SIEMENS

Microcontrollers

ApNote AP1634

☑ additional file
AP163402.EXE available

Fast - Fourier - Transformation

The Fast Fourier Transformation (FFT) is an algorithm frequently used in various applications, like telecommunication, signal and image processing. It transforms the time domain into the frequency domain where the spectrum of amplitude and frequency can be analyzed.

K. Westerholz / Siemens HL MCB AT

AP1634 ApNote - Revision History		
Actual Revision : Rel.01 Previous Revison: none		
Page of actual Rel.	Page of prev. Rel.	Subjects changes since last release)

1 Abstract

The Fast Fourier Transformation (FFT) is an algorithm frequently used in various applications, like telecommunication, signal and image processing. It transforms the time domain into the frequency domain where the spectrum of amplitude and frequency can be analyzed.

This application note describes an implementation of a real-valued 1024 point decimation in time radix-2 FFT for the C166 microcontroller family. Assuming that the code is started out of the internal ROM via a 16-bit demultiplexed bus, an execution time of 10 ms has been achieved for a C165 running at 25 MHz internal clock. The code comprises 828 bytes.

This application note is based on an application note performed by **pls** (Programmierbare Logik Systeme, Hoyerswerda, Germany).

2 FFT - Derivation of the algorithm

Starting from the continuous time Fourier Transformation, the discrete Fourier Transformation can be derived as a function of sample points (N):

$$X(k) = \sum_{n=0}^{N-1} W_N^{nk} x(n) \qquad \text{k=0,1,...,N-1} \qquad W_N^{nk} = e^{-j2\pi/N \cdot nk} = \cos\left(\frac{2\pi}{N} \cdot nk\right) - j\sin\left(\frac{2\pi}{N} \cdot nk\right)$$

This formula can be regarded as a matrix W_N^{nk} multiplied by the input data vector x(n). This simple DFT has the complexity N^2. The coefficients of the matrix W_N^{nk} will be denoted in the following as twiddle factors.

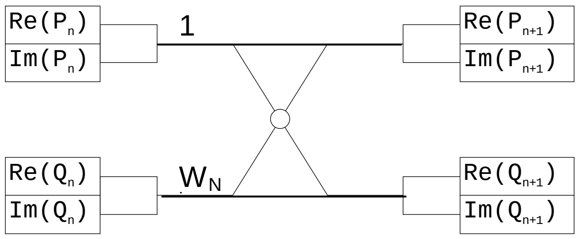
In order to derive the radix-2 FFT algorithm, we decompose the transformation into two partial transformations, one containing the input data with even indices, and the other with odd. Exploiting symmetries $W_N^k = -W_N^{k+(N/2)}$ and $W_N^k = W_N^{N+k}$ of the twiddle factors a decomposition is achieved that only requires $(\frac{1}{2}N^2 + \frac{1}{2}N)$ multiplications.

$$\begin{aligned} X(k) &= \sum_{n=0}^{N/2-1} x(2n) W_{N/2}^{nk} + W_N^k \sum_{n=0}^{N/2-1} x(2n+1) W_{N/2}^{nk} \\ &= P(k) + W_N^k Q(k) \end{aligned}$$

$$\begin{aligned} X(k+N/2) &= \sum_{n=0}^{N/2-1} x(2n) W_{N/2}^{nk} - W_N^k \sum_{n=0}^{N/2-1} x(2n+1) W_{N/2}^{nk} \\ &= P(k) - W_N^k Q(k) \end{aligned} \qquad 0 \le k \le (N/2)-1$$

The number of multiplications can be further cut to $(N \log N)$ by repetitive employing this process to the partial transformations P(k) and Q(k) till they are completely decomposed into 2 point DFTs (Discrete Fourier Transformations). The 2-point DFT is commonly referred to as the butterfly operation. The butterfly requires two complex multiplications resulting in four real valued multiplications for computing the terms P(k) and W_N^k Q(k).

$P_{n+1} = P_n + Q_n \cdot W_N^k$ *n=stage*
$Q_{n+1} = P_n - Q_n \cdot W_N^k$
since $W_N^k = e^{-j(2\pi/N)k} = \cos(u) - j \sin(u)$
and $u = (2\pi/N)k$



Having only real valued input data x(n), the computational effort of a N point FFT can be reduced to a N/2 point complex FFT. Firstly, even indexed data $h(n) = x(2n)$ and odd indexed data $g(n) = x(2n+1)$ are separated. The index k is running from 0 to N-1. h(n) and g(n) have the spectra H(k) and G(k) respectively. The spectrum X(k) can be decomposed into the spectra H(k) and G(k) as follows:

$$x(n) \xrightarrow{\text{Fourier Transformation}} X(k) = H(k) + j(\cos \tfrac{2\pi k}{N} - j \sin \tfrac{2\pi k}{N})G(k)$$

In order to cut the above N point transformation into an N/2 point transformation a complex input vector y(n) = h(n) + jg(n) is formed with the index n running from 0 to N/2-1. The real input values are formed by the even indexed input data h(n). The imaginary part is formed by the odd indexed input data g(n). Then y(n) is transformed into the frequency domain resulting in a spectrum consisting of a superposition of the spectra H(k) and G(k).

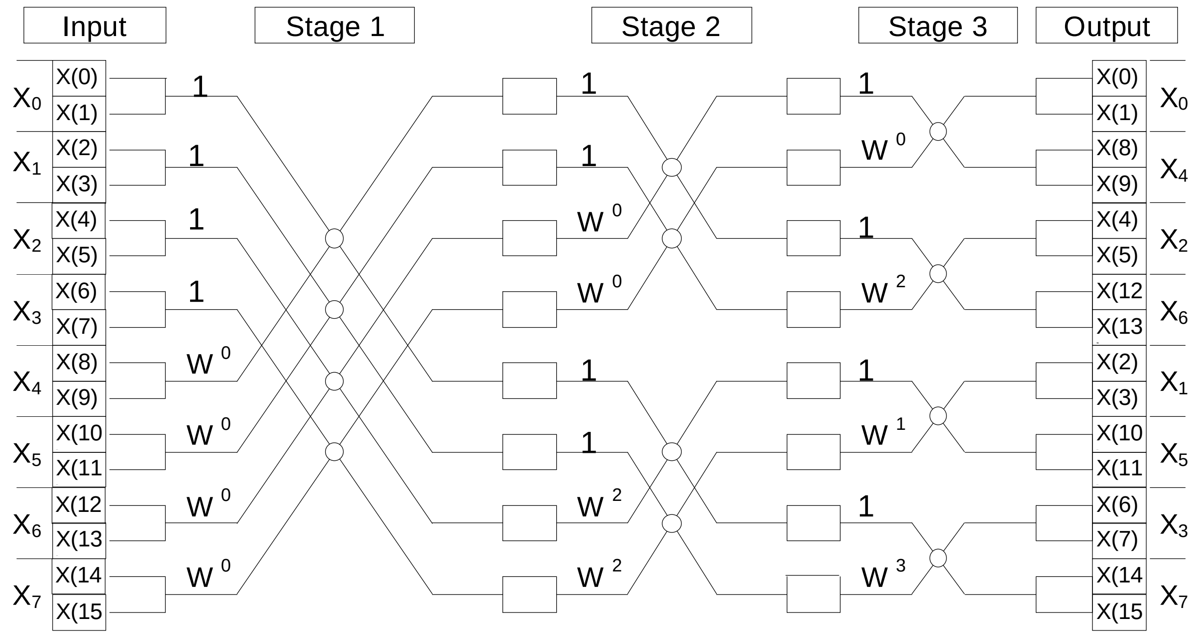
$$y(n) = h(n) + jg(n) \xrightarrow{\text{Fourier Transformation}} Y(k) = H(k) + jG(k) = R(k) + jI(k)$$

Now the complex spectra H(k) and G(k) have to be extracted out of the complex spectrum $Y(k) = H(k) + jG(k) = R(k) + jI(k)$. By employing symmetry relations, the spectra H(k) and G(k) can be derived from the spectrum Y(n) as follows:

$$\mathrm{Re}\{H(k)\} = \frac{R(k) + R(N/2-k)}{2} \qquad \mathrm{Im}\{H(k)\} = \frac{I(k) - I(N/2-k)}{2}$$
$$\mathrm{Re}\{G(k)\} = \frac{I(k) + I(N/2-k)}{2} \qquad \mathrm{Im}\{G(k)\} = \frac{R(k) - R(N/2-k)}{2}$$

These spectra inserted into the equation for X(n) deliver the full spectrum.

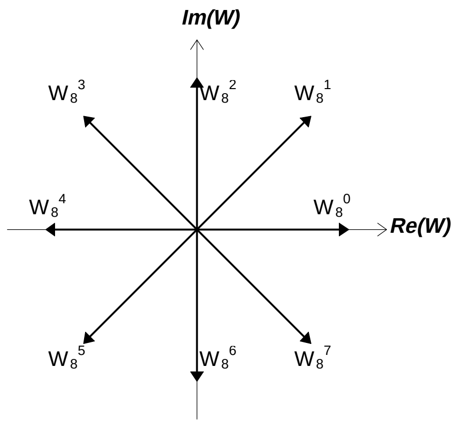
In order to illuminate the FFT algorithm, the example given below shall demonstrate the computational process during an 8-point FFT. The input data consists of 8 complex numbers X_0, ... X_7 which can also be perceived as 16 real numbers x(0)...x(15).



Corresponding to the above butterfly operation, the input data are connected with each other. Due to the nature of this operation the input data is sequentially ordered, while the output data is in bitreversed order. In each stage four (=N/2) butterflies are computed. The twiddle factor W in the 1st stage is always $W_8^0 = 1$. In the 2nd stage the upper half consists again of the butterflies from the previous stage. The second half of the butterflies are computed with twiddle factor W_8^2. In the third stage the twiddle factors of the previous stage appear again in the upper half and the lower half contains the factors W_8^1 and W_8^3.

For an 8-point-FFT the twiddle factors $W_N^k = e^{-j(2\pi/N)k} = \cos(X) - j\sin(X)$ have the following values.

Twiddle factor	cos(X)	sin(X)
W_8^0	1	0
$W_8^1 = e^{-j(\pi/4)}$	½√2	½√2
$W_8^2 = e^{-j(\pi/2)}$	0	1
$W_8^3 = e^{-j(3\pi/4)}$	-½√2	½√2
$W_8^4 = e^{-j\pi} = -W_8^0$	-1	0
$W_8^5 = e^{-j(5\pi/4)} = -W_8^1$	½√2	½√2
$W_8^6 = e^{-j(3\pi/2)} = -W_8^2$	0	1
$W_8^7 = e^{-j(7\pi/4)} = W_8^3$	-½√2	½√2



Because of the symmetry of the twiddle factor only the first four (W_8^0 ,..., W_8^3) must be computed.

These four twiddle factors lead to degenerated butterflies. They can be used to precompute the butterflies in the first three stages of the FFT to reduce the number of multiplications. For the twiddle factors W_N^0 , W_N^1 , W_N^2 and W_N^3 the butterfly computation can be simplified.

For $\mathbf{W_N^k = W_N^0}$ = 1 we have cos(X) = 1 ; sin(X) = 0. This results in a butterfly without any multiplications:

P_{n+1} = $[Re(P_n) + Re(Q_n) + j\,[Im(P_n) + Im(Q_n)]$
Q_{n+1} = $[Re(P_n) - Re(Q_n)] + j\,[Im(P_n) - Im(Q_n)]$

For $\mathbf{W_N^k = W_N^1}$ = ½√2(1 - j) we get cos(X) = sin(X) = ½√2. This results in a butterfly for the third stage where only two multiplications must be executed:

P_{n+1} = [Re(P_n) + Re(Q_n) · cos(X) + Im(Q_n) · cos(X)] + j [Im(P_n) + Im(Q_n) · cos(X) - Re(Q_n) · cos(X)]
Q_{n+1} = [Re(P_n) - Re(Q_n) · cos(X) - Im(Q_n) · cos(X)] + j [Im(P_n) - Im(Q_n) · cos(X) + Re(Q_n) · cos(X)]

For $\mathbf{W_N^k = W_N^2}$ = -j we have cos(X) = 0 ; sin(X) = 1. This results in a butterfly without any multiplications used in the second and third stage:

P_{n+1} = [Re(P_n) + Im(Q_n)] + j [Im(P_n) - Re(Q_n)]
Q_{n+1} = [Re(P_n) - Im(Q_n)] + j [Im(P_n) + Re(Q_n)]

For $\mathbf{W_N^k = W_N^3}$ = ½√2(-1 - j) we get -cos(X) = sin(X) = ½√2. This results in a butterfly for the third stage where only two multiplications must be executed:

P_{n+1} = [Re(P_n) + Re(Q_n) · cos(X) - Im(Q_n) · cos(X)] + j [Im(P_n) + Im(Q_n) · cos(X) + Re(Q_n) · cos(X)]
Q_{n+1} = [Re(P_n) - Re(Q_n) · cos(X) + Im(Q_n) · cos(X)] + j [Im(P_n) - Im(Q_n) · cos(X) - Re(Q_n) · cos(X)]

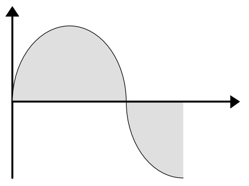
The output of the decimation in time FFT shows a bitreversed order that has to be ordered to calculate the final frequency spectrum. Supposing the input data has been in a sequential order, the indices of the output data can be easily computed by bit reversing the binary presentation of the input indices. The table below gives an example of the bit reversal for an 8 point FFT.

Order of Input data		Order of output data bitreversed	
index	binary	binary	index
0	000	000	0
1	001	100	4
2	010	010	2
3	011	110	6
4	100	001	1
5	101	101	5
6	110	011	3
7	111	111	7

3 Implementation

For the implementation we assume real valued input data. Based on this assumption, a real valued 1024 point FFT can be reduced to a 512 point complex FFT followed by an unweave phase in order to compute the 1024 point spectrum. The 512 point complex FFT consists of $\log_2$ (512) = 9 stages and each stage calculates 512/2 = 256 butterflies. The twiddle factors in the first three stages are the same as for the 8 point complex FFT.

The input data is stored in the table FFT_IN which consists of 1024 16 bit words. Since we perform an in-place FFT, the input data field will be overwritten by the output data. However, the final output will be stored in the separate FFTOUT output field. For providing the trigonomic functions, a table (FFT_DAT) is stored encompassing the precomputed sinus and cosines values. Since cos(x) = sin(x+π/4) the table consists of ¼ sinus period and ½ cosines period. Together they form a ¾ sinus period.



To rearrange the bitreversed output of the complex FFT and to calculate the twiddle factor W^k, a bitreversal table (FFT_BR) has also been precomputed.

The input data and the trigonometric function table are represented by a 15 bit fixed-point fraction in two's complement. This means that the MSB of such a number represents the sign, followed by the 15 bits representing a fraction of one.

$$\begin{array}{|c|c|c|c|c|c|c|c|}\hline s & , & b_1 & b_2 & b_3 & b_4 & ... & b_{15} \\ \hline \uparrow & & \uparrow & \uparrow & & & & \uparrow \\ \text{sig} & & 2^{-1} & 2^{-2} & & & ... & 2^{-15} \end{array}$$

examples:

binary	hex	dez	value
0111 1111 1111 1111	7FFF	32767	+1
0110 0000 0000 0000	6000	24576	+ 0.75
1010 0000 0000 0000	A000	-24576	- 0.75
1000 0000 0000 0000	8000	-32768	- 1

Since the addition of two numbers having the same sign might cause an overflow, numbers have to be divided by 2 before adding them. This is done by an arithmetic shift right. When multiplying two 15-bit-numbers, note that the signs of both are multiplied too, and the result is stored in the 32-bit wide multiplication register.

$$\boxed{s \,|\, \leftarrow 15 \text{ bit} \rightarrow} \times \boxed{s \,|\, \leftarrow 15 \text{ bit} \rightarrow} = \boxed{s \,|\, s \,|\, \leftarrow 30 \text{ bit} \rightarrow}$$

Therefore the result is equal to a scaled multiplication that means that it consists of the multiplication and a subsequent arithmetic shift right as a side effect. Because 32 bit precision is not required, only the first 16 bits contained in the MDH register are used for further calculations.

Attached you will find a program flow chart. The first part of the program consists of a 512-point complex radix-2 FFT which is executed in nine stages. To avoid multiplications by means of the degenerated butterflies, 5 different Mid- and Inloops are implemented. The idea is to cut the amount of calculation needed by using the degenerated butterflies in all stages. This is very effective in the first three stages, but time savings decreases in the rear stages.

Regarding the 512 complex point FFT, the number of twiddle factors amounts 512. However, due to the symmetry only the first 256 (0..255) are used. In the program source the twiddle factors denoted as W0, W4, W8, W12 refer to the angles 0, 90°, 45° and 135°. Thus, W_8^0 corresponds to W0, W_8^2 to W4, W_8^1 to W8, and W_8^3 to W12.

Stage 1: Since in stage one all twiddle factors are W0, all 256 butterflies are performed in Inloop_0.

Stage 2:	128 butterflies with W0 in Inloop_0
	128 butterflies with W4 in Inloop_1
Stage 3:	64 butterflies with W0 in Inloop_0
	64 butterflies with W4 in Inloop_1
	64 butterflies with W8 in Inloop_2
	64 butterflies with W12 in Inloop_3
Stage 4:	32 butterflies with W0 in Inloop_0
	32 butterflies with W4 in Inloop_1
	32 butterflies with W8 in Inloop_2
	32 butterflies with W12 in Inloop_3
	128 butterflies with Wk in Inloop

The second part of the program unweaves the bitreversed output of the 512-point FFT to extract the 1024 point real valued FFT. Optional, the final stage of the algorithm calculates an amplitude spectrum. On the last page you will find the register use during program execution.

Assuming that the code is started out of the internal ROM, the input data is stored in the external RAM and accessed via a 16-bit demultiplexed bus without wait states (Syscon: ROMEN=1, Buscon: MTTC=1, MCTC = 1111, BTYP = 10) an execution time of 10 ms for a real valued 1024-point FFT is achieved for the C165 running at 25 Mhz internally. The code size amounts 828 bytes without data tables. The optional computation of the amplitude spectrum consumes additional 2 ms. The execution time is independent from the input data. Changing the number of sample points N_, the number of stages exp, and reducing the tables, the algorithm can be tailored for various resolutions. In the table below you will find execution times for different numbers of sample points.

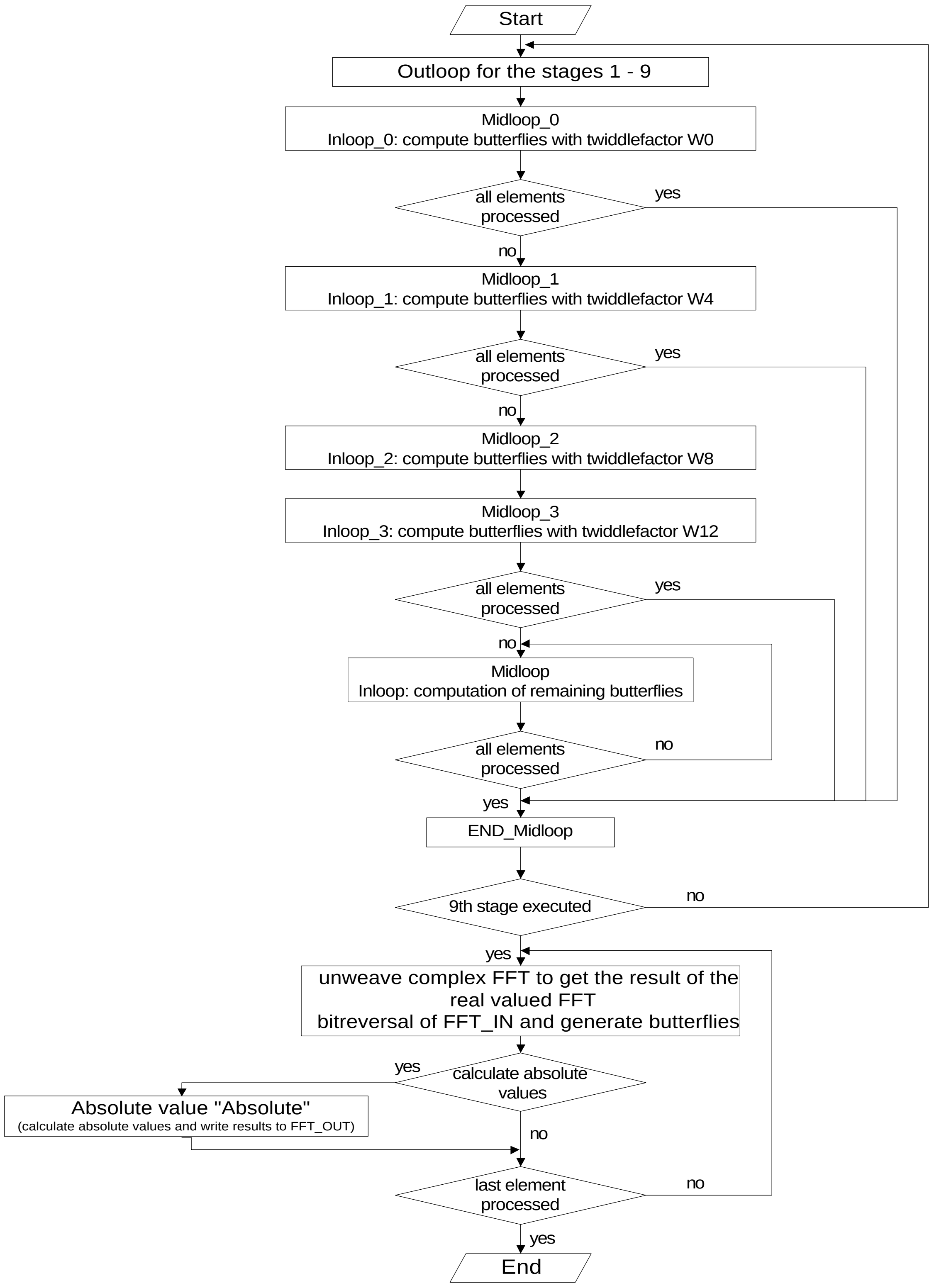
	64-Points	256- Points	1024- Points
SAB-C165 (25 MHz)	0,56 ms	2,6 ms	10,4 ms
SAB-C167CR (20 MHz)	0,7 ms	3,3 ms	13 ms

This figures demonstrate that the C166 architecture is even superior to some signal processors. This result is founded on a multiplication execution time of 400ns and the RISC like register file. Using a more complex radix-4 FFT algorithm combining two butterflies a further run time saving can be expected. Keeping the input and output tables in the internal RAM an additional speed up can be achieved for the 64 and 256 point FFT. If only parts of the spectrum have to be analyzed, a partial FFT can be performed [7] by omitting all calculations not contributing to the frequency window.

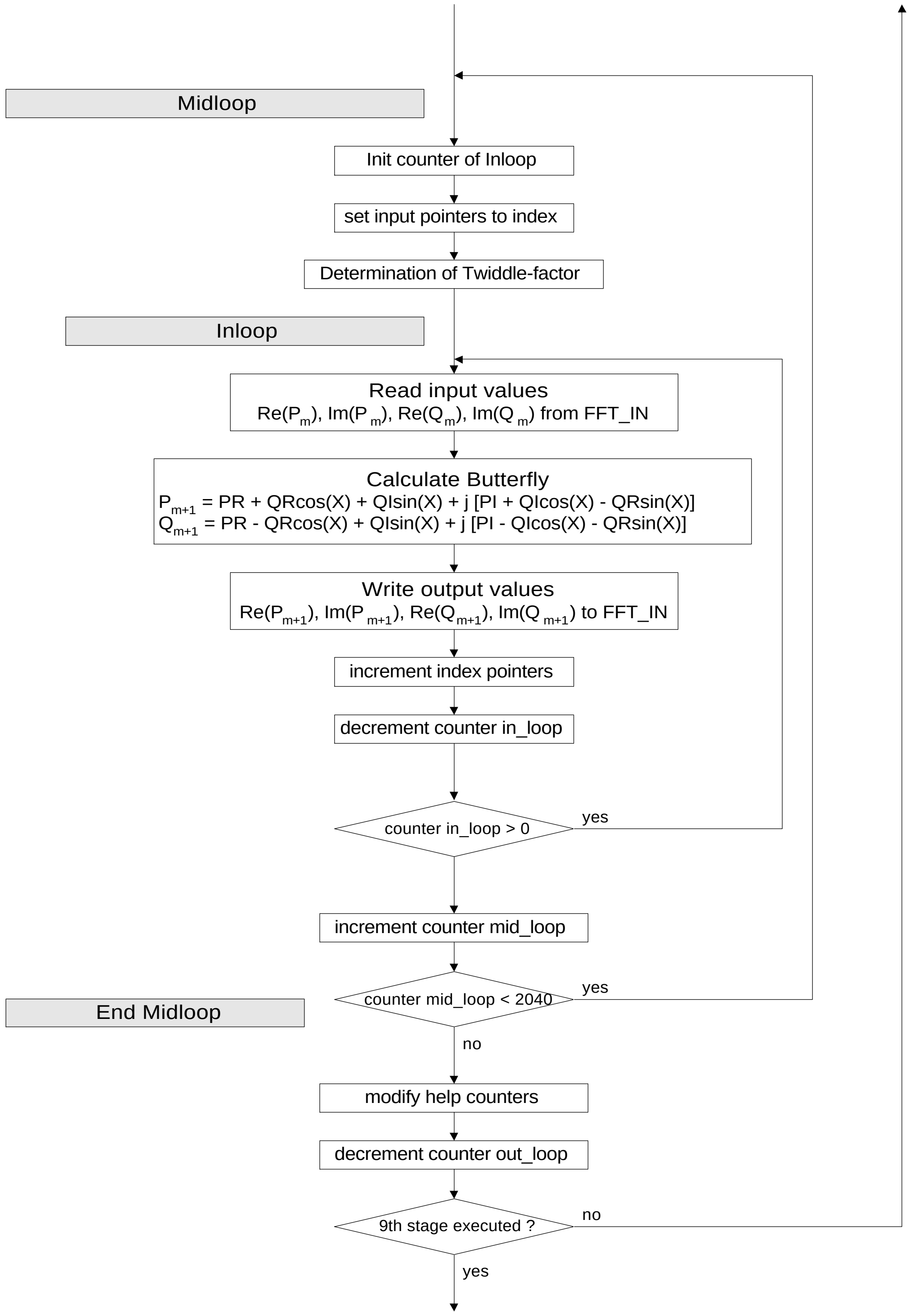
Literature

[1] Application Note AP-275 "An Algorithm for MCS-96 Products including Supporting Routines and Examples", intel September 1986
[2] MS-C-Program
[3] TMS32010 High Performance 16/32-Bit Digital Signal Processor Product Description Texas Instruments 1985
[4] Digital Signal Processing Applications with the TMS320 Family Texas Instruments 1988
[5] TMS320C25 Digital Signal Processor Product Description Texas Instruments 1986
[6] S.K. Mitra and J.F. Kaiser, Handbook for Digital Signal Processing, John Wiley & Sons, 1993
[7] Maurice Bellanger, Digital Processing of Signals, John Wiley & Sons, 1989

Flow chart of 1024-point real valued FFT


Start
Outloop for the stages 1 - 9
Midloop_0
Inloop_0: compute butterflies with twiddlefactor W0
all elements processed
yes
no
Midloop_1
Inloop_1: compute butterflies with twiddlefactor W4
all elements processed
yes
no
Midloop_2
Inloop_2: compute butterflies with twiddlefactor W8
Midloop_3
Inloop_3: compute butterflies with twiddlefactor W12
all elements processed
yes
no
Midloop
Inloop: computation of remaining butterflies
all elements processed
no
yes
END_Midloop
9th stage executed
no
yes
unweave complex FFT to get the result of the real valued FFT
bitreversal of FFT_IN and generate butterflies
calculate absolute values
yes
Absolute value "Absolute"
(calculate absolute values and write results to FFT_OUT)
no
last element processed
no
yes
End

Flowchart of the Midloop


Midloop
Init counter of Inloop
set input pointers to index
Determination of Twiddle-factor
Inloop
Read input values
$Re(P_m), Im(P_m), Re(Q_m), Im(Q_m)$ from FFT_IN
Calculate Butterfly
$P_{m+1} = PR + QRcos(X) + QIsin(X) + j [PI + QIcos(X) - QRsin(X)]$
$Q_{m+1} = PR - QRcos(X) + QIsin(X) + j [PI - QIcos(X) - QRsin(X)]$
Write output values
$Re(P_{m+1}), Im(P_{m+1}), Re(Q_{m+1}), Im(Q_{m+1})$ to FFT_IN
increment index pointers
decrement counter in_loop
counter in_loop > 0
yes
increment counter mid_loop
counter mid_loop < 2040
yes
End Midloop
no
modify help counters
decrement counter out_loop
9th stage executed ?
no
yes

Register Use and content

Stage 1 to 9:

	Stage1	Stage2	Stage3	Stage4	Stage5	Stage6	Stage7	Stage8	Stage9	Note
R0	9	8	7	6	5	4	3	2	1	Outloop
R1	1024	512	256	128	64	32	16	8	4	counter
R2	2048	1024	512	256	128	64	32	16	8	
R3	0 , N*2-4	0...N*2-4	0...N*2-4	0...N*2-4	0...N*2-4	0...N*2-4	0...N*2-4	0...N*2-4	0...N*2-4	Midloop
R4			Twid/BF	Twid/BF	Twid/BF	Twid/BF	Twid/BF	Twid/BF	Twid/BF	
R5	pP	pP	pP	pP	pP	pP	pP	pP	pP	pP_FFT_IN
R6	pQ	pQ	pQ	pQ	pQ	pQ	pQ	pQ	pQ	pQ_FFT_IN
R7	BF	BF	cos[k]	cos[k]	cos[k]	cos[k]	cos[k]	cos[k]	cos[k]	
R8	BF	BF	BF	sin[k]	sin[k]	sin[k]	sin[k]	sin[k]	sin[k]	
R9	BF	BF	BF	BF	BF	BF	BF	BF	BF	
R10	BF	BF	BF	BF	BF	BF	BF	BF	BF	
R11	BF	BF	BF	BF	BF	BF	BF	BF	BF	
R12	BF	BF	BF	BF	BF	BF	BF	BF	BF	
R13	256...0	128...0	64...0	32...0	16...0	8...0	4...0	2...0	1, 0	Inloop
R14	256	128	64	32	16	8	4	2	1	
R15	p_FFT_In	p_FFT_In	p_FFT_In	p_FFT_In	p_FFT_In	p_FFT_In	p_FFT_In	p_FFT_In	p_FFT_In	p_FFT_In

N = number of samples (1024); BF = butterfly; Twid = twiddle factor; pP= pointer onto element P_n; pQ= pointer onto element Q_n

Stage 10 and procedure „Absolute“:

	Stage 10	Absolute
R0		Input (Low)
R1	Input a	Input (High)
R2		Output
R3	temp	temp
R4	sin[k]	temp
R5	Input b	temp
R6		temp
R7	temp	temp

	Stage10	Absolute
R8	cos[k]	
R9	Input c	
R10		
R11	Input d	
R12	0,2,4,6,8,...,1022	
R13		
R14		FFT_OUT
R15		