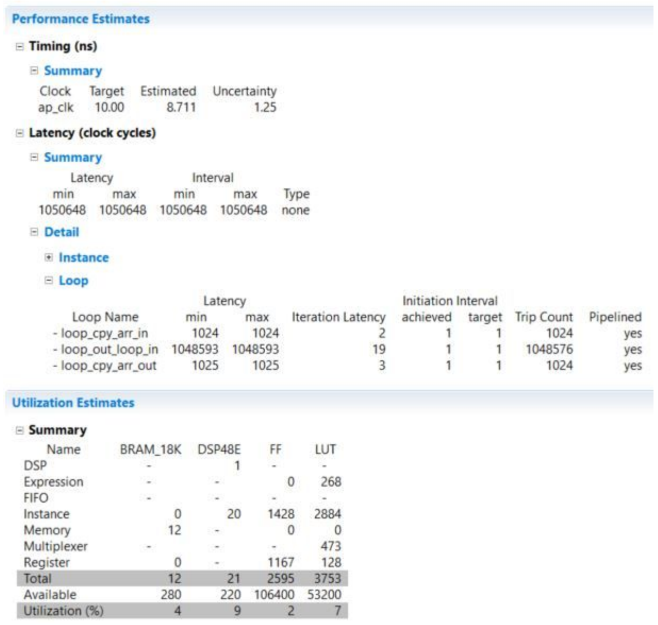
Patrick Hanrahan phanraha@eng.ucsd.edu
Benjamin Hobbs bhobbs@eng.ucsd.edu
Frank Chang fychang@eng.ucsd.edu

Project 4: FFT - OFDM Receiver

Direct Fourier Transform (DFT) utilizes matrix multiplication with a fourier transform matrix to get frequency data from time sampled data. For a 1024 DFT, this typically involves 1024*1024 = 1048576 operations (shown in Figure 1). Fast Fourier Transform (FFT) exploits patterns from this matrix multiplication process to increase performance at the cost of resource usage. A 1024 FFT typically involves 10*1024 = 10240 operations (shown in Figure 3). This is two order magnitude performance increase.

We first start by bit-reversing the address of each elements and shifting the input array according to the new address table, this will enable us to leverage patterns in the FFT later in the process. We then start by taking the 2 point FFT (butterfly) and passing the output to the next FFT (butterfly), which is a 4 point FFT. Each output of the nth point FFT will be phase corrected then passed to the next stage. The last stage will compute two 512 point FFT. There will be 10 stages of these FFT (butterfly) computed.

The OFDM demodulator will parse every sample and match them accordingly to their respective symbol based on which quadrant they are in on the constellation.


Performance Estimates

⊟ **Timing (ns)**

⊟ **Summary**

Clock	Target	Estimated	Uncertainty
ap_clk	10.00	8.711	1.25

⊟ **Latency (clock cycles)**

⊟ **Summary**

Latency		Interval		
min	max	min	max	Type
1050648	1050648	1050648	1050648	none

⊟ **Detail**

⊞ **Instance**

⊟ **Loop**

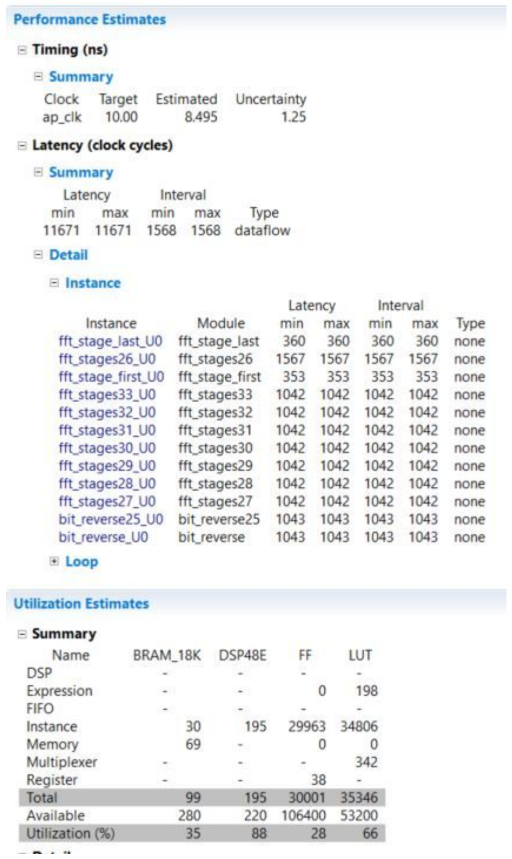
	Latency			Initiation Interval			
Loop Name	min	max	Iteration Latency	achieved	target	Trip Count	Pipelined
- loop_cpy_arr_in	1024	1024	2	1	1	1024	yes
- loop_out_loop_in	1048593	1048593	19	1	1	1048576	yes
- loop_cpy_arr_out	1025	1025	3	1	1	1024	yes

Utilization Estimates

⊟ **Summary**

Name	BRAM_18K	DSP48E	FF	LUT
DSP	-	1	-	-
Expression	-	-	0	268
FIFO	-	-	-	-
Instance	0	20	1428	2884
Memory	12	-	0	0
Multiplexer	-	-	-	473
Register	0	-	1167	128
Total	12	21	2595	3753
Available	280	220	106400	53200
Utilization (%)	4	9	2	7

Figure 1 - DFT performance.



Performance Estimates

Timing (ns)

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	10.00	8.495	1.25

Latency (clock cycles)

Summary

Latency		Interval		
min	max	min	max	Type
11671	11671	1568	1568	dataflow

Detail

Instance

		Latency		Interval		
Instance	Module	min	max	min	max	Type
fft_stage_last_U0	fft_stage_last	360	360	360	360	none
fft_stages26_U0	fft_stages26	1567	1567	1567	1567	none
fft_stage_first_U0	fft_stage_first	353	353	353	353	none
fft_stages33_U0	fft_stages33	1042	1042	1042	1042	none
fft_stages32_U0	fft_stages32	1042	1042	1042	1042	none
fft_stages31_U0	fft_stages31	1042	1042	1042	1042	none
fft_stages30_U0	fft_stages30	1042	1042	1042	1042	none
fft_stages29_U0	fft_stages29	1042	1042	1042	1042	none
fft_stages28_U0	fft_stages28	1042	1042	1042	1042	none
fft_stages27_U0	fft_stages27	1042	1042	1042	1042	none
bit_reverse25_U0	bit_reverse25	1043	1043	1043	1043	none
bit_reverse_U0	bit_reverse	1043	1043	1043	1043	none

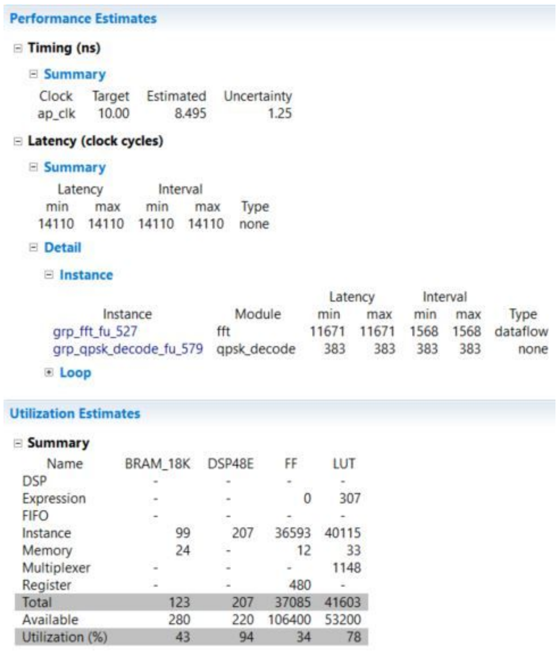
Loop

Utilization Estimates

Summary

Name	BRAM_18K	DSP48E	FF	LUT
DSP	-	-	-	-
Expression	-	-	0	198
FIFO	-	-	-	-
Instance	30	195	29963	34806
Memory	69	-	0	0
Multiplexer	-	-	-	342
Register	-	-	38	-
Total	99	195	30001	35346
Available	280	220	106400	53200
Utilization (%)	35	88	28	66

Figure 2: FFT performance.



Performance Estimates

Timing (ns)

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	10.00	8.495	1.25

Latency (clock cycles)

Summary

Latency		Interval		
min	max	min	max	Type
14110	14110	14110	14110	none

Detail

Instance

		Latency		Interval		
Instance	Module	min	max	min	max	Type
grp_fft_fu_527	fft	11671	11671	1568	1568	dataflow
grp_qpsk_decode_fu_579	qpsk_decode	383	383	383	383	none

Loop

Utilization Estimates

Summary

Name	BRAM_18K	DSP48E	FF	LUT
DSP	-	-	-	-
Expression	-	-	0	307
FIFO	-	-	-	-
Instance	99	207	36593	40115
Memory	24	-	12	33
Multiplexer	-	-	-	1148
Register	-	-	480	-
Total	123	207	37085	41603
Available	280	220	106400	53200
Utilization (%)	43	94	34	78

Figure 3: OFDM receiver performance.

There were five main optimization techniques we used in our design:

1. Pipeline - We pipelined our design for a factor of 1 whenever possible.
2. Loop unrolling and array partitioning - We unrolled our loops and partitioned the same number of factors to enable parallel access to the array. The *fft_stages()* did not utlize this technique but could have benefited from this. The variable *stage* passed to the *fft_stages()* function is different at each stage making the cyclic array partitioning difficult on the arrays.
3. Trigonometric LUT - The *fft_stages()* function utilizes trigonometric LUT. Instead of calling a performance and resource intensive trigonometric function from a math library, we are able to utilize our understanding of how the the nth stage affects our phase offset.
4. Bit shifting - *reverse_input()* function is called 1024 times and utilizes bitshifting fixed point integers to optimize on speed and performance.
5. Dataflow - We applied dataflow to the entire architecture which uses the fft stage functions to pipeline on a function level.

Our FFT has an interval latency of 1568 clock cycles at 8.495ns which yields 75.1k FFT operations per second therefore meeting design requirements.