2017 Digital IC Design
Homework 6: Frequency Analysis System

NAME	林聖軒
Student ID	P76061425

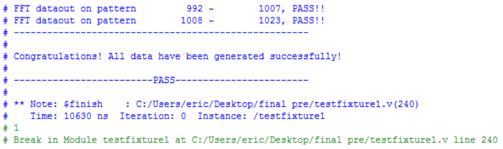
Simulation Result

Testfixture 1

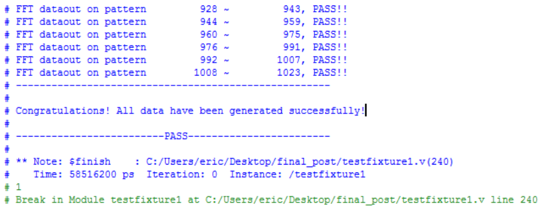
Functional RANK	A
Gate-level RANK	A
Gate-level simulation time	58516.2 ns

Testfixture 2

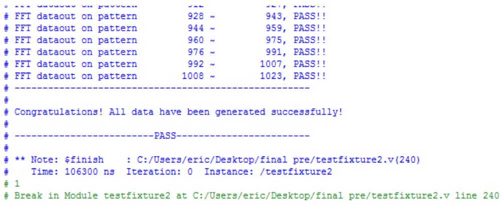
Functional RANK	A
Gate-level RANK	A
Gate-level simulation time	53100 ns



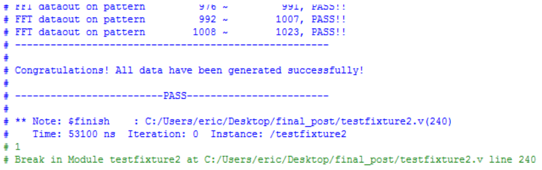
```
# FFT dataout on pattern        992 ~       1007, PASS!!
# FFT dataout on pattern       1008 ~       1023, PASS!!
# -----------------------------------------------------
#
# Congratulations! All data have been generated successfully!
#
# -------------------------PASS------------------------
#
# ** Note: $finish    : C:/Users/eric/Desktop/final pre/testfixture1.v(240)
#    Time: 10630 ns  Iteration: 0  Instance: /testfixture1
# 1
# Break in Module testfixture1 at C:/Users/eric/Desktop/final pre/testfixture1.v line 240
```



```
# FFT dataout on pattern        928 ~        943, PASS!!
# FFT dataout on pattern        944 ~        959, PASS!!
# FFT dataout on pattern        960 ~        975, PASS!!
# FFT dataout on pattern        976 ~        991, PASS!!
# FFT dataout on pattern        992 ~       1007, PASS!!
# FFT dataout on pattern       1008 ~       1023, PASS!!
# -----------------------------------------------------
#
# Congratulations! All data have been generated successfully!
#
# -------------------------PASS------------------------
#
# ** Note: $finish    : C:/Users/eric/Desktop/final_post/testfixture1.v(240)
#    Time: 58516200 ps  Iteration: 0  Instance: /testfixture1
# 1
# Break in Module testfixture1 at C:/Users/eric/Desktop/final_post/testfixture1.v line 240
```



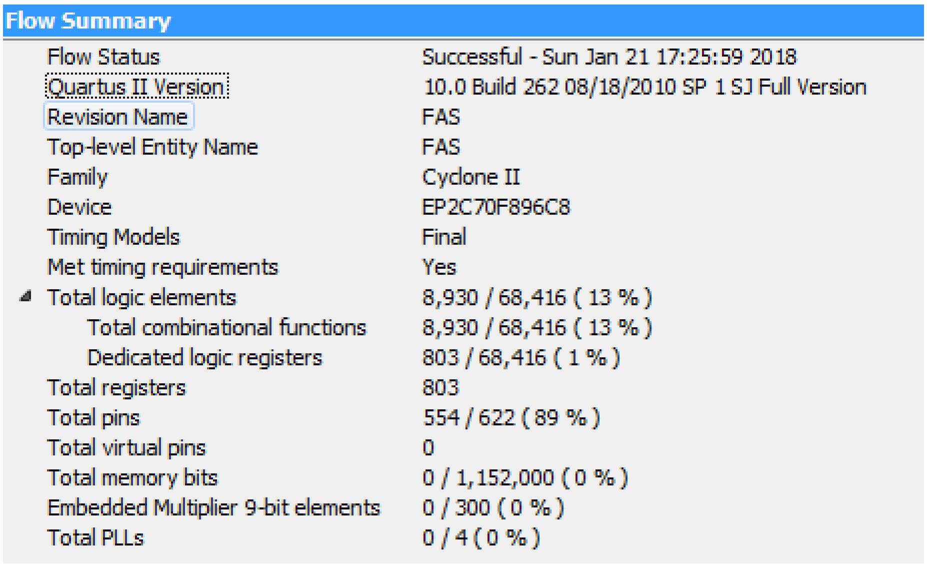
```
# FFT dataout on pattern        912 ~        927, PASS!!
# FFT dataout on pattern        928 ~        943, PASS!!
# FFT dataout on pattern        944 ~        959, PASS!!
# FFT dataout on pattern        960 ~        975, PASS!!
# FFT dataout on pattern        976 ~        991, PASS!!
# FFT dataout on pattern        992 ~       1007, PASS!!
# FFT dataout on pattern       1008 ~       1023, PASS!!
# -----------------------------------------------------
#
# Congratulations! All data have been generated successfully!
#
# -------------------------PASS------------------------
#
# ** Note: $finish    : C:/Users/eric/Desktop/final pre/testfixture2.v(240)
#    Time: 106300 ns  Iteration: 0  Instance: /testfixture2
# 1
# Break in Module testfixture2 at C:/Users/eric/Desktop/final pre/testfixture2.v line 240
```



```
# FFT dataout on pattern        976 ~        991, PASS!!
# FFT dataout on pattern        992 ~       1007, PASS!!
# FFT dataout on pattern       1008 ~       1023, PASS!!
# -----------------------------------------------------
#
# Congratulations! All data have been generated successfully!
#
# -------------------------PASS------------------------
#
# ** Note: $finish    : C:/Users/eric/Desktop/final_post/testfixture2.v(240)
#    Time: 53100 ns  Iteration: 0  Instance: /testfixture2
# 1
# Break in Module testfixture2 at C:/Users/eric/Desktop/final_post/testfixture2.v line 240
```

Synthesis Result

Total logic elements	8930
Total memory bit	0
Embedded multiplier 9-bit element	0



Flow Summary

Flow Status	Successful - Sun Jan 21 17:25:59 2018
Quartus II Version	10.0 Build 262 08/18/2010 SP 1 SJ Full Version
Revision Name	FAS
Top-level Entity Name	FAS
Family	Cyclone II
Device	EP2C70F896C8
Timing Models	Final
Met timing requirements	Yes
Total logic elements	8,930 / 68,416 (13 %)
Total combinational functions	8,930 / 68,416 (13 %)
Dedicated logic registers	803 / 68,416 (1 %)
Total registers	803
Total pins	554 / 622 (89 %)
Total virtual pins	0
Total memory bits	0 / 1,152,000 (0 %)
Embedded Multiplier 9-bit elements	0 / 300 (0 %)
Total PLLs	0 / 4 (0 %)

Description of your design

本作業共使用了3個檔案，分別是FAS主模組，內含有FIR的電路，PU負責算單筆的fft_a及fft_b，FFT負責16點的快速傅利葉轉換。

FAS中的Fir電路部份使用一個陣列來將fir的input data存入循序電路中，再用一個組合電路將input data與相對應的參數相乘，並相加，最後把結果所需的bit位置拉出來給fir_d。

PU算fft_a及fft_b，該電路輸入為單點的快速傅利葉轉換X,Y、W的實部和虛部，並將輸入做適當乘法加法運算，最後輸出fft_a、fft_b。

FFT部份則先將FIR的結果從尾端存入陣列中，當做快速傅利葉轉換架構的輸入，並呼叫PU模組將16點的快速傅利葉轉換架構給予適當接線，最後依照需求把結果拉線出來。

Scoring = (Total logic elements + total memory bit + 9*embedded multiplier 9-bit element) × (longest gate-level simulation time in ns)