Journal of Scientific & Industrial Research
Vol. 80, March 2021, pp. 230-238

ASIC Design of Radix-2,8-Point FFT Processor

Prasad Kulkarni[1*], B G Hogade[1], Vidula Kulkarni[2] and Varsha Turkar[3]

[1]Terna Engineering College, Navi Mumbai 400 706, India

[2]Sanpada College of Commerce and Technology Sanpada, Navi Mumbai 400705, India

[3]Don Bosco College of Engineering, Goa 403 602, India

Received 15 August 2020; revised 08 February 2021; accepted 14 February 2021

In split radix architecture, large sizes Fast Fourier Transforms (FFT) are decomposed into small independent computations to reduce storage burden. Radix-2, 8-point is one the popular choice in split radix for small independent computation. Authors proposes the FFT processor architecture for this small independent computation i.e. radix-2, 8-point FFT. This paper brief architecture comprising Butterfly Unit (BU), register set and controller. The novelty of this architecture is that it replaces the series of Processing Elements (PE) by single BU. BU computes two halves of the computations concurrently. Arithmetic computations are performed in floating point form to overcome the nonlinearities. All computations are controlled by tailored instruction set. All instructions are of same size and have same execution time. Twiddle constants are implicitly available in the instruction. Internal computations are stored in register set to avoid the load and store operations with memory. The mean square error of the computation is reduced by 41.95% and 55.76% in magnitude and phase respectively as compared with computations performed by rounding the twiddle constant. This FFT processor is synthesized, placed and routed for 45 nm technology of nangate open cell library. The BU of this architecture is 18.89% smaller and 5.13% faster as compared with smallest and fastest BU reported previously. The hardware cost metric i.e. AT^2_{norm} D_p mm^2 ns^2 mW of proposed processor is 1.37. This cost metric is also 32.51% less as compared with the previous work.

Keywords: Butterfly Unit, Fast Fourier Transform, Fused Floating Point Addition–Subtraction, Non-redundant arithmetic

Introduction

Digital Signal Processor (DSP) widely use FFT for signal processing in variety of fields such as entertainment devices, wireless broadband communication system, microwave access (Wi Max), long term evolution, image processing and biomedical signal processing. In the past decade, various pipelined FFT processor architectures were presented on split radix in which large size FFTs were decomposed into small independent computations. Radix-2, 8-point FFT computation was majorly used as the one of decomposition in split-radix architectures. The decomposition of large size FFT helped to balance the functionality and increases the performance of FFT processor. The performance of the processor is also increased by eliminating memory to store the intermediate computations. The pipeline architectures were of mixed radix multipath delay feedback,[1,2] ring structured multiprocessor,[3] scalable array structure,[4] single delay feedback,[5] fixed point reconfigurable architecture[6] and parameterisable architecture for memory based FFT algorithm.[7] On the other hand, pipeline architectures consist of an interleaved series of computational elements and data storage elements i.e. processing elements (PE). Computational elements known as butterfly unit (BU) are responsible for performing multiplication and addition. Hence the architecture of BU is also an important unit to decide the performance of FFT processor. In this decade, various BUs were proposed based on floating point arithmetic to overcome nonlinearities such as overflow of number range, rounding errors, aliasing errors and coefficient errors. However, floating point arithmetic has sluggish nature. To improve speed and to reduce area of consumption, various arithmetic hardware were proposed by sharing common logic,[8] dual path pipeline,[9] multi-operand adder[10] and redundant arithmetic.[11,12] Lookup table enabled multiplier, hash indexing function[13] and Gauss-Eisenstein representation[14] was also used for arithmetic operations. This paper proposes architecture of radix-2, 8-point FFT processor for small independent

*Author for Correspondence
E-mail: prasad26276@gmail.com

computation suitable in split radix architectures. The novelty of this architecture is that single BU free from series of processing elements (PE), computes two halves of the computation concurrently. This BU also computes FFT in time domain as well as in frequency domain. Dual path fused floating point addition-subtraction (DFFAS) and two floating point multipliers (FMULT) are the major entities of BU.

The computation program based on radix-2 algorithm is written by author and stored in program memory. This paper briefs on the following:

1 Architecture of FFT processor.
2 BU, comprising DFFAS.
3 Tailored instruction set to perform arithmetic operations.
4 Comparison of FFT computational error occurred using floating point against the fixed-point representation of twiddle constant.

Architecture of Proposed FFT Processor

Architecture of proposed FFT processor is shown in Fig.1. BU, three register files, multiplexers and controller are the main entities in proposed FFT processor. Features of this FFT processor are

- It is16-bit processor.
- BU performing addition and multiplication on floating point numbers represented in 16 bits simple 2's complement form.
- Tailored instruction set. All instructions have equal length i.e., 20-bit and same execution time.
- It has three register files named as main, real and imaginary. Each register file consists of 8, 16-bit registers.

BU comprises of DFFAS, multiplexers and FMULT. This BU is responsible to perform arithmetic operations. Register files are used to hold the input sequence, intermediate computational operand and output sequence. 4:1 multiplexer is used to select the operands for arithmetic operations. 2:1 multiplexer enables data transfer between two registers.

The program memory is interfaced with the FFT processor using interfacing signals. These interfacing signals are shown in Table 1. The interfacing signals consist of 20-bit data bus, 6-bit address bus, clock input and reset input. Controller writes the address of program memory to fetch the instruction. The fetched instruction is decoded by controller. After decoding instruction, controller generates controls signals as shown in Table 2. The control signals WREN, WAD and RAD are used by register files to perform write and read operation. Register file has one input data bus and two output data buses. The input data bus is used to perform write operation. The register write operation is enabled by asserting WREN signal. The write operation is performed on the register whose address is available in WAD. Simultaneously, two

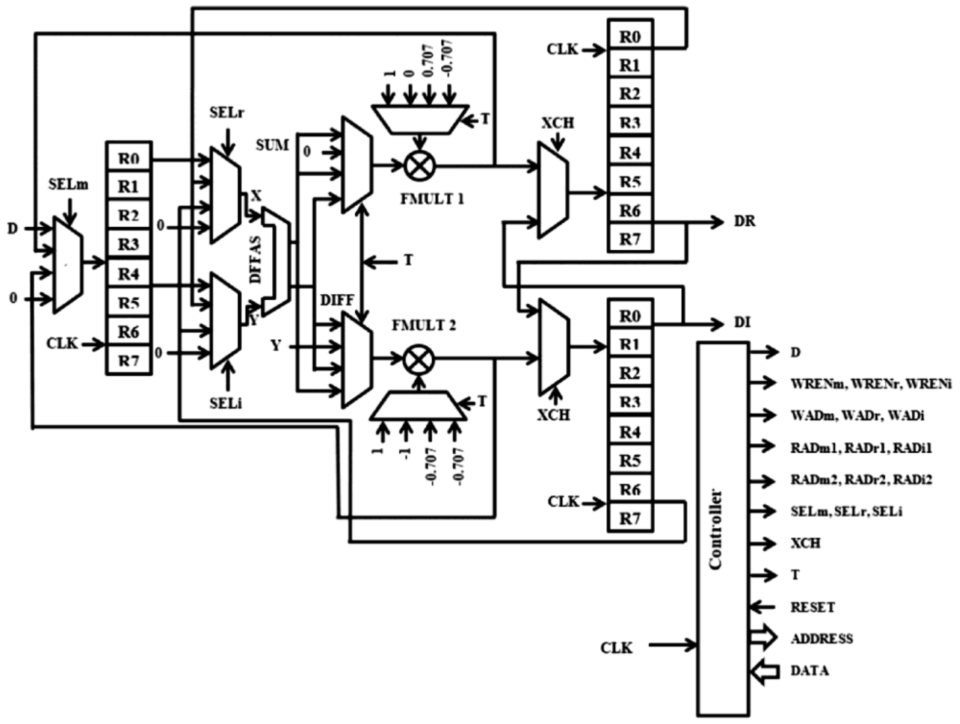

Fig.1 — Proposed FFT Architecture

registers are read through output data buses. Register file has two additional input buses i.e., RAD 1 and RAD 2 to perform read operation. RAD 1 and RAD 2 holds the addresses of two registers to perform read operation on them.

The signal XCH is used to copy the information from register available in real register file to register available in imaginary file and vice-versa. The register addresses are shown in Table 3. The Rm denotes the register from main file, Rr denotes the register from real file and Ri denotes the register from imaginary file.

The main register file stores the immediate data sequence (D), products from multipliers and 0d. They are selected through the select line SELm. Similarly, the operands X and Y for DFFAS are selected through SELr and SELi respectively. This operand selection is listed in Table 4.

The twiddle constants and butterfly operations are selected by T. The twiddle constants selected through T are listed in Table 5.

Table 1 — Details of Interfacing Signals

Symbol	Status	Description
DATA	Input	20-bit data bus.
ADDRESS	Input	6-bit address lines.
Clock	Input	Clock signal for synchronization of the operation.
Reset	Input	Active high synchronous reset. On reset, initializes the operation at default level. Address lines are initialized at 000000b and others signals are maintained the state at high impedance level.

Table 2 — Signals Generated by Controller

Symbol	Width	Description
D	16	A data line carries the immediate data bits.
WREN	1	Register write enable: Active high signal enables the register to write the information in specified register.
WAD	3	Register write address: Denotes the address of register to write the information in it.
RAD	3	Register read address: Denotes the address of register to read information from it.
SEL	2	Select lines to select the operand.
XCH	1	Enables the data transfer between two register files. XCH= 0b transfer the data from Rr to Ri XCH= 1b transfer the data from Ri to real Rr
T	2	Select the stage of FFT operation.

Table 3 — Registers Address for Read, Write Operations

WAD	RAD	Rm	Rr	Ri
000b	000b	R0	Rr0	Ri0
001b	001b	R1	Rr1	Ri1
010b	010b	R2	Rr2	Ri2
011b	011b	R3	Rr3	Ri3
100b	100b	R4	Rr4	Ri4
101b	101b	R5	Rr5	Ri5
110b	110b	R6	Rr6	Ri6
111b	111b	R7	Rr7	Ri7

Instruction Set

Instructions are available to perform the trivial as well as complex arithmetic on operand. Instruction set is shown in Table 6. The 16-bit immediate data is indicated by "nn". SRC indicates the source and DST points the destination. X denotes the BU stage. Twiddle constants are implicitly available in the instruction. Here the memory is not used for load and store operation. The source and destination address of the registers are mention in the instruction itself. This saves the load and store time with off chip memory. Each instruction takes 2 cycles to decode and execute. Here data is represented in 16-bit simple 2's complement form.[15] All floating-point operations are performed as described by Kulkarni *et al.*[16]

Butterfly Unit

BU design reported by Kulkarni *et al.*[16] uses fused floating-point addition-subtraction (FFAS), FMULT and four 4:1 multiplexer. However, in this FFAS unit, exponent comparator, compares two exponents by taking difference between them. If this difference is too large, then the mantissa of the number having smaller exponent will be insignificant and truncated after the mantissa shifted more than 16 bits. Hence this logic sets operand having smaller exponent to zero value. Therefore, additional path is proposed in the FFAS design to skip FFAS algorithm and result is

Table 4 — Operands for Register Write in Main Register File and Operands for DFFAS

Operand for main Register file		Operands for DFFAS			
SELm	Operand	SELr	X	SELi	Y
00	Immediate data sequence (D)	00	Rm	00	Rm
01	Output from FMULT 1	01	Rr	01	Rr
10	Output from FMULT 2	10	Ri	10	Ri
11	0000 H	11	0000H	11	0000H

Table 5 — Twiddle Constants

W_N^{nk}	Twiddle Constant
W_8^0	1
W_8^2	-j
W_8^1	$0.707 - j0.707$
W_8^3	$-0.707 - j0.707$

Table 6 — Instruction Set

Instruction	Description
Load Rm, ##nn	Load immediate 16-bit data in main register
BU X SRC1, SRC2, DST1, DST2	BU stands for butterfly computation. X indicates the stage (0 to 3), SRC and DST from Rm, Rr or Ri. BU0 R0, R1, Rr0, Ri0 perform R0±R1, store the sum in Rr0 and difference in Ri0. However, SRC1/DST1 is either Rm or Rr. Similarly, SRC2/DST2 is either Rm or Ri.
BU X #0, SRC2, DST1, DST2	There is a special case in which the SRC1 is '0' and DST1 and DST2 are same as described above.
BU X SRC1,#0,DST1,DST2	There is also a special case in which the SRC2 is '0' and DST1 and DST2 are same as described above
Mov Rr,Ri	Copy the contents of Ri in to Rr.
Mov Ri,Rr	Copy the contents of Rr, Ri.
Out Rr,Ri	Read the contents of Rr, Ri
Halt	Termination of Program

Table 7 — Decision Table for Special Cases

Input X,Y	Sum	Difference
$X \neq 0$,$Y \neq 0$	X+Y	X-Y
$X \neq 0$,Y= 0	X	X
X =0 ,$Y \neq 0$	Y	−Y
X =0 ,Y= −1	Y	−Y = 1
X = 0 ,Y= 0	0	0

set to predefined value. Operands -1d or 0d or 1d are the frequently used coefficient in FFT computation. Hence additional path for operands -1d,0d and 1d is introduced. Additional path comprises magnitude comparator and multiplexers. Magnitude comparator compares the operand with -1d, 0d and 1d. The output of comparator enables the multiplexers to set sum/difference to predefined value as mentioned in decision Table 7. The FFAS design with this additional path is named as dual path fused floating point addition-subtraction (DFFAS) as shown in Fig. 2. Floating point addition-subtraction perfomed by DFFAS for the operands other than –1d,0d and 1d is similar to FFAS designed by Kulkarni *et al.*[16] This DFFAS is proposed at the place FFAS in BU designed by Kulkarni *et al.*[16] This new proposed BU is shown in Fig. 3.

The signal flow graph (SFG) of radix-2, 8-point FFT is shown in Fig. 4. It has regular and symmetric structure. This SFG has three stages. In stage 1, a single butterfly operation is present. In stage 2, two butterfly operations are present. Similarly, in stage 3, four butterfly operations are available. Therefore, a single BU is designed to perform all butterfly operations instead of using different processing elements for each stage butterfly operation. The twiddle constants required for butterfly operations are shown in Table 5 previously.

At the first stage, $W_8^0 = 1$. Hence trivial butterfly operation is performed on X and Y. They are added and subtracted. This butterfly operation is initiated by T = 00b. The sum and difference of X and Y are available at output R and I respectively. Second stage of SFG has two butterfly operations. Here $W_8^0 = 1$ and $W_8^2 = -j$. Hence trivial butterfly operation remained same. Another butterfly operation is performed with input Y. Input Y is multiplied by – 1. This second butterfly operation is selected when T= 01b. The product of multiplication is available at the output I. Third stage of SFG has four butterfly operations. The butterfly operations with twiddle constant $W_8^0 = 1$ and $W_8^2 = -j$ are similar to previous stages. The additional two butterfly operations are performed on input X and Y. In this stage input X denotes the real part and Y denotes the complex part of the intermediate computation available from previous stage. When T = 10b, intermediate computation from previous stage is multiplied by $W_8^1 = 0.707 - j0.707$. The similar complex multiplication of $W_8^3 = -0.707 - j0.707$ with intermediate computation is performed when T=11b. The real part of multiplication is available at output R and imaginary part of it is available at output I. The operational methodology[16] for butterfly operations is shown in Table 8.

FFT Computation and Error Analysis

FFTs of input sequences shown in Table 9 are computed using designed FFT processor. The computation is performed using decimation in time (DIT) as well as decimation in frequency (DIF). A computational program is written using the tailored instruction set shown in Table 6. The binary file of computational program is the part of design to test the functionality. Verilog entity of this binary file is named as program memory. Xilinx 14.7 is used to simulate the computational program. The FFTs of same input sequences are also computed by rounding the twiddle factors on proposed processor. To validate the result, FFTs of sequences are also

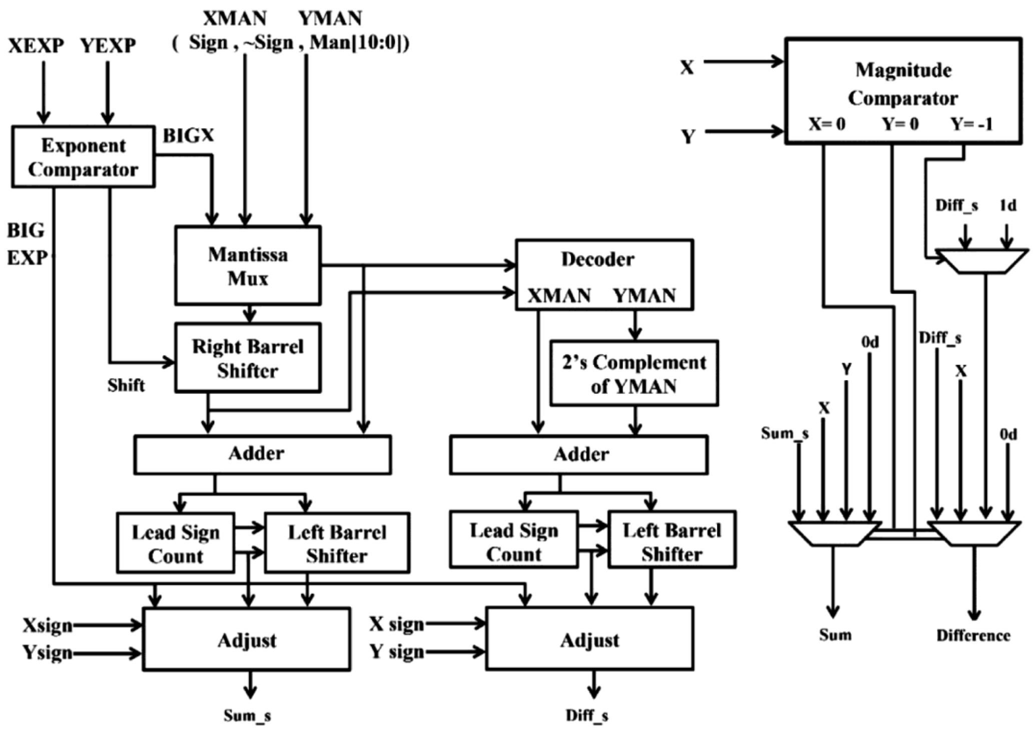

Fig. 2 — Dual Path Fused Floating Point Addition-Subtraction

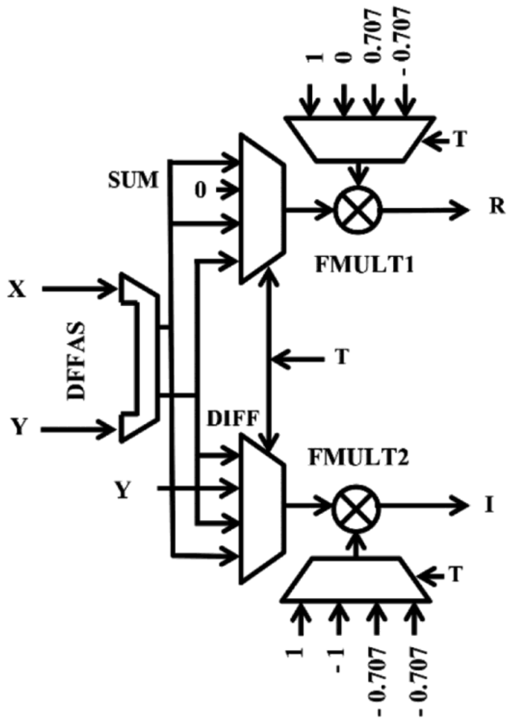

Fig. 3 — Radix-2,8-point Butterfly Unit

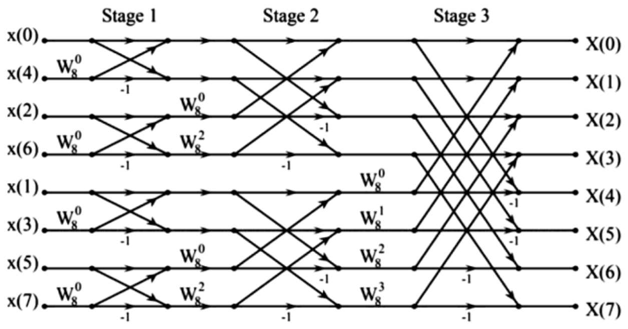

Fig. 4 — Signal Flow Graph for Radix-2,8-Point FFT

simulated using Scilab. The magnitude X[k] and phase < X[k] of FFT output is used to compare the result. The magnitudes and phases of the FFT ouputs are calculated using Eqs 1 & 2.

$$|X[k]| = \sqrt{X_{real}^2 + X_{imaginary}^2} \quad \ldots(1)$$

$$< X[k] = \tan^{-1}\frac{X_{imaginary}}{X_{real}} \quad \ldots(2)$$

These calculated magnitudes and phases values are compared with their Scilab simulated values. The comparison is in terms of mean square error (MSE). MSE is calculated using Eq. 3.

$$MSE = \frac{1}{N}\sum_{k=0}^{N-1} E(|X(k) - X_{calculated} \quad (k)|)^2 \ldots(3)$$

$X(k)$ is simulated value using Scilab. $X_{calculated}$ (k) is calculated value using proposed FFT processor. N represents the numbers of computations. The MSE is calculated for each sequence without and with rounding the twiddle constant as shown in Table 9. The MSE, without

Table 8 — Operational Methodology

Inputs	Stage	Twiddle factor	FMULT 1 Inputs	FMULT 2 Inputs	Output R	Output I
X,Y	I T=00	$W_8^0 = 1$	$X+Y, 1$	$X-Y, 1$	$X+Y$	$X-Y$
$0,Y$	II T=01	$W_8^2 = -j$	0,0	$Y, -1$	‘0’	$-jY$
$X+jY$	III T=10	$W_8^1 = 0.707 - j0.707$	$X+Y$, 0.707	$X-Y$, -0.707	Real Part	Imaginary Part
$X+jY$	III T=11	$W_8^3 = -0.707 - j0.707$	$X-Y$, -0.707	$X+Y$, -0.707	Real Part	Imaginary Part

rounding and with rounding of twiddle constants are 3.16 and 5.45 respectively for magnitude. Similarly, the MSE are 0.29 and 0.66 without and with rounding the twiddle constants respectively for phase. The proposed architecture reduces the MSE by 41.95% and 55.76% in magnitude and phase respectively as compared against its simulation performed by rounding the twiddle constant. The standard deviation and standard error are also computed for MSE. The standard error for magnitude is 2.83 in case of twiddle constants are rounded off and 1.14 otherwise. Similarly, the standard errors for phase values are 0.16 and 0.39 without and with rounding the twiddle constant respectively. Hence the proposed architecture reduces the standard error by 59.71% and

Table 9 — MSEs of FFT Computations

Sr No.	Input Sequence	Domain	MSE in Magnitude		MSE in Phase	
			Without Rounding	With Rounding	Without Rounding	With Rounding
1	1,2,3,4,4,3,2,1	DIT	0.006563	0.018	0.044063	0.353
2	1,2,3,4,4,3,2,1	DIF	0.003525	0.018	0.054141	0.349741
3	1,1,1,1,-1,-1,-1,-1	DIT	0.032038	0.023	0.02905	0.966077
4	1,1,1,1,-1,-1,-1,-1	DIF	0.028061	0.05698	0.004449	0.348237
5	1,-1,1,-1,0,0,0,0	DIT	0.006875	0.0065	0.00897	0.039812
6	1,-1,1,-1,0,0,0,0	DIF	0.007069	0.006432	0.006554	0.039812
7	2,1,2,1,2,1,2,1	DIT	0	0	0	0
8	2,1,2,1,2,1,2,1	DIF	0	0	0	0
9	1,2,3,2,1,2,3,2	DIT	0	0	0	0
10	1,2,3,2,1,2,3,2	DIF	0	0	0	0
11	1,1,1,1,0,1,1,1	DIT	0	0	0	0
12	1,1,1,1,0,1,1,1	DIF	0	0	0	0
13	1,2,4,8,16,32,64,128	DIT	29.00963	59.14134	0.007001	0.034374
14	1,2,4,8,16,32,64,128	DIF	28.23196	62.15168	0.005252	0.035111
15	128,64,32,16,8,4,2,1	DIT	7.722965	15.83844	0.001468	0.129015
16	128,64,32,16,8,4,2,1	DIF	23.21319	15.83844	0.013673	0.02153
17	64,32,16,8,4,2,1,0	DIT	1.931382	3.998601	0.001428	0.020782
18	64,32,16,8,4,2,1,0	DIF	1.720276	3.998601	0.000643	0.020782
19	0,1,2,1,0,-1,-2,-1	DIT	0.040316	0.10663	0	0.61685
20	0,1,2,1,0,-1,-2,-1	DIF	0.047816	0.10663	0.308644	0.61685
21	2,1,0,-1,-2,-1,0,1	DIT	0.301449	0.462885	0	0
22	2,1,0,-1,-2,-1,0,1	DIF	0.328785	0.462885	0.063263	0
23	0,1,2,3,4,5,6,7	DIT	0.124693	0.103462	0.019086	0.040108
24	0,1,2,3,4,5,6,7	DIF	0.288528	0.103462	0.083398	0.040108
25	7,6,5,4,3,2,1,0	DIT	0.124693	0.103462	0.776286	0.040108
26	7,6,5,4,3,2,1,0	DIF	0.126876	0.103462	0.023278	0.040108
27	16,8,4,2,1,0.5,0.25,0	DIT	0.638105	0.13083	0.024978	0.047688
28	16,8,4,2,1,0.5,0.25,0	DIF	1.026858	0.672341	0.029083	0.105215
29	-1,-1,-1,-1,1,1,1,1	DIT	0.030933	0.288264	3.609475	10.85544
30	-1,-1,-1,-1,1,1,1,1	DIF	0.066433	0.025874	3.687191	5.137126
		Mean	3.168634	5.458873	0.293379	0.663262
		Standard Error	1.14	2.83	0.16	0.39
		Standard deviation	8.18	15.53	0.92	2.14

58.97% in magnitude and phase respectively as compared against the fixed point representation of twiddle constant.

Hardware Implementation and Comparison of Result

Verilog codes of DFFAS, BU and proposed architecture of FFT processor are synthesized and placed using Mentor-Graphics - Oasys for 45 nm technology of nangate open cell library. Operating conditions are set to typical values. Authors have also synthesized the Verilog codes of the discrete design of floating point addition-subtraction and FFAS. In discrete design, common logic i.e., exponent comparator, mantissa mux and right barrel shifter are not shared. Similarly, in FFAS, no additional path is used. The comparative statistics of synthesized result is shown in Table 10. Discrete addition-subtraction design consumes 11422 µm^2 area with the delay of 1.47 ns. Similarly, FFAS and DFFAS design contributes area 10330 µm^2 and 10836 µm^2 respectively. FFAS and DFFAS design causes delay of 1.56 ns and 1.63 ns respectively. Area and delay of DFFAS are increased by 4.66% and 4.06% respectively as compared with FFAS.[16] This addition in area and delay is due to the additional pathused in DFFAS. Proposed BU design reports a delay of 3.51 ns with placement area of 20423 µm^2. Comparison of butterfly designs with previously reported work[8,10,11,16] is shown in Table 11. The proposed BU design reduces area by 14.58% with the additional delay of 1.99% as compared with authors previous work.[16] Similarly, the proposed BU design reduces area by 18.89% and delay by 5.13% as compared with the previous work reported by Kaivani *et al.*[10]. In addition to this, work reported by Kaivani *et al.*[10] computes one halves with five operand adder and two dot products. However proposed BU computes two halves with single DFFAS and two FMULT. The trade-offs between area and delay are usual conflicts. Hence the second order area time complexity parameter AT^2 i.e. Area × Time^2 is mentioned in comparison. It is worth mentioning that proposed BU design has smallest AT^2. The work reported by Kaivani *et al.*[11] is based on redundant algorithm. Here additional logic is required to convert the data available in non-redundant form to redundant form and vice-versa. Redundant to non-redundant logic contributes the additional delay and area.

BU of proposed FFT processor takes two cycles are required to complete one butterfly operation which one more cycle to write back the result in register file. However, the BU designed by Noor *et al.*[13] takes 12 cycles to complete one butterfly operation and additional 6 cycles for memory read, write back and scaling process. Therefore total 18 cycles to complete one BU operation and is too large as compared with the proposed design. The Mentor Graphics Oasys-Nitro flow is used to place and route the proposed architecture of FFT processor. The logical hierarchical placement details of proposed FFT processor in Nitro is shown in Table 12. Design summary is shown in Table 13. Synthesized, placed and routed results show that proposed processor has die area of 37251 µm^2 at 60.86% chip utilization.

Table 10 — Comparative Statistics of Floating-point Addition and Subtraction

Parameter	DFFAS Proposed	Discrete Addition –subtraction[16]	FFAS[16]
Technology	Nangate Open Cell Library 45nm	Nangate Open Cell Library 45nm	Nangate Open Cell Library 45nm
Area (A) in (µm^2)	10836	11422	10330
Delay (T) in (ns)	1.63	1.47	1.56
AT^2 in (mm^2 ns^2)	0.028	0.024	0.025

Table 11 — Comparison of Butterfly Unit

Parameter	Proposed Design	SwartzlanderJr *et al.*[8]	Kaivani *et al.*[10]	Kaivani *et al.*[11]	Kulkarni *et al.*[16]
Technology	Free PDK Nangate Open cell 45nm Lib	45nm Bulk CMOS Standard Lib	45nm Opennangate	STM CMOS 90nm Liband Scaled to 45nm	Free PDK Nangate Open cell 45nm Lib
Area (µm^2)	20423	47489	25182	93836	23910
Delay(ns)	3.51	4.00	3.70	2.59$^{\#}$	3.44
Area (µm^2) × Delay ^{2} (ns)^2	251613	759824	344741	629461	282941
Input-Output	Non-redundant	Non-redundant	Non-redundant	Redundant	Non-redundant

# Redundant to non-redundant and vice-versa logic and its delay is not included in the design

Table 12 — Logical Hierarchical Placement in Nitro

Module	No of Cells	Cell Area in μm^2
FFT (TOP)	6839	10753
FFAS	1031	1170.13
Fmult x 2	2746	4111.55
Register files x 3	1623	3758.58
Mux 2:1 x 2	32	59.58
Mux 4:1 x 7	602	555.11
Controller	584	878.06
Program Memory	219	220.51

Table 13 — Design Summary of FFT Processor

Library	Nangate Open cell Library
Technology	45nm
Die Area	37251μm^2
Max Clock Frequency	500MHz
Standard Cell utilization	60.86 %
Power	4654.2μW
Total Cycles to compute 8-point FFT	76 cycles

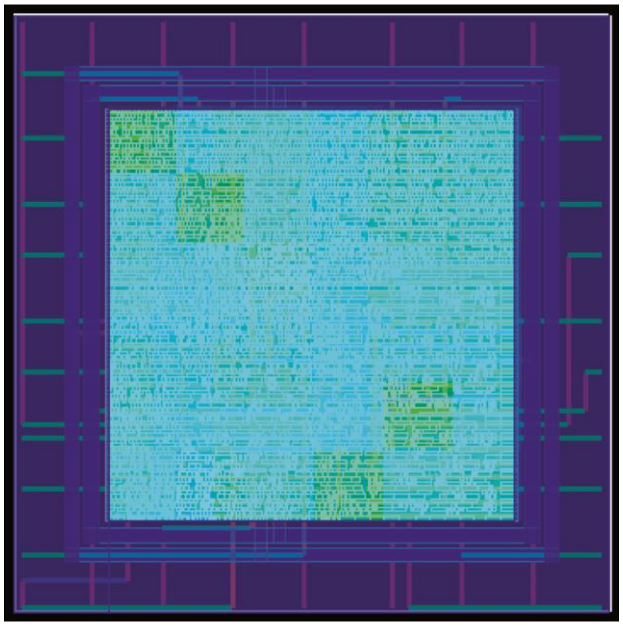
Fig. 5 — Placements in Chip of FFT Processor

Proposed processor dissipates 4.65 mW power. Operating voltage is 0.85V. The maximum clock frequency applied to this processor is 500MHz. The placement of logical cells is shown in Fig. 5. For fair comparison of proposed application specific integrated circuit (ASIC) design of FFT processor with previously reported designs having different FFT sizes, area-time complexity ($AT^2{}_{norm}$) , stated by Diego *et al.*[14] is used. Area-time complexity is the second order normalised term and given by Eq. 4, in which A, s, N and T represents area, processing technology in μm, FFT sequence size and time respectively.

Table 14 — Comparative Statistics of Hardware Utilization and Cost Metrics

Parameter	Proposed Design	Xiao *et al.*[6]	Velncia *et al.*[7]	Noor *et al.*[13]	Diego *et al.*[14]
Technology (T) nm	45	130	45	90	180
Area (A) μm^2	37251	2700000	348100	198404	740000
Voltage V	0.85	1.2	1.1	1.2	1.8
Clock Rate MHz	500	40	317	100	505
Power (D_p)mW	4.65	35.7	10	3.44	192
Data Length	16	10	16	16	8
FFT Points (N)	8	8192	32	128	12
$AT^2{}_{norm}$ mm^2 ns^2	0.296	0.394	1.73	0.594	0.242
$AT^2{}_{norm}$ D_p mm^2 ns^2 mW	1.37	14.09	17.32	2.03	46.42

$$AT^2{}_{norm} = \frac{Area}{N\,(^{s}/_{0.18})^2}T^2 \qquad \ldots(4)$$

The hardware cost metric is represented by the product of $AT^2{}_{norm}$ and power (D_p).[14] The hardware cost metric of proposed processor is 1.37. Comparative statistics of hardware utilization and cost metric with previous work[6,7,13,14] is given in Table 14. The proposed processor dissipates more power as compared with the ASIC design reported by Noor *et al.*[13] It is worth mentioning that proposed processor has lowest hardware cost metric and $AT^2{}_{norm}$. The hardware cost metric is 32.51 % less as compared with hardware cost metric of ASIC design given by Noor *et al.*[13]

Conclusions

The proposed FFT processor can be suitably suitable to adopt in radix–r pipelined split radix architecture for small independent, radix-2, 8-point computation. Twiddle constants are implicitly available in instructions to avoid the additional fetch cycle for them. Intermediate computational result are stored in register files which saves the load and store time required in memory-based architecture. Computational unit i.e., BU of proposed FFT processor is formatting smaller. It replaces a set of two five operand adder and two multipliers by dual path fused floating point addition-subtraction, two floating point multiplier as compared with previous work. The proposed BU performs arithmetic computation in floating point form to reduce the nonlinearities. Hence the proposed architecture reduces the MSE by 41.95% and 55.76% in magnitude and phase respectively as compared with computations performed by rounding the twiddle constants. The proposed processor also offers the flexibility to compute FFT in time and frequency domain without changing the BU design. It is also observed that hardware cost metric of the proposed architecture is 32.51% less than previous work.

References

1 Yu-Wei Lin, Hsuan-Yu Liu and Chen-Yi Lee, "A 1-GS/s FFT/IFFT Processor for UWB Applications," *IEEE Journal of Solid-State Circuits*, **40(8)** (2005) 1726–1735.

2 Yu-Wei Lin and Chen-Yi Lee, "Design of an FFT/IFFT Processor for MIMO OFDM Systems,"*IEEE Transactions on circuits and systems –I regular papers,* **54(4)** (2007) 807–815.

3 Guichang Zhong, Fan Xu and Alan N. Willson, "A Power-Scalable Reconfigurable FFT/IFFT IC Based on a Multi-Processor Ring,"*IEEE Journal of Solid-State Circuits*, **41(2)** (2006) 483–495.

4 Xuan Guan,Yunsi Fei and Hai Lin, " Hierarchical Design of an Application-Specific Instruction Set Processor for High-Throughput and Scalable FFT Processing," *IEEE Transactions on Very Large Scale Integration (vlsi) Systems*, **20(3)** (2012) 551–563.

5 Chu Yu and Mao-Hsu Yen, "An Area Efficient 128 to 2048/1536-Point Pipeline FFT Processor for LTE and Mobile Wi-Max System,"*IEEE Transactions on Very Large Scale Integration (vlsi) Systems*, **23(9)** (2015) 1793–1800.

6 Hao Xiao, XiangYin, Ning Wu, Xin Chen, Jun Li and Xiaoxing Chen, " VLSI design of low –cost and high-precision fixed-point reconfigurable FFT processor,"*IET Comput. Digit. Tech.*,**12(3)** (2018) 105–110.

7 Daniel Velncia and Amirhossein Alimohammad, "Compact and high-throughput parameterisable architectures for memory-based FFT algorithms," IET *Circuits Devices Syst.*,**13(5)** (2019) 696–703.

8 Earl E. SwartzlanderJr., and H. H. Saleh, "FFT implementation with fused floating-point operations," *IEEE Trans. Comput.*,**61(2)** (2012) 284–288.

9 J. Sohn and E. E. Swartzlander, Jr., "Improved architectures for a floating-point fused dot product unit," in *IEEE Transactions on circuits and systems –I regular papers*,**59(10)** (2012) 2285–2291.

10 Amir Kaivani and Seokbum Ko "Area efficient Floating-Point butterfly Architecture Based on multi operand adders," *Electronics Letter*,**51(2)** (2015) 895–897.

11 Amir Kaivani and Seokbum Ko "Floating-Point Architecture Based on Binary Signed-Digit Representation," *IEEE Transaction on VLSI Systems*, **24(3)** (2016) 1208–1211.

12 Klaus Schneider , Adrian Willenbucher, " A New Algorithm for Carry free Addition of Binary Signed Digit numbers,"*IEEE Int. Symp. On Field Programmable Custom Computing Machines*, (2014) 44–51.

13 Safwat Mostafa Noor , Eugene John, and Manoj Panday , "Design and Implementation of an Ultralow-Energy FFT ASIC for Processing ECG in Cardiac Pacemakers,"*IEEE Transaction on VLSI Systems* ,**27(4)** (2019) 983–987.

14 Diego F G Coelho, Renato J. Cintra, Nilanka Rajapaksha, Gihan J. Mendis, Arjuna Madanayake, Vassil S. Dimitrov, "DFT Computation using Gauss-Eisenstein Basis: FFT Algorithms and VLSI Architectures,"*IEEE transactions on computers*,**66(8)** (2017) 1442–1448.

15 Prasad Kulkarni, B G Hogade , Vidula Kulkarni, "Designing of Radix-2 Butterfly for Digital Signal Processor for FFT Computation," *Proceedings of third International Conference on Information and Communication Technology for Intelligent Systems, Smart Innovation, Systems and Technologies* ,**107** (2019) 603–610.

16 Prasad Kulkarni, B G Hogade, Vidula Kulkarni, " ASIC Design of Butterfly Unit based on Non-redundant and Redundant Algorithm,"*Iranian Journal of Electrical and Electronics Engineering*, accepted for publication, in press.IJEEE.2021:**17(1):**1809–1809.