

WES237C: Project 4 FFT and OFDM Receiver

Yi Song

vis057@ucsd.edu

PID# A53266440

Sebastian Nevarez

senevare@ucsd.edu

PID# A59021322

1. Fast Fourier Transform (FFT)

Our 1024-point FFT design starts with a bit reverse function to change the order of the input IQ samples so that the output of the FFT will be in an increasing numerical order. We then set up 10 stages of the FFT algorithm to run sequentially to generate the output. We separated the first and last stage since we know the bound and so we can optimize further in this aspect.

The first optimization we did was to separate the input and output matrix of each function. By doing this we avoid data being read and written to the same matrix during the same cycle. Additionally, this decreases the II from 2 to 1 which we used ``#pragma pipeline II=1`` to do.

The next optimization done was to make VITIS understand our loop bounds. The majority of the code runs in nested for loops with the outer loop iterating butterfly times and the inner loop iterating 1024 times. As the outer loop varies with FFT stages, it was a good idea to flip the fixed bound of the inner loop and the varying bound of the outer loop, so VITIS has a better idea of how long the loop is going to run. At this stage of development, we noticed that the latency for the bit-reverse function was over 2 times that of the FFT stages.

Finally, using separate inputs and outputs in the bit-reverse function, we are able to remove memory swapping and reduce the II of the stage to 1. This improved the latency of all functions to about 1000 or less, achieving a throughput of $1/(1042 \cdot 10\text{ns}) = 95.97\text{kHz}$. See Figures 1 & 2 below for screenshots from our synthesis.

Timing Estimate

<

Figure 1: Performance and Resource Utilization

Summary					
Name	BRAM_18K	DSP	FF	LUT	URAM
DSP	-	-	-	-	-
Expression	-	-	0	132	-
FIFO	-	-	198	134	-
Instance	18	213	28283	42353	-
Memory	80	-	0	0	0
Multiplexer	-	-	-	198	-
Register	-	-	22	-	-
Total	98	213	28503	42817	0
Available	280	220	106400	53200	0
Utilization (%)	35	96	26	80	0

Figure 2: Resource Utilization Summary

2. OFDM Receiver

Similar to the FFT project, we started with the FFT bit-reverse function and FFT stages function copied to the OFDM project. The next thing we needed to do to complete the QPSK decode function was to decode the IQ samples into messages (integer values from 0 to 3). We created four if statements to check which quadrant the samples fall into and assigned different integer values to the samples that fell into each quadrant. The throughput calculation for our OFDM receiver is $1000/(1042 \cdot 10\text{ns}) = 95.97\text{kHz}$. See Figures 3 & 4 below for screenshots from our synthesis.

```
34994 match: i: 34984 golden: 0 output: 0
34995 match: i: 34985 golden: 1 output: 1
34996 match: i: 34986 golden: 2 output: 2
34997 match: i: 34987 golden: 3 output: 3
34998 match: i: 34988 golden: 0 output: 0
34999 match: i: 34989 golden: 1 output: 1
35000 match: i: 34990 golden: 2 output: 2
35001 match: i: 34991 golden: 3 output: 3
35002 match: i: 34992 golden: 0 output: 0
35003 match: i: 34993 golden: 1 output: 1
35004 match: i: 34994 golden: 2 output: 2
35005 match: i: 34995 golden: 3 output: 3
35006 match: i: 34996 golden: 0 output: 0
35007 match: i: 34997 golden: 1 output: 1
35008 match: i: 34998 golden: 2 output: 2
35009 match: i: 34999 golden: 3 output: 3
35010 Comparing against output data
35011 *****
35012 PASS: The output matches the golden output!
35013 *****
35014 INFO: [SIM 1] CSim done with 0 errors.
35015 INFO: [SIM 3] ***** CSIM finish *****
```

Figure 3: CSIM Output Verification

