MATHSTAR™

Fast Silicon, Faster Solutions

1024 Point FFT Implementation Using MathStar's SOA13D40-01 Field Programmable Object Array (FPOA)

Silicon Objects Application Note **040409**

This document describes one possible early implementation of an N-point Radix2 FFT using MathStar's SOA13D40-01 Filter Builder FPOA. FPOAs consist of an array of 16-bit processing elements called Silicon Objects. Please reference MathStar's FPOA data sheet for a complete description of FPOA architecture and functions.

This mapping is scalable from 64-point to 1024-point via parameters. The current best performance for an N-pt FFT is:

(log 2 N) * N/(2*BF) * 6 cycles

where BF is the number of Radix 2 Butterflies used. For the SOA13D40-01 FPOA, 32 Radix 2 Butterflies are available. This configuration requires a total of 960 cycles. Operating at 1 GHz, this translates to 960ns. Further performance improvement is expected.

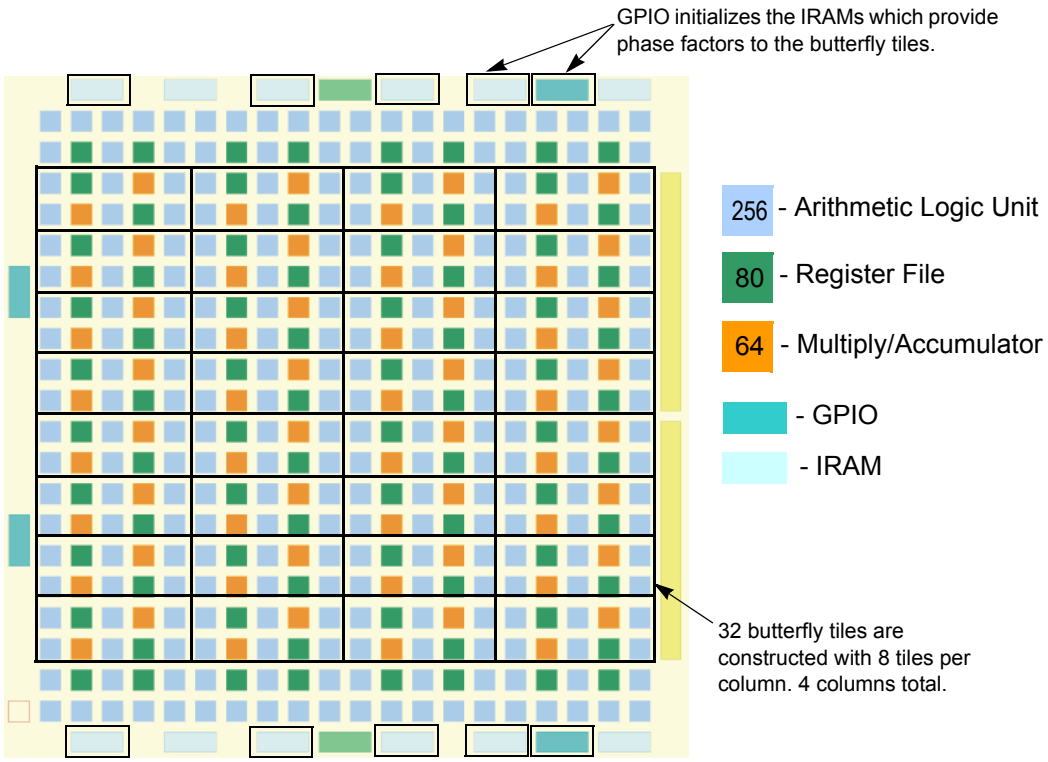


Figure 1. SOA13D40-01 Field Programmable Object Array (FPOA)

Figure 1 shows the 20x20 FPOA object array, including the periphery or I/O objects, used for this application mapping. In the object array, the three Silicon Object types used are:

- Arithmetic Logic Unit (ALU)
- Register File (RF)
- Multiply/Accumulator (MAC)

IRAM and GPIO periphery objects are also used for this application.

This applications note will focus on the implementation of a 1024 point radix2 DIF (decimation in frequency) FFT using the MathStar SOA13D40-01 FPOA.

The physical size of this array, and the composition of the butterfly tile (2x5 array), gives us to a total of 32 butterflies. Implementation of a 1024 point FFT requires that each of the 32 butterflies be used 16 times per stage over ten stages to complete the entire FFT operation.

Phase (or twiddle) factors are pre-calculated and stored in the IRAM on the upper and lower edges of the device. The data in and out of the chip is handled through the chip's GPIO interfaces.

Radix 2 DIF FFT Implementation

Fast Fourier Transform (FFT) is derived from Discrete Fourier Transform (DFT). It transforms data samples between the time domain and the frequency domain. The underlying assumption here is that the number of points to be calculated will be a power of 2, such as 4, 16, 32, 64, 128, 256, etc. FFT is a typical "divide-and-conquer" application. Large numbers of data samples are divided into two groups; for our purposes odd and even indexed. Each group is further divided until there are only two data points to process. Refer to any basic FFT book for a detailed derivation of this strategy.

The method described, of halving the data points each time, is called a radix2. Each Radix2 DIF FFT must process two complex numbers using the proper twiddle factors. This processing is called a butterfly operation. The figure below illustrates the operations that comprise the butterfly building block used in a radix2 DIF FFT.

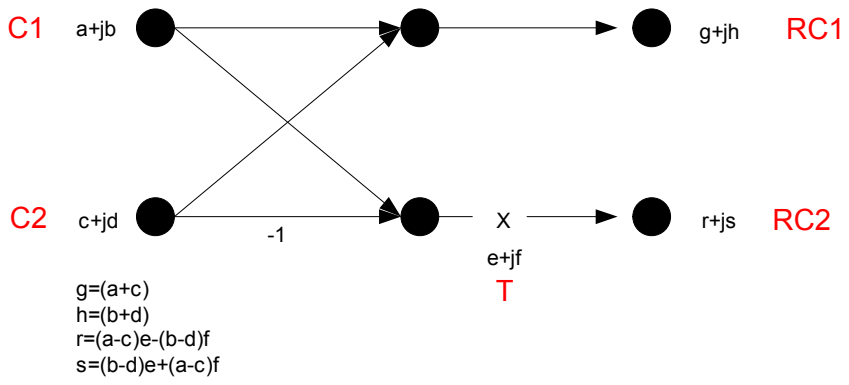


Figure 2. Radix2 Butterfly Computations

The computations performed by each butterfly tile are defined as:

- C1 = a + bj;

- C2 = c + dj;
- T = e + fj;
- RC1 = g + jh;
- RC2 = r + js;

Where RC1 and RC2 are the result complex numbers from butterfly operation.

- RC1 = C1 + C2;
- RC2 = (C1 - C2) * T;

The derivation of these equations can be found in any text covering FFTs.

Substituting the definition of C1, C2, and T into the expression of RC1 and RC2 we have:

- RC1 = (a+c) + (b+d)j;
- RC2 = ((a-c) + (b-d)j) * (e+fj)
  = ((a-c)e - (b-d)f) + ((a-c)f + (b-d)e)j;

This yields the desired outputs:

- G = a + c;
- H = b + d;
- R = ((a-c)e - (b-d)f);
- S = ((b-d)e + (a-c)f);

Where a, b, c, d, e, and f, are inputs; g, h, r, and s are the outputs.

The radix2 butterfly DIF execution sequence is illustrated here describing the per-clock cycle operation of the butterfly tile.

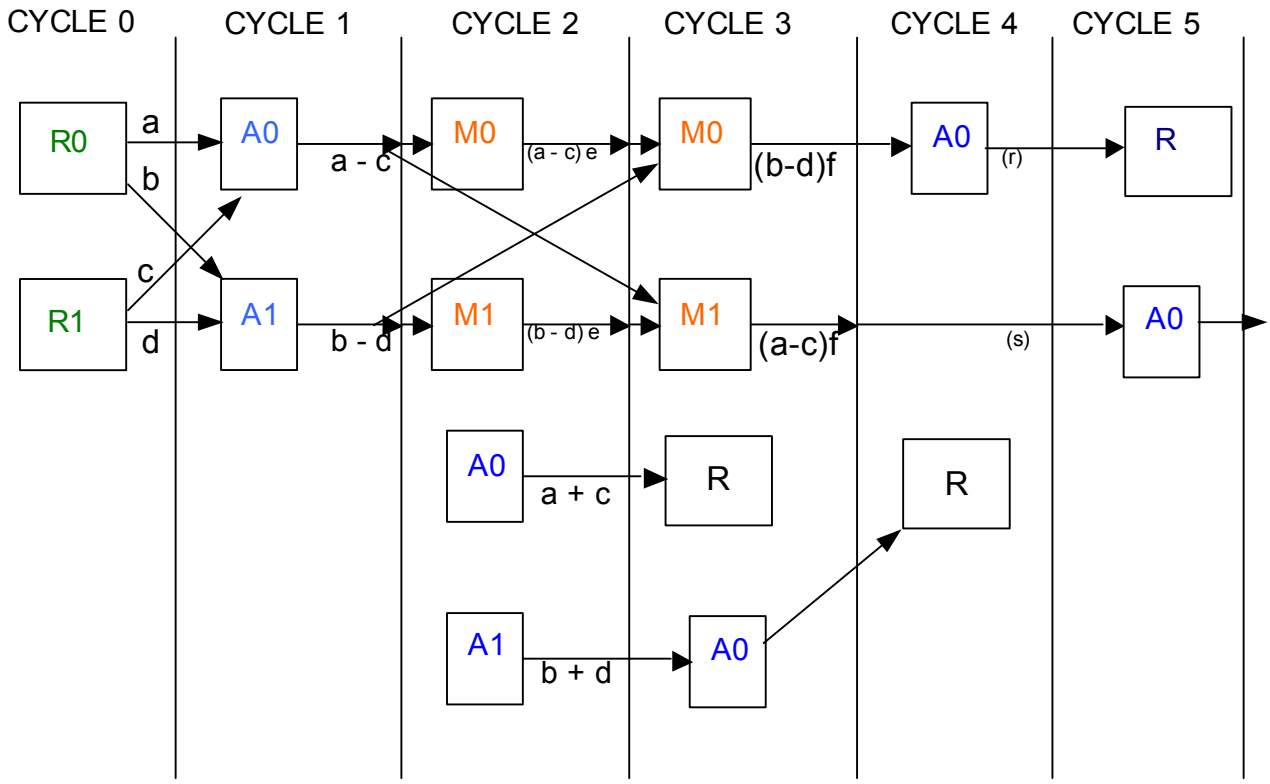


Figure 3. Butterfly Tile DIF Execution by Clock Cycle

Butterfly Operation and Tile Structure

Each butterfly tile consists of two Multiply/Accumulate (MAC), two Register File (RF), and six Arithmetic Logic Unit (ALU) silicon objects. They are arranged in the following configuration.

Figure 4. Butterfly Tile Configuration (1 of 32)

These ten silicon objects, in this configuration, comprise the butterfly tile. The SOA13D40-01 FPOA contains 32 of these tiles using a four column x eight tile matrix as

illustrated in Figure 1. The following is a brief description of the operation of each silicon object and the data flow within the butterfly tile as illustrated in Figure 5.

ALU0	Processes a-c and collects g, h, r, s, results and sends them out.
ALU1	Generates control signals for MAC0 and MAC1, and read enables for RF0 and RF1.
RF0	Provides a and b inputs to ALU0 and ALU1. It also receives inputs redirected from ALU3.
RF1	Provides c and d inputs to ALU0 and ALU1. It also receives inputs redirected from ALU4
MAC0	Processes (a-c)e and (b-d)f and then accumulates.
MAC1	Processes (a-c)f and (b-d)e and then accumulates.
ALU2	Generates the write address to RF0
ALU5	Generates the write address to RF1
ALU3	Muxes in GPIO data inputs and inputs from the source butterfly and redirects it to RF0.
ALU4	Muxes in GPIO data inputs and inputs from the source butterfly and redirects it to RF1.

To improve efficiency and conserve resources for inter-tile routing, communications within the butterfly should avoid using party line resources when possible. Using nearest neighbor communications will implement the butterfly operation in six clock cycles per stage, per butterfly. To complete the entire 1024 point operation will require ten stages per-butterfly.

The radix2 DIF data flow within the butterfly tile is shown here.

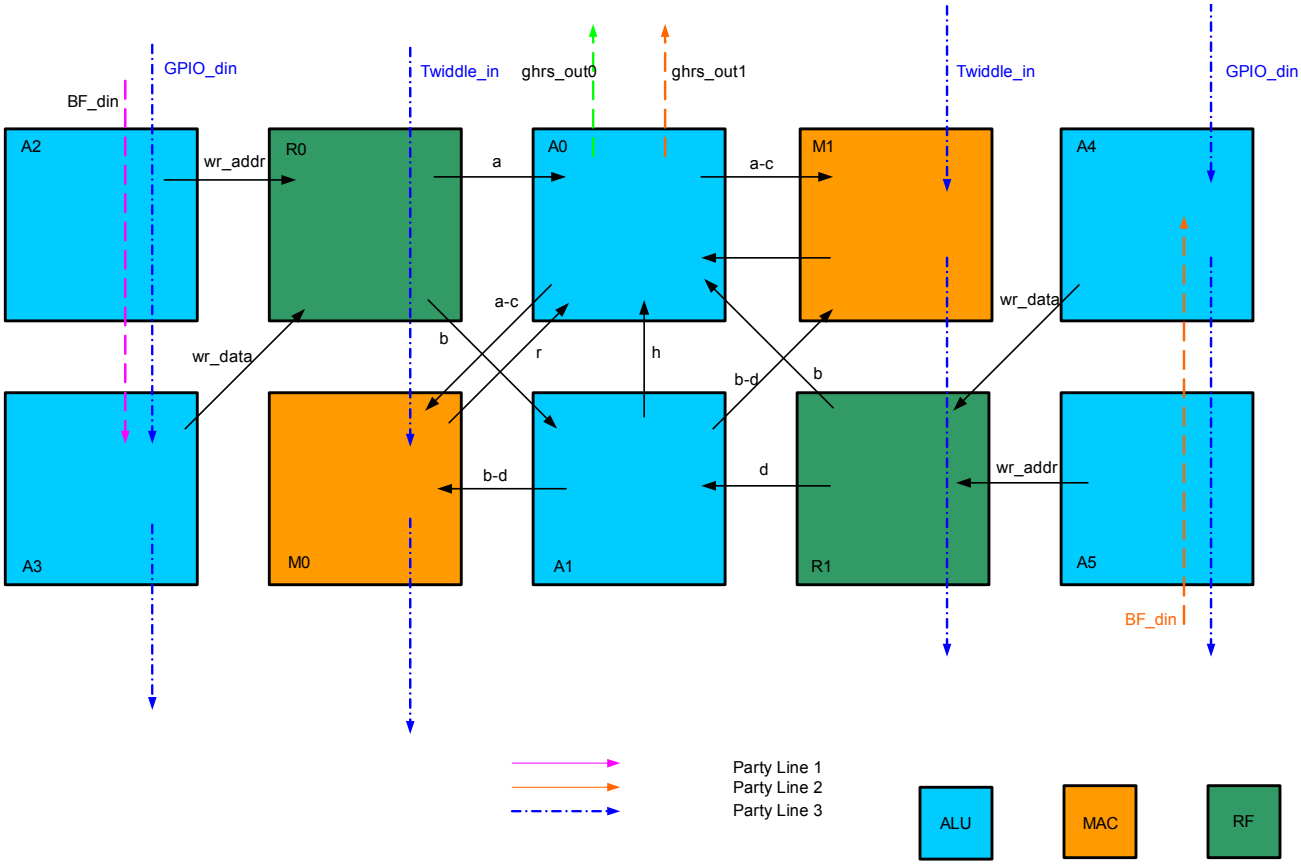


Figure 5. Radix2 DIF Data Flow within the Butterfly.

Inter-butterfly Data Flow and Routing

Data must be moved between butterflies after the execution of each butterfly stage. Since there are 32 butterflies and the number of data points to be processed is 1024, each butterfly will repeat its operation 16 times to complete the FFT. The butterfly distribution and numbering on the array is shown here.

BF0	BF1	BF2	BF3
BF16	BF17	BF18	BF19
BF4	BF5	BF6	BF7
BF20	BF21	BF22	BF23
BF8	BF9	BF10	BF11
BF24	BF25	BF26	BF27
BF12	BF13	BF14	BF15
BF28	BF29	BF30	BF31

The data pairs that BF0 will initially process are as follows:

BF0 <= [0/512, 1/513, 2/514, …15/527]

The same rule then applies to BF1 through BF31:

BFx <= [(0+x)/(512+x), (1+x)/(513+x), … (15+x)/(527+x)]

where x = 1, 2, ...31

Each butterfly must process 16 data pairs sequentially. It doesn't wait for outputs from other butterflies until it finishes all 16 pairs. This is referred to as one stage. By the time all 16 pairs have finished there should be 14 or 15 renewed data pairs available for further processing from the RF within the butterfly. After several data pairs are processed the remaining data pairs will have arrived, at which point the second stage will be finished. Continuing this for 10 stages will properly process all 1024 data points.

After six stages of data processing using this scheme, the data stored in each butterfly is exactly the same index as it was from the beginning. For example, BF0 would have:

BF0 <= [0/512, 1/513, 2/514, …15/527]

To handle this anomaly requires a slight modification to the data moving scheme. Each of the butterflies will need to process 16 pairs of data before going into next stage. There are 10 stages for each data pair and each stage will have to be repeated 16 times. An odd cycle is defined as the first eight butterfly cycles out of the sixteen at each stage. An even cycle is defined as the next eight butterfly cycles out of the sixteen at each stage.

The data transport pattern is described as following.

BFm, BFm+n/2 -> BF2m @ odd cycle, -> BF2m+1 @ even cycle

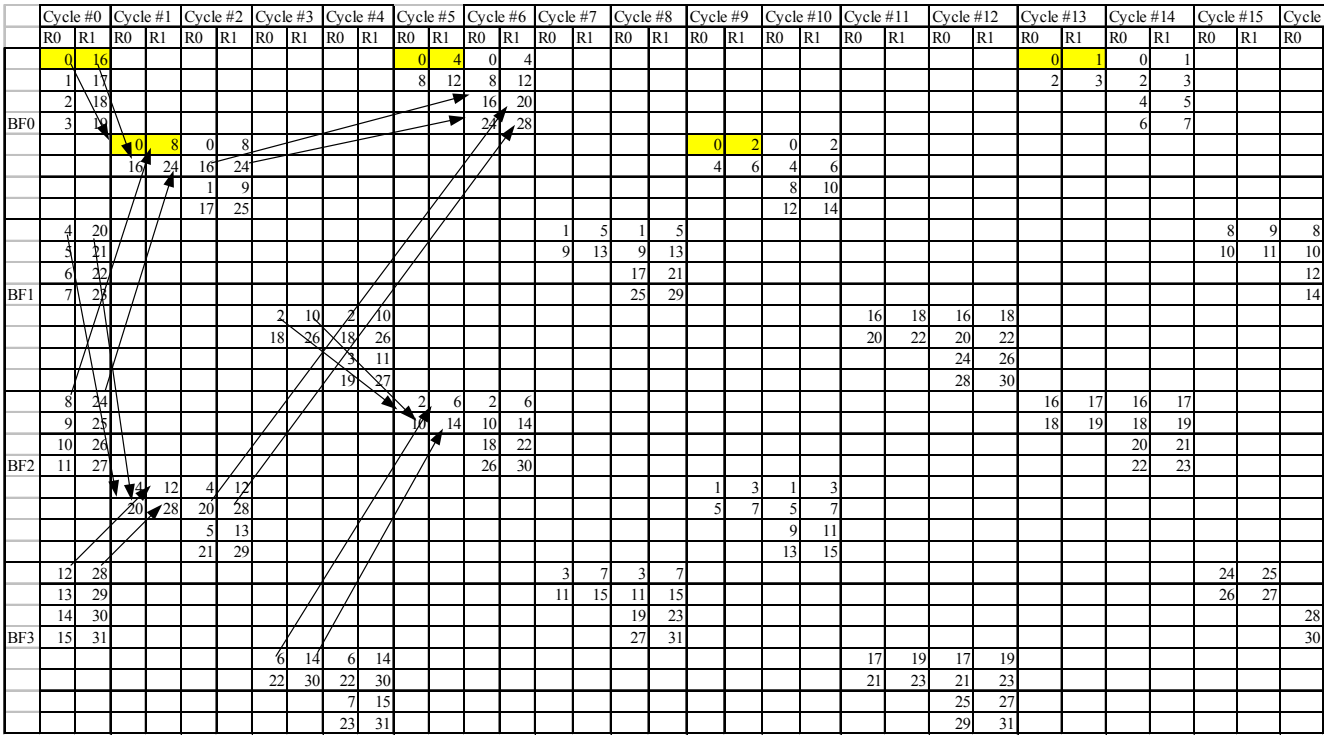
Where n is the number of tiles, m = 0, 1, … (n/2 - 1).

The output of BFm will go to BF2m storage RFs at odd cycles and BF2m+1 at even cycles. Thus the data moving pattern between tiles is as follows:

bf0,bf16	-> bf0 @ odd cycle,	-> bf1 @ even cycle
bf1,bf17	-> bf2 @ odd cycle,	-> bf3 @ even cycle
bf2,bf18	-> bf4 @ odd cycle,	-> bf5 @ even cycle
bf3,bf19	-> bf6 @ odd cycle,	-> bf7 @ even cycle
bf4,bf20	-> bf8 @ odd cycle,	-> bf9 @ even cycle
bf5,bf21	-> bf10 @ odd cycle,	-> bf11 @ even cycle
bf6,bf22	-> bf12 @ odd cycle,	-> bf13 @ even cycle
bf7,bf23	-> bf14 @ odd cycle,	-> bf15 @ even cycle
bf8,bf24	-> bf16 @ odd cycle,	-> bf17 @ even cycle
bf9,bf25	-> bf18 @ odd cycle,	-> bf19 @ even cycle
bf10,bf26	-> bf20 @ odd cycle,	-> bf21 @ even cycle
bf11,bf27	-> bf22 @ odd cycle,	-> bf23 @ even cycle
bf12,bf28	-> bf24 @ odd cycle,	-> bf25 @ even cycle
bf13,bf29	-> bf26 @ odd cycle,	-> bf27 @ even cycle
bf14,bf30	-> bf28 @ odd cycle,	-> bf29 @ even cycle
bf15,bf31	-> bf30 @ odd cycle,	-> bf31 @ even cycle

Odd cycles are butterfly cycles 0~7, 16~23, 32~39, …
Even cycles are butterfly cycles 8~15, 24~31, 40~47, …

The following graph shows how the data moving is performed for the 1024 point FFT using the first four butterflies (BF0 - BF3).



	Cycle #0		Cycle #1		Cycle #2		Cycle #3		Cycle #4		Cycle #5		Cycle #6		Cycle #7		Cycle #8		Cycle #9		Cycle #10		Cycle #11		Cycle #12		Cycle #13		Cycle #14		Cycle #15		Cycle
	R0	R1	R0	R1	R0	R1	R0	R1	R0	R1	R0	R1	R0	R1	R0	R1	R0	R1	R0	R1	R0	R1	R0	R1	R0	R1	R0	R1	R0	R1	R0
	0	16									0	4	0	4													0	1	0	1			
	1	17									8	12	8	12													2	3	2	3			
	2	18											16	20															4	5			
BF0	3	19											24	28															6	7			
			0	8	0	8													0	2	0	2											
			16	24	16	24													4	6	4	6											
					1	9															8	10											
					17	25															12	14											
	4	20													1	5	1	5													8	9	8
	5	21													9	13	9	13													10	11	10
	6	22															17	21															12
BF1	7	23															25	29															14
							2	10	2	10													16	18	16	18							
							18	26	18	26													20	22	20	22							
									3	11															24	26							
									19	27															28	30							
	8	24									2	6	2	6													16	17	16	17			
	9	25									10	14	10	14													18	19	18	19			
	10	26											18	22															20	21			
BF2	11	27											26	30															22	23			
			4	12	4	12													1	3	1	3											
			20	28	20	28													5	7	5	7											
					5	13															9	11											
					21	29															13	15											
	12	28													3	7	3	7													24	25	
	13	29													11	15	11	15													26	27	
	14	30															19	23															28
BF3	15	31															27	31															30
							6	14	6	14													17	19	17	19							
							22	30	22	30													21	23	21	23							
									7	15															25	27							
									23	31															29	31							

Figure 6. Sample Data Moving Pattern 1024 Point FFT w/Four Butterflies

Each butterfly will have only two output words, ghrs_out_odd and ghrs_out_even. One ALU inside the butterfly will collect all four output words and send them out sequentially to the designated butterfly.

Twiddle Factors

All the twiddle factors are pre-calculated and stored in the IRAM on the FPOA device. They are read out in the order needed. There are eight IRAMS used for the 1024 FFT application. Four of them are at the top of the array and the other four IRAMs are at the bottom of the array.

Each IRAM is 768x76 bits.

IRAM Initialization

GPIO north is responsible for the initialization of the four IRAMs on the north side of the FPOA. Inputs will consist of 16 data bits and 5 control bits. The leftmost ALU beneath the GPIO is responsible for the control of the GPIO. Data coming through GPIO is time-multiplexed into the corresponding IRAM depending on the control bit coming with data.

IRAM control is provided by the two ALUs directly beneath the IRAM. The leftmost ALU in the tile is also responsible for generating IRAM addresses in both read and write modes.

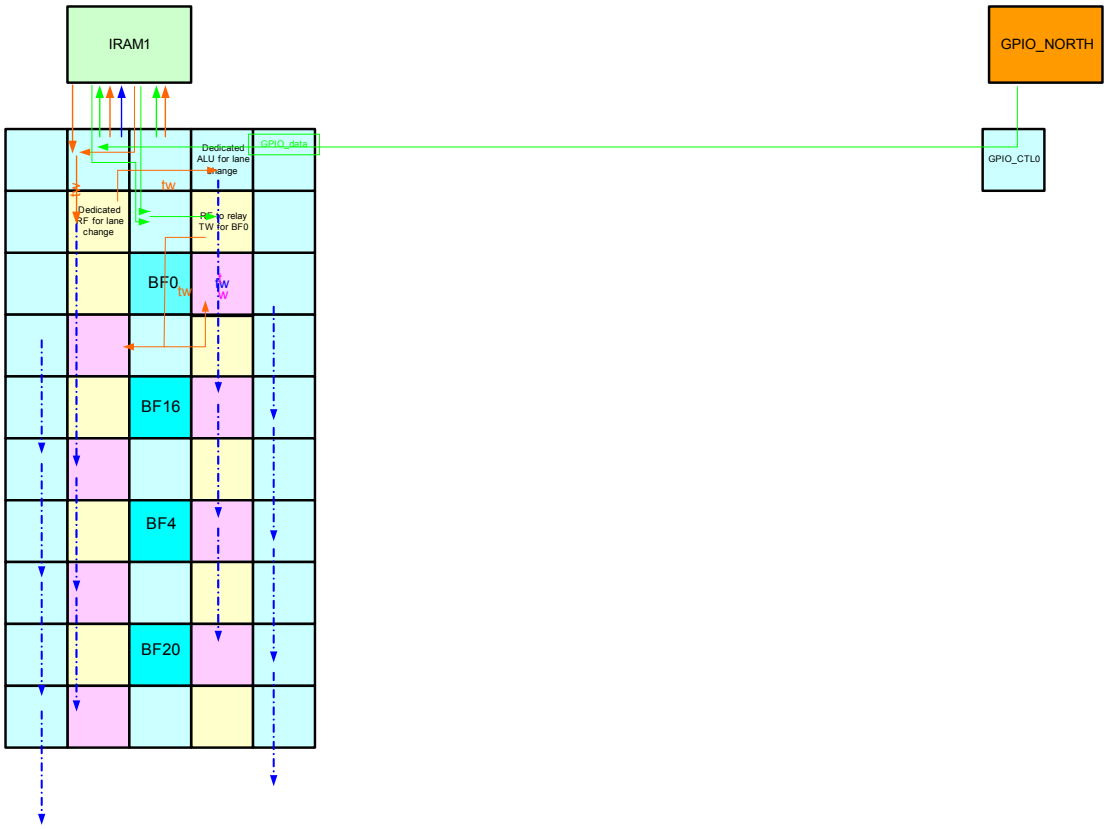


Figure 7. Supplying Twiddle Factors for Column 0, Rows 0-3 (BF0 - BF3)

Providing Twiddle Factors to the MACs

Providing twiddle factors to each MAC in each butterfly requires some additional work.The FFT core is divided into four columns, with each column containing eight rows of butterfly tiles.

Each butterfly needs different twiddle factors at each stage, for each data pair input. Every six clock cycles there will be two words presented to the two MACs in each butterfly tile.

For discussion purposes our focus will be on the first tile column, top four butterfly tiles (BF0-BF3).

IRAM1 provides twiddle factors to BF16, BF4, BF20 using Party Line 3 of the column where MAC0 and MAC1 are. Because party Line 3 cannot turn either east or west some buffers are necessary to allow a lane change. One RF and one ALU are dedicated for this purpose.

The same IRAM also provides twiddle factors to BF0 through the dedicated RF. That RF is necessary to balance timing compared to BF16, BF4, BF20.

Data Input and Output

Data input is provided through the GPIO at the top of the array (GPIO North). Data output is provided through the GPIO South. Note GPIO South must also input data for the four IRAMs at the south side of the FPOA during IRAM initialization.

During the FFT operation all the RFs will start to fill in new FFT inputs via GPIO north one butterfly, (actually one RF) half a butterfly at a time, each column sequentially. Starting from BF0, BF16, BF4, BF20, BF8, BF24, BF12, BF28, BF1, BF17, etc.

At the same time RF gets in new data, the center ALU begins to output results calculated during previous the FFT operation. Each column is chained together through Party Line muxing in the center ALU column. Each of the four columns is muxed out to GPIO South.