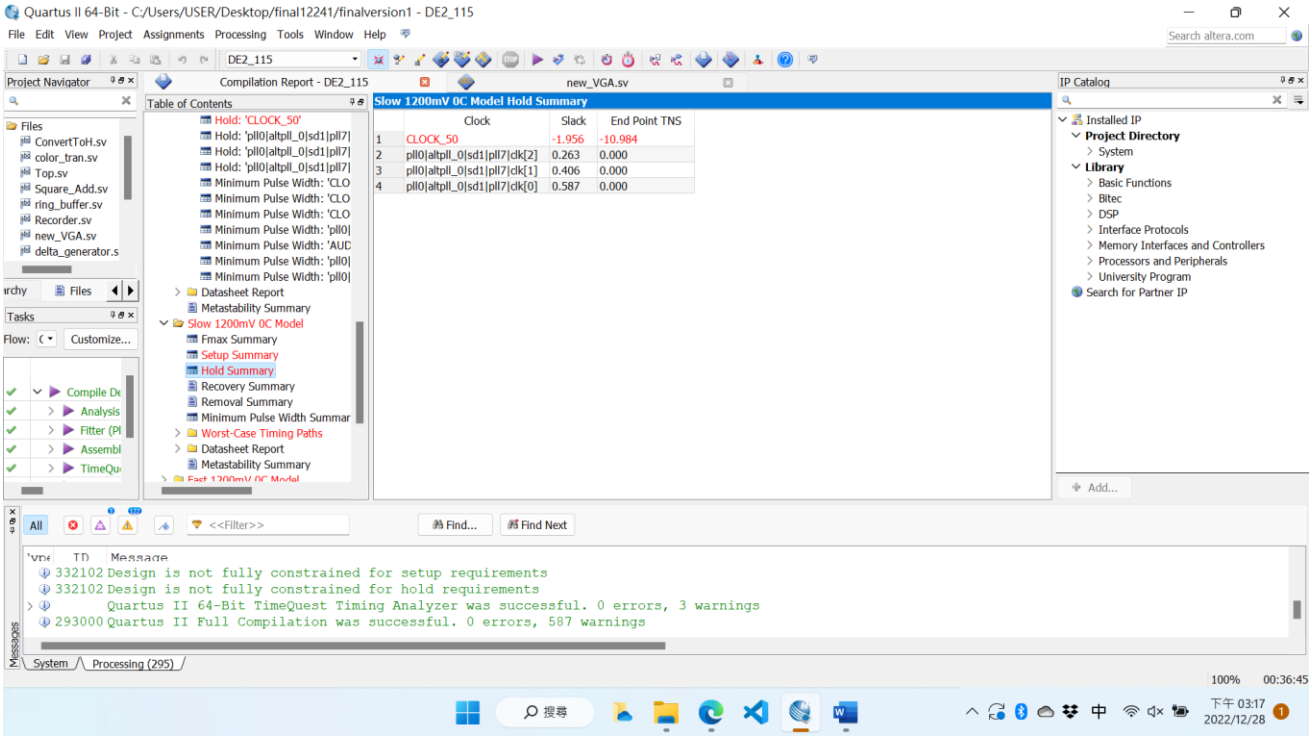

Quartus II 64-Bit - C:/Users/USER/Desktop/final12241/finalversion1 - DE2_115
Fast 1200mV 0C Model Setup Summary
Clock
Slack
End Point TNS
pll0|altpll_0|sd1|pll7|clk[1]
-2.282
-675.353
CLOCK_50
4.562
0.000
pll0|altpll_0|sd1|pll7|clk[2]
10.489
0.000
pll0|altpll_0|sd1|pll7|clk[0]
10.811
0.000
332102 Design is not fully constrained for setup requirements
332102 Design is not fully constrained for hold requirements
Quartus II 64-Bit TimeQuest Timing Analyzer was successful. 0 errors, 3 warnings
293000 Quartus II Full Compilation was successful. 0 errors, 587 warnings

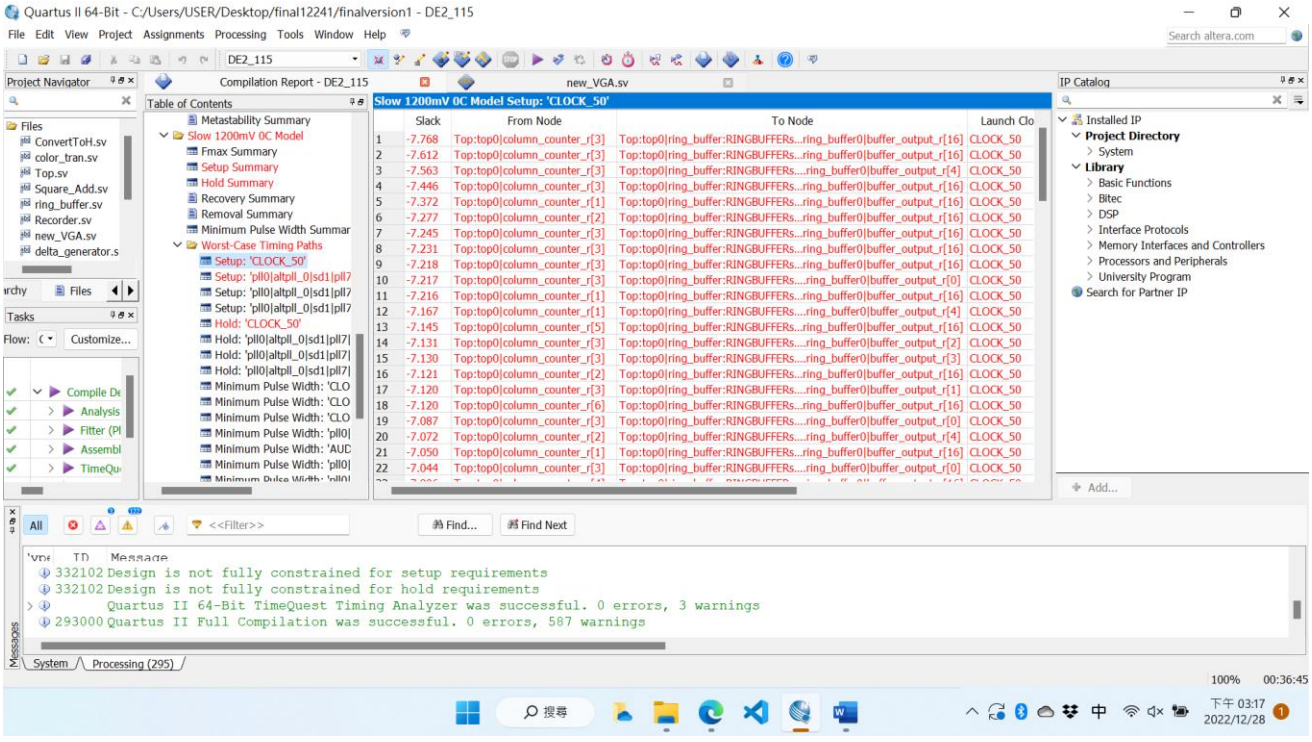

Quartus II 64-Bit - C:/Users/USER/Desktop/final12241/finalversion1 - DE2_115
Fast 1200mV 0C Model Hold Summary
Clock Slack End Point TNS
1 CLOCK_50 -1.380 -20.102
2 pll0|altpll_0|sd1|pll7|clk[2] 0.096 0.000
3 pll0|altpll_0|sd1|pll7|clk[1] 0.193 0.000
4 pll0|altpll_0|sd1|pll7|clk[0] 0.293 0.000
332102 Design is not fully constrained for setup requirements
332102 Design is not fully constrained for hold requirements
Quartus II 64-Bit TimeQuest Timing Analyzer was successful. 0 errors, 3 warnings
293000 Quartus II Full Compilation was successful. 0 errors, 587 warnings

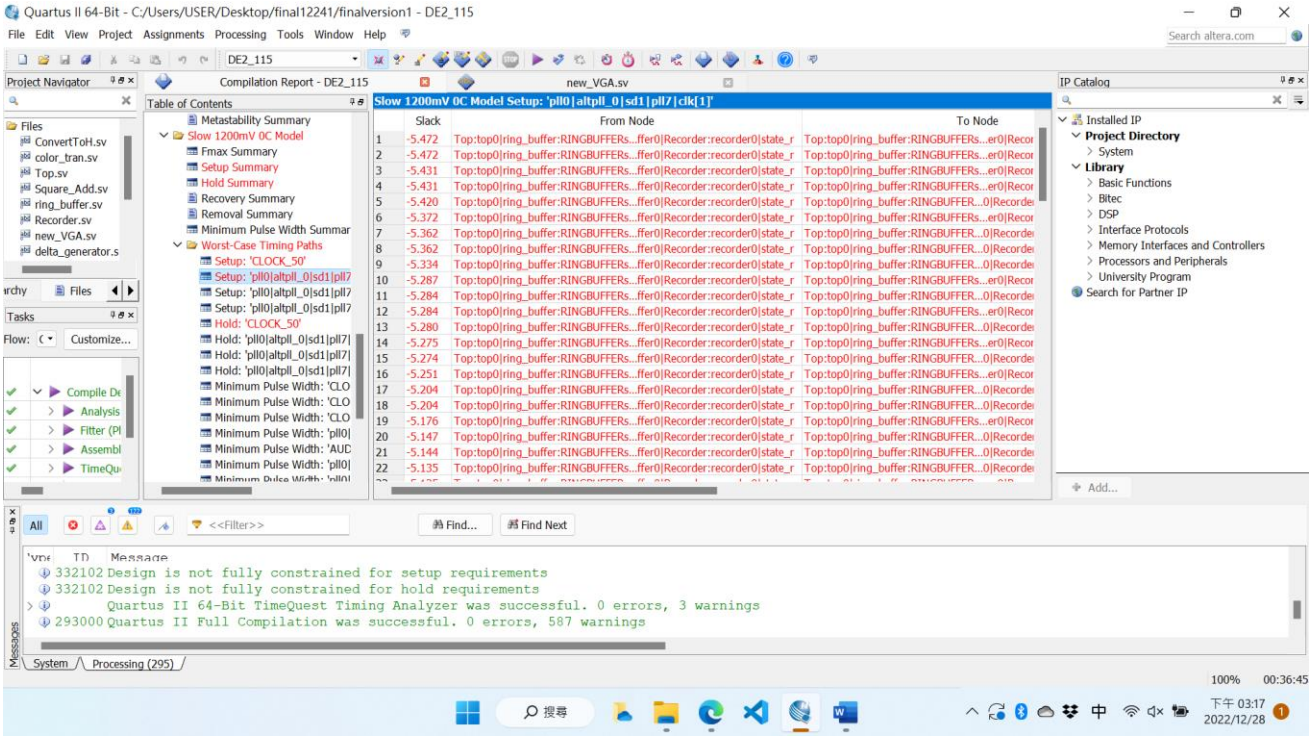

Quartus II 64-Bit - C:/Users/USER/Desktop/final12241/finalversion1 - DE2_115
Fast 1200mV 0C Model Setup: 'pll0|altpll_0|sd1|pll7|clk[1]'
Slack From Node To Node
1 -2.282 Top:top0|ring_buffer:RINGBUFFERs...ffer0|Recorder:recorder0|state_r Top:top0|ring_buffer:RINGBUFFERs...er0|Recor
2 -2.281 Top:top0|ring_buffer:RINGBUFFERs...ffer0|Recorder:recorder0|state_r Top:top0|ring_buffer:RINGBUFFER...0|Recorde
3 -2.265 Top:top0|ring_buffer:RINGBUFFERs...ffer0|Recorder:recorder0|state_r Top:top0|ring_buffer:RINGBUFFERs...er0|Recor
4 -2.265 Top:top0|ring_buffer:RINGBUFFERs...ffer0|Recorder:recorder0|state_r Top:top0|ring_buffer:RINGBUFFERs...er0|Recor
5 -2.257 Top:top0|ring_buffer:RINGBUFFERs...ffer0|Recorder:recorder0|state_r Top:top0|ring_buffer:RINGBUFFER...0|Recorde
6 -2.255 Top:top0|ring_buffer:RINGBUFFERs...ffer0|Recorder:recorder0|state_r Top:top0|ring_buffer:RINGBUFFERs...er0|Recor
7 -2.222 Top:top0|ring_buffer:RINGBUFFERs...ffer0|Recorder:recorder0|state_r Top:top0|ring_buffer:RINGBUFFER...0|Recorde
8 -2.208 Top:top0|ring_buffer:RINGBUFFERs...ffer0|Recorder:recorder0|state_r Top:top0|ring_buffer:RINGBUFFERs...er0|Recor
9 -2.208 Top:top0|ring_buffer:RINGBUFFERs...ffer0|Recorder:recorder0|state_r Top:top0|ring_buffer:RINGBUFFERs...er0|Recor
10 -2.206 Top:top0|ring_buffer:RINGBUFFERs...ffer0|Recorder:recorder0|state_r Top:top0|ring_buffer:RINGBUFFERs...er0|Recor
11 -2.205 Top:top0|ring_buffer:RINGBUFFERs...ffer0|Recorder:recorder0|state_r Top:top0|ring_buffer:RINGBUFFER...0|Recorde
12 -2.202 Top:top0|ring_buffer:RINGBUFFERs...ffer0|Recorder:recorder0|state_r Top:top0|ring_buffer:RINGBUFFER...0|Recorde
13 -2.182 Top:top0|ring_buffer:RINGBUFFERs...ffer0|Recorder:recorder0|state_r Top:top0|ring_buffer:RINGBUFFERs...er0|Recor
14 -2.166 Top:top0|ring_buffer:RINGBUFFERs...ffer0|Recorder:recorder0|state_r Top:top0|ring_buffer:RINGBUFFERs...er0|Recor
15 -2.165 Top:top0|ring_buffer:RINGBUFFERs...ffer0|Recorder:recorder0|state_r Top:top0|ring_buffer:RINGBUFFER...0|Recorde
16 -2.165 Top:top0|ring_buffer:RINGBUFFERs...ffer0|Recorder:recorder0|state_r Top:top0|ring_buffer:RINGBUFFERs...er0|Recor
17 -2.161 Top:top0|ring_buffer:RINGBUFFERs...ffer0|Recorder:recorder0|state_r Top:top0|ring_buffer:RINGBUFFERs...er0|Recor
18 -2.157 Top:top0|ring_buffer:RINGBUFFERs...ffer0|Recorder:recorder0|state_r Top:top0|ring_buffer:RINGBUFFER...0|Recorde
19 -2.153 Top:top0|ring_buffer:RINGBUFFERs...ffer0|Recorder:recorder0|state_r Top:top0|ring_buffer:RINGBUFFERs...er0|Recor
20 -2.151 Top:top0|ring_buffer:RINGBUFFERs...ffer0|Recorder:recorder0|state_r Top:top0|ring_buffer:RINGBUFFERs...er0|Recor
21 -2.138 Top:top0|ring_buffer:RINGBUFFERs...ffer0|Recorder:recorder0|state_r Top:top0|ring_buffer:RINGBUFFER...0|Recorde
22 -2.138 Top:top0|ring_buffer:RINGBUFFERs...ffer0|Recorder:recorder0|state_r Top:top0|ring_buffer:RINGBUFFER...0|Recorde
332102 Design is not fully constrained for setup requirements
332102 Design is not fully constrained for hold requirements
Quartus II 64-Bit TimeQuest Timing Analyzer was successful. 0 errors, 3 warnings
293000 Quartus II Full Compilation was successful. 0 errors, 587 warnings

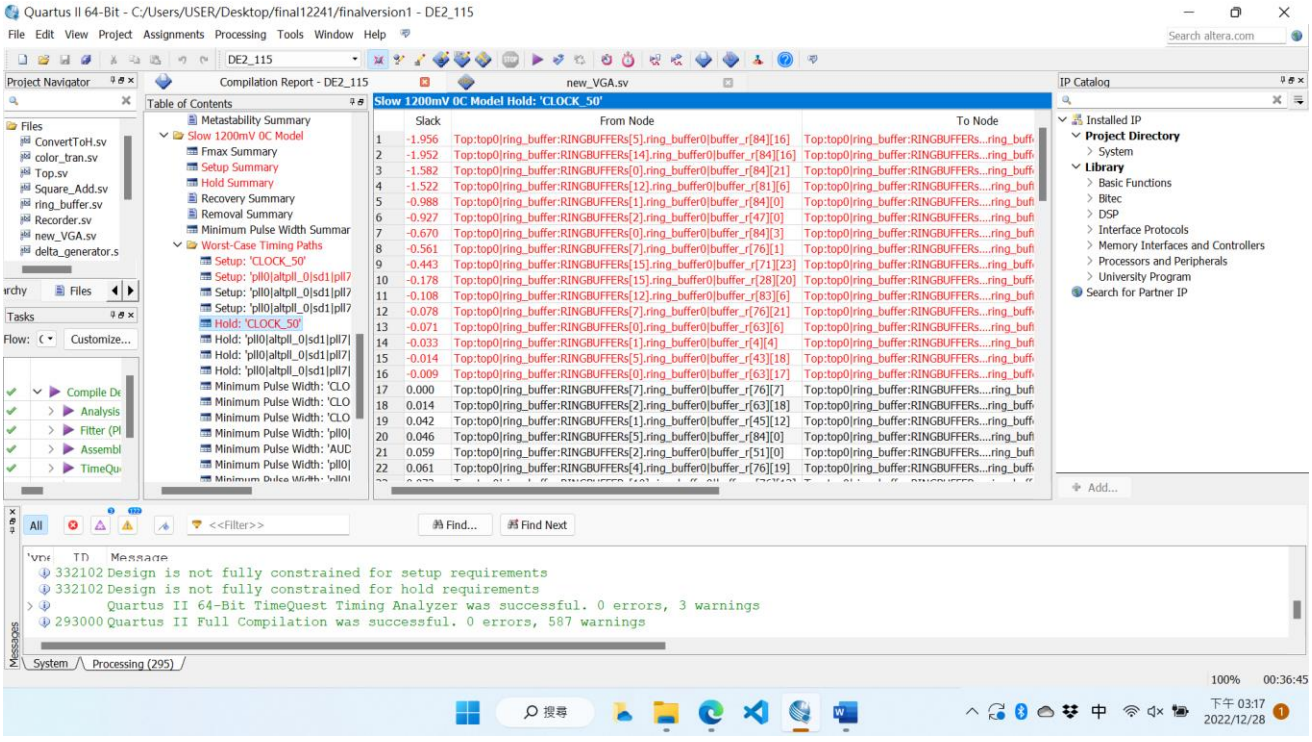

Quartus II 64-Bit - C:/Users/USER/Desktop/final12241/finalversion1 - DE2_115
Fast 1200mV 0C Model Hold: 'CLOCK_50'
Slack
From Node
To Node
332102 Design is not fully constrained for setup requirements
332102 Design is not fully constrained for hold requirements
Quartus II 64-Bit TimeQuest Timing Analyzer was successful. 0 errors, 3 warnings
293000 Quartus II Full Compilation was successful. 0 errors, 587 warnings

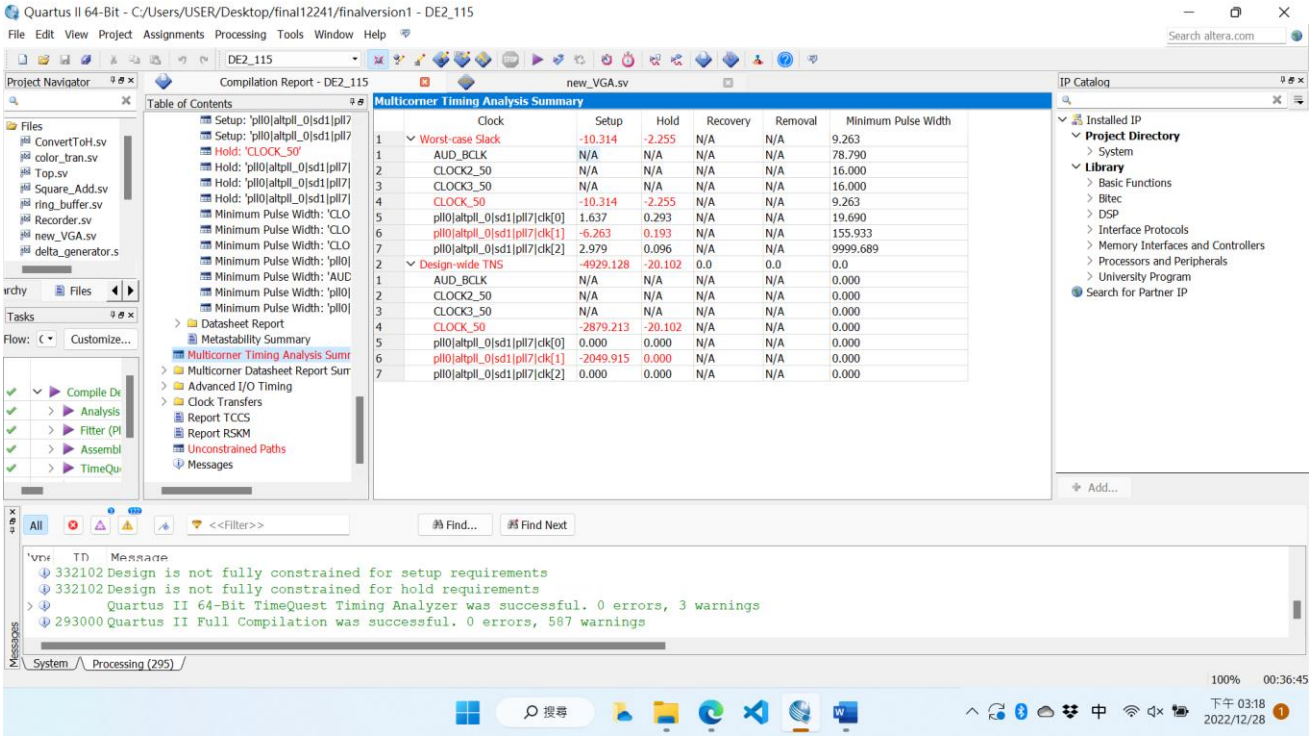

Quartus II 64-Bit - C:/Users/USER/Desktop/final12241/finalversion1 - DE2_115
Slow 1200mV 0C Model Setup Summary
Clock
Slack
End Point TNS
CLOCK_50 -7.768 -1919.534
pll0|altpll_0|sd1|pll7|clk[1] -5.472 -1752.825
pll0|altpll_0|sd1|pll7|clk[0] 3.483 0.000
pll0|altpll_0|sd1|pll7|clk[2] 4.741 0.000
332102 Design is not fully constrained for setup requirements
332102 Design is not fully constrained for hold requirements
Quartus II 64-Bit TimeQuest Timing Analyzer was successful. 0 errors, 3 warnings
293000 Quartus II Full Compilation was successful. 0 errors, 587 warnings

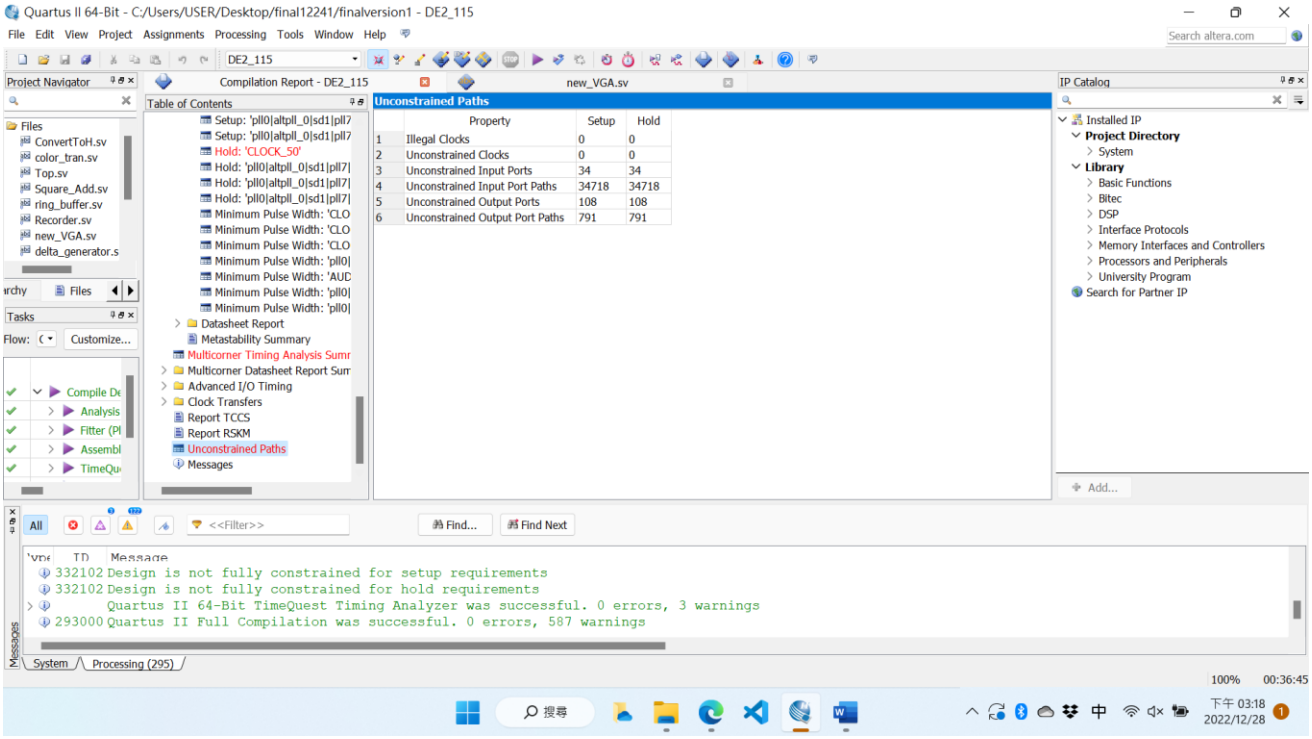

Quartus II 64-Bit - C:/Users/USER/Desktop/final12241/finalversion1 - DE2_115
Slow 1200mV 0C Model Hold Summary
Clock Slack End Point TNS
1 CLOCK_50 -1.956 -10.984
2 pll0|altpll_0|sd1|pll7|clk[2] 0.263 0.000
3 pll0|altpll_0|sd1|pll7|clk[1] 0.406 0.000
4 pll0|altpll_0|sd1|pll7|clk[0] 0.587 0.000
332102 Design is not fully constrained for setup requirements
332102 Design is not fully constrained for hold requirements
Quartus II 64-Bit TimeQuest Timing Analyzer was successful. 0 errors, 3 warnings
293000 Quartus II Full Compilation was successful. 0 errors, 587 warnings


Quartus II 64-Bit - C:/Users/USER/Desktop/final12241/finalversion1 - DE2_115
Slow 1200mV 0C Model Setup: 'CLOCK_50'
Slack From Node To Node Launch Clo
1 -7.768 Top:top0|column_counter_r[3] Top:top0|ring_buffer:RINGBUFFERs...ring_buffer0|buffer_output_r[16] CLOCK_50
2 -7.612 Top:top0|column_counter_r[3] Top:top0|ring_buffer:RINGBUFFERs...ring_buffer0|buffer_output_r[16] CLOCK_50
3 -7.563 Top:top0|column_counter_r[3] Top:top0|ring_buffer:RINGBUFFERs...ring_buffer0|buffer_output_r[4] CLOCK_50
4 -7.446 Top:top0|column_counter_r[3] Top:top0|ring_buffer:RINGBUFFERs...ring_buffer0|buffer_output_r[16] CLOCK_50
5 -7.372 Top:top0|column_counter_r[1] Top:top0|ring_buffer:RINGBUFFERs...ring_buffer0|buffer_output_r[16] CLOCK_50
6 -7.277 Top:top0|column_counter_r[2] Top:top0|ring_buffer:RINGBUFFERs...ring_buffer0|buffer_output_r[16] CLOCK_50
7 -7.245 Top:top0|column_counter_r[3] Top:top0|ring_buffer:RINGBUFFERs...ring_buffer0|buffer_output_r[16] CLOCK_50
8 -7.231 Top:top0|column_counter_r[3] Top:top0|ring_buffer:RINGBUFFERs...ring_buffer0|buffer_output_r[16] CLOCK_50
9 -7.218 Top:top0|column_counter_r[3] Top:top0|ring_buffer:RINGBUFFERs...ring_buffer0|buffer_output_r[16] CLOCK_50
10 -7.217 Top:top0|column_counter_r[3] Top:top0|ring_buffer:RINGBUFFERs...ring_buffer0|buffer_output_r[0] CLOCK_50
11 -7.216 Top:top0|column_counter_r[1] Top:top0|ring_buffer:RINGBUFFERs...ring_buffer0|buffer_output_r[16] CLOCK_50
12 -7.167 Top:top0|column_counter_r[1] Top:top0|ring_buffer:RINGBUFFERs...ring_buffer0|buffer_output_r[4] CLOCK_50
13 -7.145 Top:top0|column_counter_r[5] Top:top0|ring_buffer:RINGBUFFERs...ring_buffer0|buffer_output_r[16] CLOCK_50
14 -7.131 Top:top0|column_counter_r[3] Top:top0|ring_buffer:RINGBUFFERs...ring_buffer0|buffer_output_r[2] CLOCK_50
15 -7.130 Top:top0|column_counter_r[3] Top:top0|ring_buffer:RINGBUFFERs...ring_buffer0|buffer_output_r[3] CLOCK_50
16 -7.121 Top:top0|column_counter_r[2] Top:top0|ring_buffer:RINGBUFFERs...ring_buffer0|buffer_output_r[16] CLOCK_50
17 -7.120 Top:top0|column_counter_r[3] Top:top0|ring_buffer:RINGBUFFERs...ring_buffer0|buffer_output_r[1] CLOCK_50
18 -7.120 Top:top0|column_counter_r[6] Top:top0|ring_buffer:RINGBUFFERs...ring_buffer0|buffer_output_r[16] CLOCK_50
19 -7.087 Top:top0|column_counter_r[3] Top:top0|ring_buffer:RINGBUFFERs...ring_buffer0|buffer_output_r[0] CLOCK_50
20 -7.072 Top:top0|column_counter_r[2] Top:top0|ring_buffer:RINGBUFFERs...ring_buffer0|buffer_output_r[4] CLOCK_50
21 -7.050 Top:top0|column_counter_r[1] Top:top0|ring_buffer:RINGBUFFERs...ring_buffer0|buffer_output_r[16] CLOCK_50
22 -7.044 Top:top0|column_counter_r[3] Top:top0|ring_buffer:RINGBUFFERs...ring_buffer0|buffer_output_r[0] CLOCK_50
332102 Design is not fully constrained for setup requirements
332102 Design is not fully constrained for hold requirements
Quartus II 64-Bit TimeQuest Timing Analyzer was successful. 0 errors, 3 warnings
293000 Quartus II Full Compilation was successful. 0 errors, 587 warnings


Quartus II 64-Bit - C:/Users/USER/Desktop/final12241/finalversion1 - DE2_115
Slow 1200mV 0C Model Setup: 'pll0|altpll_0|sd1|pll7|clk[1]'
332102 Design is not fully constrained for setup requirements
332102 Design is not fully constrained for hold requirements
Quartus II 64-Bit TimeQuest Timing Analyzer was successful. 0 errors, 3 warnings
293000 Quartus II Full Compilation was successful. 0 errors, 587 warnings


Quartus II 64-Bit - C:/Users/USER/Desktop/final12241/finalversion1 - DE2_115
Slow 1200mV 0C Model Hold: 'CLOCK_50'
332102 Design is not fully constrained for setup requirements
332102 Design is not fully constrained for hold requirements
Quartus II 64-Bit TimeQuest Timing Analyzer was successful. 0 errors, 3 warnings
293000 Quartus II Full Compilation was successful. 0 errors, 587 warnings




Some real time imaging videos:

https://youtu.be/gSOAMRXZsIY

https://youtu.be/PNLFveaGfNs

## VII. DISCUSSION

Several issues encountered during final projects includes:

- We design a microphone bracket by laser cutting and wired the microphone signal to solderless breadboard. For each microphone, there are 6 I/O port so it is hard for us to maintain the whole structure. Once a bug occurs, we have to test all wire in order to pick out the broken wire.
- We have 4 clock signal: 50M clock, 25M clock(for VGA), BCLK, LRCK so we have to be careful when we design our control signal. The low frequency may be influenced by high frequency controls and low frequency signal may not be trigger by a high frequency pulse control. It takes us a lot of time to design an architecture with simple control signals.
- The beamforming algorithm works only when the sound frequency is high enough, since low frequency sound does not have efficient demonstration of result. We use 1.7kHz in this project so we make a lot of noise in the classroom when we test our design.
- It is hard to understand what a sequence of sound data means. In the end, we use logic analyzer to export a sequence of sound data to csv file, then use Excel to draw the pattern.
- The presentation of sound source on the screen has clear sidelobes. After searching on the internet, we know that it is a normal phenomenon of this algorithm and the sidelobes can be curbed by increasing the number of microphones. However, due to the budget constraint, we didn't do this in the end.

## VIII. FINAL THOUGHTS

Some final thoughts of the DC Lab:

### 劉則暐

期末 project 真的與前面的 lab HW 非常不同，題目、演算法、架構、模組等都需要自己設計或組裝，做完的時候真的很有成就感也感覺自己進步了很多。

我覺得我在這次 project 中學到最多的是，如何切割一個硬體的大專案讓大家可以最有效率的分工，並讓 module 與 module 間的控制訊號越簡單越好。我們最初的設計就是因為控制訊號太過複雜，不僅很難 debug 也讓我們組員間的溝通變得很凌亂，大家實做的 module 架構也因為彼此的控制訊號而頻繁修改。在這次期末 project 中，我們總共設計了三個架構，最終版的架構相較於第一版可以說是簡單了不少，也讓我們在 debug 的時候輕鬆很多。一個好的架構設計，真的可以省去很多不必要的時間浪費。

### 張哲銘

這學期修完數位電路實驗真的收穫很多，除了扎實地走過整個硬體專案的開發流程，更能夠增加與組員合作開發的機會。這個期末專案一開始覺得蠻難做出來的，因為上述許多信號處理層面的內容是沒有辦法直觀看出相對應的成果，但幸好後續寫 code 都還算流暢，自己也第一次嘗試使用 Python 自動化生成出其他程式語言需要的 code。除此之外，版本控制也是這次學到一個很重要的內容，因為很常我們資訊更新不同步又沒有講清楚的情況下，可能一不小心就把有錯的檔案混到現在的專案當中了。但是以結果來說，我對於我們做出來的結果感到非常滿意，也很謝謝兩位組員這學期的凱瑞！

### 謝博揚

其實一開始選這個題目的時候，我對於會做出結果只有抱著 50%的信心，首先是因為硬體的限制，適合我們的麥克風真的非常難買，我們甚至是直到了定案的前一天半夜，才偶然發現這個麥克風可以用，不然本來已經要放棄這個主題了，因此最後有做出這樣的結果真的還算滿意但也有一點意外。不

過在做 final project 的時候也是遇到許多困難，在模擬和 debug 的時候可以說是用盡了電機系兩年多來所學吧，除了 verilog 外，matlab、python、邏輯分析儀、Excel 都是我們的工具，整體來說雖然中間過程的熬夜很痛苦，整個大架構改了三次，但也學到了很多知識，以及 debug 的耐心😊，數電實驗的確讓我有很多收穫並且留下一個難忘的回憶。

## REFERENCE

[1] B. Zimmermann and C. Studer, "FPGA-based real-time acoustic camera prototype," Proceedings of 2010 IEEE International Symposium on Circuits and Systems, 2010, pp. 1419-1419, doi: 10.1109/ISCAS.2010.5537301.

[2] B. da Silva, L. Segers, Y. Rasschaert, Q. Quevy, A. Braeken and A. Touhafi, "A Multimode SoC FPGA-Based Acoustic Camera for Wireless Sensor Networks," 2018 13th International Symposium on Reconfigurable Communication-centric Systems-on-Chip (ReCoSoC), 2018, pp. 1-8, doi: 10.1109/ReCoSoC.2018.8449381.