# Device Resources - M2S010/M2GL010

Existing Smartfusion2 device: M2S010/M2GL010

|  |  |
| --- | --- |
| Resource | Available |
| LSRAM 18432bit (512x36, 1024x18) | 2 x ? = 21 |
| µSRAM 64 x 18 bit | 2 x 11 = 22 |
| MACC DSP 18x18 signed mult, 9x9+9x9 DOTP | 2 x 11 = 22 |

# FFT

## Math

## Structure

### Big Picture

### Stage

### Butterfly

## Determining Data Width

The 3 considerations for this project are:

1. Data storage
2. Data math (mainly multiplication)
3. Result accuracy

### 1 Data Storage

µSRAM blocks in Smartfusion2 devices can store up to 18 bits in a single memory location and have dual read ports. In order to maximize resource utilization I’ve elected to store 1 complex number in a µSRAM location as concatenated signed bits. e.g.

Where & indicates concatenation of bits, RS & RD represent the signed 9 bit real value of a complex number, and IS & ID represent the signed 9 bit imaginary value of a complex number.

This would allow 1 µSRAM location to store 1 complex number used in FFT calculations and the dual port nature of the µSRAM block to feed 1 FFT butterfly in a single clock cycle.

### 2 Data Math

The DSP math blocks within a Smartfusion2 device allows for up to 18bit x 18bit signed multiplications. This is more than enough data width for the data being stored in RAM and indeed is more than the data width I expect to be sending to this FFT.

More interestingly is the Dot Product (DOTP) mode of the DSP blocks. This allows a single DSP block to perform the calculation:

Where A, B, C, and D are 9 bit signed numbers and P is the signed 18 bit result.

This is useful for finding the complex magnitude of the result so it can be sent to a human readable interface (LCD screen) or some other use. This also fits very nicely with the configuration of the data in the µSRAM blocks and all told allows 1 DSP block (fully utilized in DOTP mode) to provide the final complex magnitude result for 1 frequency bin.

Having decided that the data within the FFT will occupy pairs of 9 bit signed numbers. We must work backwards and determine the input data widths.

|  |  |
| --- | --- |
| Decimal | Binary |
| 255 | 1111\_1111 |
| 65025 | 1111\_1110\_0000\_0001 |

Remember that the twiddle factors represent cosine and sine values that range from 1 to -1. This is key. When 2 numbers of 8 bits are multiplied together, the result will be a 16 bit number:

More generally the bit width is but we’re intending to use same length numbers.

Therefore, on any butterfly multiplication we are multiplying our data at 9bits signed with a sine or cosine value that has been scaled up (literally or ) to be a 9bit signed number. However, as mentioned before, our sine and cosine values are really only between 1 and -1, I have scaled them up so they can be represented in an integer format. Therefore, each result of the multiplication can be divided by the same amount:

;

;

;

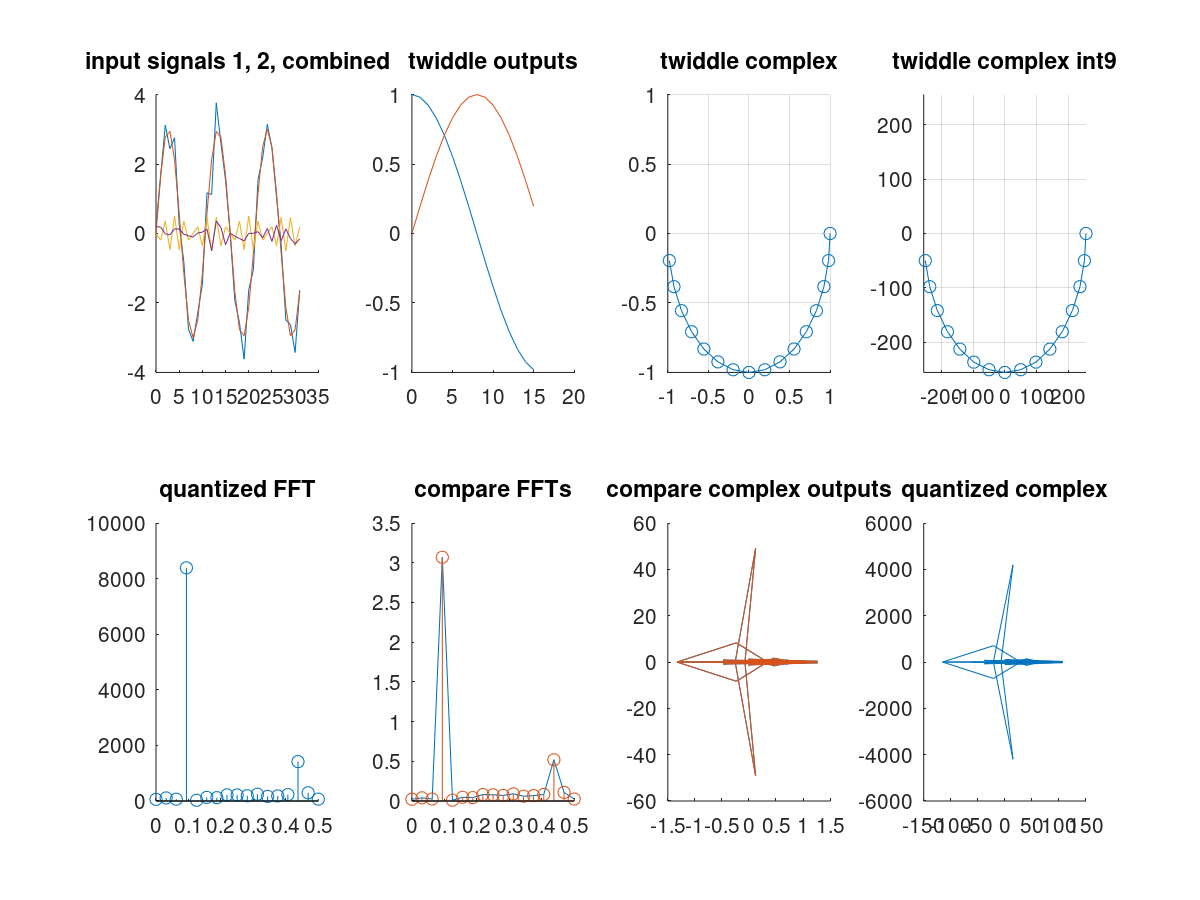
;

;

As can be seen this is not an exact process but the results should remain within the resolution of the 9bits signed, it’s the best that can be done.

### 3 Result Accuracy

Limiting performing on an FFT transform on 5 bit signed data seems like it would be limiting.



* Data bits = 8
* Input signals:
* Twiddle outputs:
  + Visualize the parts of the Cosine and Sine used as twiddle factors
* Twiddle complex:
  + Cosine and Sine values represent real and imaginary values respectively of complex numbers. This plot visualizes the twiddle factors used along the unit circle.
* Twiddle complex int9:
  + Cosine and Sine values quantized to represent 9bit signed values. This plot visualizes the scaled and quantized twiddle factors along the scaled unit circle.
* Quantized FFT:
  + Results of GNU Octave (Matlab) script to determine FFT results using integer operations. This is also scaled to my desired output resolution of 5 bits.
* Compare FFT:
  + Results of the built in FFT() function as well as a floating point version of my FFT implementation. My FFT version was then modified to produce the quantized FFT plot.
* Compare Complex Outputs:
  + Similar to the Compare FFT plot, this takes the real and imaginary parts of the FFT results and graphs them, just to see what differences there might be.
* Quantized Complex:
  + Plot of the complex results of the quantized FFT.

### Conclusion

## Implementation