Spring 2024
Project V2

This semester, you are designing a fft and slicing block for a simple OFDM communications system. There are many trade offs in such systems, but this system is designed to the following specifications:

- 128 point FFT (complex)
- Tones are at even numbered bins 4 to 50
    - 24 tones, each tone coded as two bits (4 levels)
        - 48 bits per symbol (One 128 point FFT)
- Tone 55 and 57 are fixed tones (Used for sync and other things)
    - Either may be present
    - Can be used to approximate 100% height

The fft works with complex inputs (From an IQ decoder) The 128 points are provided as a pair of complex fixed point signed 17 bit numbers with the binary point between bit 14 and 15. This is 2.15 notation. The two bits above the binary point allows the input to vary between -/+ 1/2^15 or +/- 1.999969.

It is up to you to come up with an FFT algorithm that meets the performance requirements. A quick google search shows 59,600,000 hits on fft. You should have limited challenges finding an FFT algorithm.

The multiplication/summations should be carried in 8.15 through the FFT for enough range at the end.

In a traditional 128 bit radix 2 DIT FFT, There are 7 levels each requiring 64 complex multiplies and 128 complex adds. This would require 64*7 complex multipliers (Each complex multiply requires 4 multiplies and two adds/subtracts) The project fft design is limited to 4 complex multipliers (total). The number of clocks required for the FFT is 64*7/4 or 112 clocks.

The DIT algorithm has a bit reversed permutation of the input samples.

The slicer looks at the energy of the used frequency bins. This is calculated by taking the sum of the square of I and Q. (Magnitude, or complex conjugate multiplication) In systems computing the abs, the square root would be taken. This value is compared to slicing points, so just the square can be used.

Each used frequency bin represents 2 bits of output. This is represented by 4 sliced amplitudes. The data is encoded at 0%, 33%, 66%, and 100% of full scale. Full scale can be found by the magnitude of frequency bin 55 or 57 being 100%. The energy levels are coded/decoded as follows:

Value	Coded	Condition
00	0%	<16% full scale
01	33%	>=16% and <49%%
10	66%	>=49% and <82%

11	100%	>=82% full scale

The output is in bytes. Each FFT block and slicing (finding the bits for a frequency) results in 48 bits. The low order two bits are in bin 4, and the upper bits are in bin 52 (24 bins total)

The module has the following ports:

Name	Dir	Bits	Comment
Clk	In	1	Positive edge system clock
Reset	In	1	Active high asynchronous reset
PushIn	In	1	Data is present on the input
FirstData	In	1	This is the first data in an FFT block
DinR	In	17	Real part of input (coded 2.15 2's complement)
DinI	In	17	Imaginary part of input (coded 2.15 2's complement)
PushOut	Out	1	Data is present on the output
DataOut	Out	48	Output data bits

A simple python model is provided (m3.py) so you can see the algorithm. It uses for loops, and is not implemented the same as the design. Python works in complex numbers, system verilog does not. You cannot cut and paste this code. There is much commented code which can be used to print or write debug information to a file. Running the python code will generate a function that will return the twiddle factors $e^{-i2\pi k/N}$ N is the number of points on each level. This table has N=128. To get the twiddle factors for the third level (4 unique numbers) each value index is k*64/(2^{Lvl}) assuming the Lvl starts with 0 performing adjacent 2 point butterflys. The above expression can be simplified by noting that 64 is 2^6 . This becomes k*$^{26-Lvl}$ then gives k<<(6-Lvl).