IEICE Electronics Express, Vol.16, No.13, 1–6

LETTER

# Implementation of a radix-$2^k$ fixed-point pipeline FFT processor with optimized word length scheme

**Long Pang[1], Shan Dong[1], Libiao Jin[1a)], Chen Yang[2], Bingyi Li[3], Yu Xie[3], Yizhuang Xie[3], and He Chen[3]**

**Abstract** To design a high-precision and low-complexity FFT/IFFT processor architecture, the optimum bit sizing technique in each stage is usually adopted. However, it is difficult to provide an accurate, fast word length scheme due to the diversity of FFT algorithms and the complexity of circuit structure. In this paper, we focus on the widely-used radix-$2^k$ Decimation-In-Frequency (DIF) Fast Fourier Transform (FFT) algorithm. Based on our previous research on fixed-point FFT Signal-to-Quantization-Noise Ratio (SQNR) assessment, an analytical expression of word lengths in different stages is deduced. We further put forward a word length optimization method based on the analytical expression. Pre-layout logic synthesis and power simulation are performed for comparison with some previous works. Eventually, we implement a 16384-point FFT processor in 0.13 µm technology. The proposed method yields more hardware resource benefit and saves more simulation time.

**Keywords:** radix-$2^k$ pipeline FFT, fixed point, quantization error analysis, word length optimization

**Classification:** Integrated circuits

## 1. Introduction

Fast Fourier transform (FFT) is one of the most fundamental algorithms used in digital signal processing area. Many applications such as orthogonal frequency division multiplexing (OFDM) [1, 2, 3, 4, 5, 6], long term evolution (LTE) [7, 8, 9, 10, 11, 12, 13] and ultra-wideband (UWB) systems [14, 15, 16, 17, 18, 19, 20] require an area efficient, high accuracy FFT processor. Traditional FFT architectures include: memory-based, pipelined, and array architectures. In particular, the pipelined FFT architecture has been mainly adopted due to its attractive properties, such as small chip area, high throughput, and low power consumption.

Many fixed-point pipeline FFT processors are designed in previous works. A 128- to 2048/1536-Point SDF pipeline FFT processor [21] is designed for LTE and mobile WiMAX systems. 12-bit data word length is selected based on fixed-point simulation. Radix-$2^3$ [22] and radix-$2^5$ [23] pipeline FFT processor are also studied to design a low-complexity FFT processor. The internal word length of 12 bit is selected using a fixed-point simulation prior to the hardware implementation [23]. A further attempt in word length selection is made in the implementation of a radix-4 MDC FFT/IFFT processor with variable length [24]. Based on fixed-point simulation, the input word length is fine-tuned to 8 bits and the output word length was 12 bits. Theoretical performance evaluation of SQNR/MSE of different FFT algorithms is discussed in previous works [25, 26, 27, 28]. They derive the output SQNR/MSE expression and verify the expression with fixed-point simulation.

Our previous work [29] discussed the SQNR assessment issues in radix-$2^2$ FFT algorithm. It is only a one-sided assessment of radix-$2^2$ algorithm under truncation case. In this work, we improve and extend the SQNR assessment to radix-$2^k$ algorithm. Both rounding and truncation cases are taken into consideration. Furthermore, we derive an analytical word length expression and propose a word length optimization method. FFT processor implementation results prove our method to be effective.

## 2. Radix-$2^k$ FFT algorithm and SDF architecture

The idea of radix-$2^k$ algorithms is to try to achieve both a simple butterfly and a reduced number of twiddle factor multiplications at the same time. As the order $k$ increases, more twiddle factors are replaced by constant factors. The essential difference between the radix-$2^k$ algorithms is the distribution of the twiddle factors. Table I shows the non-trivial twiddle factor number $n_i$ in stage $i$ in radix-$2^k$ algorithms.

**Table I.** Number of twiddle factors in radix-$2^k$ algorithm

| Algorithm | $n_i$ $(i = 1, 2, \ldots, \log_2 N)$ |
|---|---|
| Radix-2 | $N/2^i - 1$ |
| Radix-$2^2$ | $\begin{cases} (N/4^{i/2} - 1) \times 3 \times 4^{i/2-1} & \mathrm{mod}(i,2) = 0 \\ 0 & \mathrm{mod}(i,2) = 1 \end{cases}$ |
| Radix-$2^3$ | $\begin{cases} (N/8^{i/3} - 1) \times 7 \times 8^{i/3-1} & \mathrm{mod}(i,3) = 0 \\ 0 & \mathrm{mod}(i,3) = 1 \\ N/4 & \mathrm{mod}(i,3) = 2 \end{cases}$ |
| Radix-$2^4$ | $\begin{cases} (N/16^{i/4} - 1) \times 15 \times 16^{i/4-1} & \mathrm{mod}(i,4) = 0 \\ 0 & \mathrm{mod}(i,4) = 1 \\ N/4 & \mathrm{mod}(i,4) = 2 \\ 3N/4 & \mathrm{mod}(i,4) = 3 \end{cases}$ |

[1]School of Information and Communication Engineering, Communication University of China, Beijing 100024, China
[2]Huawei Hisilicon, Beijing, China
[3]Beijing Key Laboratory of Embedded Real-time Information Processing Technology, Beijing Institute of Technology, Beijing 100081, China
a) libiao@cuc.edu.cn

DOI: 10.1587/elex.16.20190181
Received March 20, 2019
Accepted April 22, 2019
Publicized June 4, 2019
Copyedited July 10, 2019



Copyright © 2019 The Institute of Electronics, Information and Communication Engineers

Memories and arithmetic logic units occupy most of area and power consumption which are the most crucial parameters of an FFT processor. Thus, we need a tradeoff between precision and circuit area. The word length optimization problem is expressed as follows:

$$\{b_1, b_2, \cdots, b_n\} = f(b_0, SQNR_{out}, NFFT). \quad (1)$$

Our goal is to optimize the word length $b_i$ of different FFT processing stages under a set of constraints: input word length $b_0$, output $SQNR$ and FFT length $NFFT$.

## 3. SQNR assessment for radix-2k fixed-point FFT

### 3.1 Modified SQNR assessment expression

In our previous work [29], we have reached an SQNR analytical expression of radix-$2^2$ fixed-point FFT. We re-list the output quantization noise power $P_E$, output signal power $P_X$, and output $SQNR$ expression as follows:

$$P_X = N \cdot (1/4)^{\sum_{i=1}^{v} T_i} \cdot \sigma_x^2, \quad (2)$$

$$P_E = P_A + P_M = N \cdot \sum_{i=1}^{v} \left(\frac{1}{4}\right)^{\sum_{j=i+1}^{v} T_j} 2^{v-i} \sigma_{ai}^2 + \sum_{i=1}^{v} \left(\frac{1}{4}\right)^{\sum_{j=i+1}^{v} T_j} 2^{v-i} \sigma_{mi}^2, \quad (3)$$

$$SQNR = \frac{P_X}{P_E} = \frac{N \cdot \left(\frac{1}{4}\right)^{\sum_{i=1}^{v} T_i} \sigma_x^2}{\left(N \cdot \sum_{i=1}^{v} \left(\frac{1}{4}\right)^{\sum_{j=i+1}^{v} T_j} 2^{v-i} \sigma_{ai}^2 + \sum_{i=1}^{v} \left(\frac{1}{4}\right)^{\sum_{j=i+1}^{v} T_j} 2^{v-i} \sigma_{mi}^2\right)}. \quad (4)$$

The variables are defined as follows:

- $\sigma_x{}^2$ is the variance of input signal.
- $\sigma_{ai}{}^2$ is the addition noise variance in stage $i$.
- $\sigma_{mi}{}^2$ is the complex multiplication noise variance in stage $i$.
- $b_0$ is the initial input word length of FFT and $b_i$ is the word length in stage $i$ ($i = 1, 2, \ldots, v = \log_2 N$).
- $T_i$ is the word length scaling variable in stage $i$.

According to addition operation rules, word length is expected to increase by 1 bit after one addition. Thus we define $T_i = 0$ if the word length increases by 1 bit after the butterfly operation in stage $i$. The relationship between $b_0$, $b_i$ and $T_i$ is described as follows:

$$b_i = b_0 + i - \sum_{j=1}^{i} T_j. \quad (5)$$

In order to establish the relationship between quantization noise variance and word length, we analysis the rounding and truncation issues based on the assumptions proposed in [30]. The round-off error range and corresponding quantization error variance when scaling a number to $b$ bit are listed in Table II.

Now the addition noise variance in both rounding and truncation issues is expressed as follows:

**Table II.** Round-off error range and corresponding variance

| | Error range for positive number | Error range for negative number | Variance*1 |
|---|---|---|---|
| Truncation | $[0, 2^{-b})$ | $(-2^{-b}, 0]$ | $2^{-b}/3$ |
| Rounding | $[0, 2^{-b}/2)$ | $(-2^{-b}/2, 0]$ | $2^{-b}/12$ |

$$\sigma_{ai}^2 = \begin{cases} N \cdot \alpha_i \cdot 2^{-2b_i}/12 & for\ rounding \\ N \cdot \alpha_i \cdot 2^{-2b_i}/3 & for\ truncation \end{cases} \quad \alpha_i = \begin{cases} 1 & b_i < b_{i-1} + 1 \\ 0 & b_i = b_{i-1} + 1 \end{cases}. \quad (6)$$

The variable $\alpha_i$ is defined according to addition operation rules.

A complex multiplication is usually composed of four real multiplications. In addition, we usually ensure that the data word length remains unchanged after a multiplication operation. Thus, the multiplication noise variance in both rounding and truncation issues can be expressed as follows:

$$\sigma_{mi}^2 = \begin{cases} n_i \cdot 2^{-2b_i}/3 & for\ rounding \\ n_i \cdot 4 \cdot 2^{-2b_i}/3 & for\ truncation \end{cases}. \quad (7)$$

$n_i$ is the number of non-trivial twiddle factors. We have revealed the value of $n_i$ above in Table I.

Although (4) is extended to both rounding and truncation issues, it is still not complete. For a simple example, if we use (4) to evaluate a 4-point radix-$2^2$ FFT in which no rounding or truncation occurs, according to (6), (7) and Table I the denominator of (4) will be zero. The $SQNR$ becomes infinite. This is undoubtedly out of reality. The total quantization noise should consist of two parts. One part is the quantization noise generated by the internal arithmetic operations of fixed-point FFT. The power of this part is shown above as (3). Another is the initial inherent quantization noise associated with the input fixed-point data. The quantization noise power of the input b0-bit fixed-point data can be expressed as follows:

$$P_{E\_ini} = \begin{cases} 2^{-2b_0}/12 & for\ rounding \\ 2^{-2b_0}/3 & for\ truncation \end{cases}. \quad (8)$$

By substituting (6), (7) and (8) into (4), the modified SQNR assessment expression is described as (9). It shows that rounding offers $10\log_2 12 - 10\log_2 3 \approx 6$ dB SQNR improvement compared with truncation. As we discuss above, the essential difference between the radix-$2^k$ algorithms is the distribution of the twiddle factors. Different radix-$2^k$ algorithms correspond to the different values of $n_i$ in the formula. Thus, the modified SQNR analytical form (9) is suitable for radix-$2^k$ algorithms.

$$SQNR = \frac{P_X}{P_{E\_ini} + P_A + P_M} = \begin{cases} \dfrac{(1/4)^{\sum_{i=1}^{v} T_i}}{\left(1 + \sum_{i=1}^{v} (1/4)^{\sum_{j=i+1}^{v} T_j - \sum_{k=1}^{i} T_k} \cdot N\alpha_i \cdot 2^{-3i} + \sum_{i=1}^{v} (1/4)^{\sum_{j=i+1}^{v} T_j - \sum_{k=1}^{i} T_k} \cdot 4n_i \cdot 2^{-3i}\right)} \cdot \dfrac{\sigma_x^2}{2^{-2b_0}/12} & rounding \\ \dfrac{(1/4)^{\sum_{i=1}^{v} T_i}}{\left(1 + \sum_{i=1}^{v} (1/4)^{\sum_{j=i+1}^{v} T_j - \sum_{k=1}^{i} T_k} \cdot N\alpha_i \cdot 2^{-3i} + \sum_{i=1}^{v} (1/4)^{\sum_{j=i+1}^{v} T_j - \sum_{k=1}^{i} T_k} \cdot 4n_i \cdot 2^{-3i}\right)} \cdot \dfrac{\sigma_x^2}{2^{-2b_0}/3} & truncation \end{cases} \quad (9)$$

### 3.2 SQNR error test

In this part, we perform an experiment to verify our modified SQNR expression. The SQNR error between real SQNR and the SQNR calculated from (9) is obtained.

It is time-consuming to obtain the real SQNR performance of an FFT processor by register transfer level (RTL) implementation. SystemC contains signed and unsigned fixed-point data types that can be used to accurately model hardware. Both rounding and truncation issues can be

modeled. Therefore, we apply SystemC platform to perform fixed-point simulation.

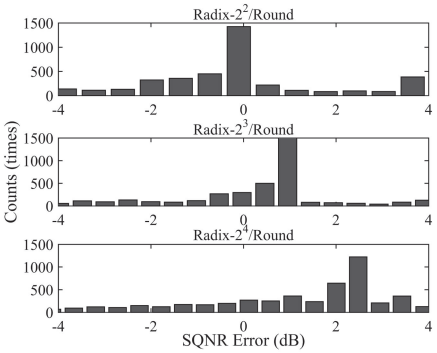
The modified analytical expression of the radix-$2^k$ FFT output SQNR is verified by the simulation-based error analysis. The SQNR error is obtained by subtracting the SQNR of the SystemC simulation from that of the analytical expression. Table III shows an example of the comparison. The word length scaling variable $T_i$ is generated randomly from $-2$ to 2. The input word length is 16 bit.

**Table III.** Example of the random test for 256-point radix-$2^2$ FFT

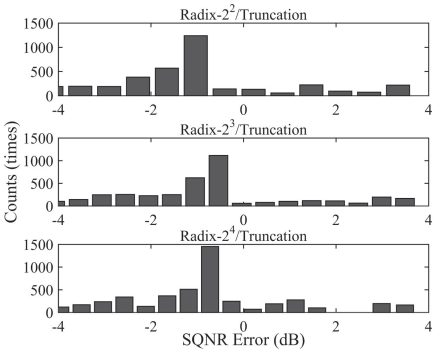
| No. | word length of stages | SQNR (dB) | | |
|---|---|---|---|---|
| | 01 02 03 04 05 06 07 08 | Sim.*1 | Est.*2 | Err. |
| 1 | 15 14 17 14 14 15 18 17 | 55.66 | 54.78 | 0.88 |
| 2 | 16 15 15 15 14 15 14 13 | 46.23 | 45.18 | 1.05 |
| 3 | 16 17 18 19 20 21 21 22 | 80.98 | 80.62 | 0.36 |
| 4 | 15 15 15 14 15 14 13 12 | 39.81 | 38.79 | 1.02 |
| 5 | 15 15 14 13 13 12 11 10 | 28.14 | 26.92 | 1.22 |
| 6 | 16 15 14 13 12 12 11 12 | 36.18 | 33.36 | 2.82 |
| 7 | 17 20 19 22 21 21 23 23 | 83.90 | 83.89 | 0.01 |
| 8 | 16 19 18 20 19 18 18 17 | 69.45 | 68.49 | 0.96 |

*1 SQNR obtained using SystemC fixed-point simulation.
*2 SQNR calculated using analytical expression (9).

Fig. 1 shows the histogram of the SQNR error with 5000 random tests for the 4096-point FFT of radix-$2^2$, radix-$2^3$ and radix-$2^4$ algorithm. Both rounding and truncation cases are tested. We choose chirp signal with white gauss noise as the input signal. The experiment result shows that the mean value of SQNR error is within 3 dB in all test scenarios.




(a) Rounding issue




(b) Truncation issue

**Fig. 1.** Histogram of the SQNR error with random word lengths.

## 4. Analytical word length expression and word length optimization method

### 4.1 Expression of internal word length $\{b_i\}$

We find that it is hard to derive the analytical form of sequence $\{T_i\}$ directly from (9). However, review (6), (7) and (9), the difference between $P_A$ and $P_M$ is the number of addition: $N\alpha_i$ and the number of non-trivial multiplication: $n_i$. According to Table I, the total number of multiplications is significantly less than the total number of additions. Therefore, in order to make it feasible to derive $\{T_i\}$, we perform an approximation as follows:

$$SQNR \approx P_X/(P_{E\_ini} + P_A). \tag{10}$$

Define that:

$$SQNR_0 = \begin{cases} 12 \cdot \sigma_x^2/2^{-2b_0} & for\ rounding \\ 3 \cdot \sigma_x^2/2^{-2b_0} & for\ truncation \end{cases}, \tag{11}$$

$$A_i = \alpha_i \cdot 2^{-3i}, \quad B = (1/4)^{\sum_{i=1}^{v} T_i}, \quad C_i = (1/4)^{\sum_{j=i+1}^{v} T_j - \sum_{k=1}^{i} T_k}.$$

Then (10) is expressed as follows:

$$SQNR = \frac{B}{\sum_{i=1}^{v}[C_i \cdot A_i] + 1} \cdot SQNR_0. \tag{12}$$

Define that:

$$Q = (1/4)^{-\sum_{i=1}^{v-1} T_i}, \quad K_i = (1/4)^{\sum_{j=i+1}^{v-1} T_j - \sum_{k=1}^{i} T_k},$$
$$P = \sum_{i=1}^{v-1} K_i A_i, \quad R = SQNR_0/SQNR, \quad x = (1/4)^{-T_v}. \tag{13}$$

Then (12) is induced as follows and x is the root of the equation:

$$x^2 + \frac{1}{Q \cdot A_v} \cdot x + \frac{P}{Q \cdot A_v} - \frac{R}{Q^2 \cdot A} = 0. \tag{14}$$

Finally, the expression of $T_i$ is derived as follows:

$$T_i = \begin{cases} \frac{1}{2}\log_2\left(\frac{R}{Q} - P\right) & \alpha_i = 0 \\ \frac{1}{2}\log_2\left(\frac{-1 + \sqrt{1 - 4A_i \cdot (Q \cdot P - R)}}{2A_i \cdot Q}\right) & \alpha_i \neq 0 \end{cases}. \tag{15}$$

For the reason of $x$ must be a positive number, the negative root is rejected.

The current stage scaling variable $T_i$ is closely related with $b_0$, $SQNR$ and the scaling variables of previous stages: $\{T_1, T_2, \ldots, T_{i-1}\}$. By substituting (15) into (5), the presentation of internal word length $\{b_i\}$ is finally obtained.

### 4.2 Word length optimization method

According to the derivation above, the internal word length $\{b_i\}$ can be directly calculated. However, the approximation performed in (10) may affect the accuracy and practicality of the calculated results to a certain extent. Considering that the modified SQNR assessment expression (9) is accurate enough, we set up a recursive feedback mechanism to ensure the calculated $\{b_i\}$ is practicable. This mechanism is summarized as a word length optimization

method. Pseudo code of the method is described as follows.

**Word length optimization method**

```
begin
  input b0, SQNR_ini, Nfft, Quantization_mode;
  while (SQNR_err ≥ 3)
  {
    calculate {T_i} using (15);
    substitute {T_i} into (9) to obtain SQNR_est;
    calculate the SQNR error of current solution {T_i} by:
      SQNR_err = SQNR_est − SQNR;
    revise the input SQNR_ini constraint by:
      SQNR_ini = SQNR_ini − SQNR_err;
  }
  transform {T_i} to {b_i} using (5);
  output {b_i};
end
```

The proposed method is completely based on the derived analytical expressions, so it takes a short time to get the word length scheme $\{b_i\}$.

## 5. Pre-layout comparison

The authors in [21] adopt fixed-point simulation for the selection of word length. The input, internal and output word lengths are all set to 12 bit. We use the proposed method to generate a set of equivalent word length schemes. Table IV shows the memory and SQNR comparison result. The memory counts only refer to the internal data buffer RAM/register, not including twiddle factor ROM. Compared with the inflexible 12 bit scheme, our schemes save more memory resource, meanwhile ensuring that the SQNR performance remains unchanged. For the 2048-point case, our method reduces the memory occupation by nearly 17%.

**Table IV.** Memory and SQNR comparison between [4] and proposed method

| FFT length | Word length scheme | Memory counts (bit) | SQNR (dB) |
|---|---|---|---|
| 128 | 12 12 12 12 12 12 12 12 | 3072 | 36.6 |
| | 9 10 10 11 11 12 12 13 | 2618 | 37.2 |
| 256 | 12 12 12 12 12 12 12 12 12 | 6144 | 34.1 |
| | 9 10 11 11 11 11 12 12 13 | 5370 | 35.4 |
| 512 | 12 12 12 12 12 12 12 12 12 12 | 12288 | 30.1 |
| | 9 10 10 10 10 11 11 12 12 13 | 10298 | 30.9 |
| 1024 | 12 12 12 12 12 12 12 12 12 12 12 | 24576 | 27.3 |
| | 10 10 10 10 10 11 11 11 12 12 13 | 20602 | 27.4 |
| 1536 | 12 12 12 12 12 12 12 12 12 12 12 | 36864 | 25.1 |
| | 10 10 10 10 10 10 10 11 12 13 13 | 30752 | 25.5 |
| 2048 | 12 12 12 12 12 12 12 12 12 12 12 12 | 49152 | 24.2 |
| | 10 10 10 10 10 10 10 11 11 12 13 13 | 41022 | 24.1 |

Based on the customized word length schemes discussed above, we replicate the variable-length SDF FFT described in [21] including the radix-3 butterfly unit design. However, due to the memory hardware-sharing mechanism in [21], the word length scheme for 1536-point FFT in Table IV cannot be realized. For fairly comparison, we only compare the power consumption of the FFT lengths corresponding to $2^k$. We synthesize the design with Synopsys DC (design compiler) using SMIC (Semiconductor Manufacturing International Corporation) 90 nm technology. We perform the power analysis with Synopsys PrimeTime PX under the same clock constraint. The comparison result is shown in Table V. The result shows that our method efficiently converts the word length optimization to area and power reduction. Chip area is reduced by about 22%. The 2048-point power consumption is reduced by about 27%.

**Table V.** Area and power comparison

| Design | | [21] | This work |
|---|---|---|---|
| Word length | | 12 bit | Proposed in Table IV. |
| Technology | | 90 nm | 90 nm |
| Supply voltage | | 0.9 V | 0.9 V |
| Working frequency | | 40 MHz | 40 MHz |
| Area | | $0.87 * 0.9\,mm^2$ | $0.61\,mm^2$ |
| Power | 2048-point | 6.43 mW | 4.66 mW |
| | 1024-point | 5.48 mW | 3.81 mW |
| | 512-point | 3.08 mW | 2.31 mW |
| | 256-point | 2.64 mW | 1.99 mW |
| | 128-point | 2.35 mW | 1.82 mW |

## 6. Implementation of a fixed-point FFT processor

According to the word length optimization method discussed above, a 16384-point FFT processor is implemented. We use the proposed method to generate a word length scheme which is equivalent to a 24 bit-in-24 bit-out regular scheme. The final word length configuration comparison is shown in Table VI. Our method significantly reduces the memory usage by 26.1%.

**Table VI.** Word length scheme comparison for a 16384-point FFT implementation

| Word length scheme | $b_0$ $b_1$ $b_2$ $b_3$ $b_4$ $b_5$ $b_6$ $b_7$ $b_8$ $b_9$ $b_{10}$ $b_{11}$ $b_{12}$ $b_{13}$ $b_{14}$ | Memory (bit) |
|---|---|---|
| Regular method | 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 | 786432 |
| Proposed method | 16 17 18 18 19 20 21 21 22 23 24 25 26 27 27 | 581004 |

Fig. 2 shows the circuits architecture of the 16384-point fixed-point FFT. It is designed based on SDF architecture and it consists of three main parts: memory units, arithmetic units and control units. Memory units include the feedback buffer RAM and the twiddle factor ROM. Leveraging the symmetry of twiddle factors, the proposed

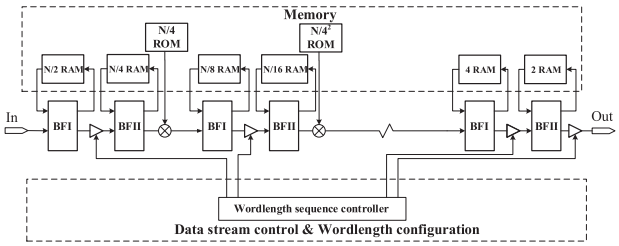



**Fig. 2.** Circuit architecture block diagram.

design requires only one quarter as much ROM space for both real and imaginary parts. Arithmetic units are butterfly operation units (adders and subtractors) and multipliers. Control units configure the word length sequence and control the data stream.

The design is modeled in VHDL language and synthesized with the Semiconductor Manufacturing International Corporation (SMIC) 0.13 µm standard cell library. Fig. 3 shows the primary layout of the chip.

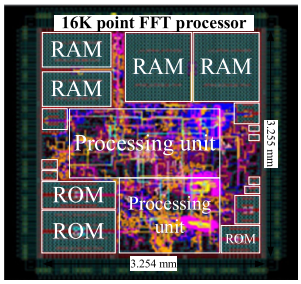



**Fig. 3.** Layout of 16K-point Radix-$2^2$ pipeline FFT processor.

**Table VII.** Specifications of the chip.

| | | | |
|---|---|---|---|
| Technology | 130 nm CMOS | | |
| Max Frequency | 125 MHz | | |
| Core Area | $3.255 \times 3.254\ mm^2$ | | |
| IO supply voltage | 3.3 V | | |
| Internal voltage | 1.2 V | | |
| Pin Count | 256 | | |
| Package | LQFP256 | | |
| Power with IO pads @ 100 MHz | I/O pads | 49.9 mW | |
| | Registers | 14.3 mW | 99.4 mW |
| | Memory | 67.2 mW | |
| | Logic | 17.9 mW | |
| | Total | 149.3 mW | |

Table VII summarizes the main specifications of the chip. The total power consumption seems a little high.

However, to fairly compare our implementation with previous works, normalized power/FFT point [31] is employed as indices to reflect the energy efficiency.

$$\begin{aligned} &\textit{Normalized Power per FFT point} \\ &= \frac{\textit{Power} \times (125/f)}{(\textit{FFTsize}/16384) \times (\textit{Voltage}/1.2)^2 \times \left(\frac{2}{3}\frac{WL}{16} + \frac{1}{3}\left(\frac{WL}{16}\right)^2\right)} \end{aligned} \quad (16)$$

$WL$ is the word length adopted in the FFT design. Here we take 16 bit as the word length corresponding to our design.

Thus, the normalized power consumption of our work is 149.3 mW, while that of [21] is 368.2 mW. There is no doubt that our word length configuration method is more efficient.

## 7. Conclusion

Fixed-point FFT is adopted by plenty of Digital Signal Processing (DSP) applications. How to deal with the word length optimization issue is a problem all the time. In this paper, we extend the SQNR assessment to radix-$2^k$ algorithm under both rounding and truncation cases. We further derive the analytical word length expression based on this modified SQNR assessment expression. A word length optimization method is proposed accordingly. Pre-layout comparison with a previous work and a real implementation of a fixed-point FFT processor show the versatility of our method. In conclusion, the proposed method rapidly and accurately generates word length optimization schemes which realize an efficient trade-off between FFT performance and hardware expenditure.

## Acknowledgments

This study is supported by “the Fundamental Research Funds for the Central Universities.

### References

[1] S. Li, *et al.*: “A 128/256-point pipeline FFT/IFFT processor for MIMO OFDM system IEEE 802.16e,” Proc. IEEE Int. Symp. Circuits Syst. (2010) 1488 (DOI: 10.1109/ISCAS.2010.5537355).
[2] S. He and M. Torkelson, “Designing pipeline FFT processor for OFDM (de)modulation,” 29 (1998) 257 (DOI: 10.1109/ISSSE.1998.738077).
[3] Y.-W. Lin and C.-V. Lee: “Design of an FFT/IFFT processor for MIMO OFDM systems,” IEEE Trans. Circuits Syst. I, Fundam. Theory Appl. **54** (2007) 807 (DOI: 10.1109/TCSI.2006.888664).
[4] F.-L. Yuan, *et al.*: “A 256-point dataflow scheduling 2 × 2 MIMO FFT/IFFT processor for IEEE 802.16 WMAN,” (2008) 309 (DOI: 10.1109/ASSCC.2008.4708789).
[5] Y.-W. Lin, *et al.*: “A dynamic scaling FFT processor for DVB-T applications,” IEEE J. Solid-State Circuits **39** (2004) 2005 (DOI: 10.1109/JSSC.2004.835815).
[6] K. Maharatna, *et al.*: “A 64-point Fourier transform chip for high-speed wireless LAN application using OFDM,” IEEE J. Solid-State Circuits **39** (2004) 484 (DOI: 10.1109/JSSC.2003.822776).
[7] C.-H. Yang, *et al.*: “Power and area minimization of reconfigurable FFT processors: A 3GPP-LTE example,” IEEE J. Solid-State Circuits **47** (2012) 757 (DOI: 10.1109/JSSC.2011.2176163).
[8] S. He and M. Torkelson: “Design and implementation of a 1024-point pipeline FFT processor,” Proc. IEEE Custom Integrated Circuits Conf. (CICC’98) (1998) 131 (DOI: 10.1109/CICC.1998.694922).
[9] J. O’Brien, *et al.*: “A 200 MIPS single-chip 1 k FFT processor,” IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. (1989) 166 (DOI: 10.1109/ISSCC.1989.48244).
[10] B. M. Baas: “A low-power high-performance 1024-point FFT processor,” IEEE J. Solid-State Circuits **34** (1999) 380 (DOI: 10.1109/4.748190).
[11] A. Wang and A. Chandrakasan: “A 180-mV subthreshold FFT processor using a minimum energy design methodology,” IEEE J. Solid-State Circuits **40** (2005) 310 (DOI: 10.1109/JSSC.2004.837945).
[12] Y. Chen, *et al.*: “A 2.4-Gsample/s DVFS FFT processor for MIMO OFDM communication systems,” IEEE J. Solid-State Circuits **43** (2008) 1260 (DOI: 10.1109/JSSC.2008.920320).

[13] K.-S. Chong, *et al.*: "Energy-efficient synchronous-logic and asynchronous-logic FFT/IFFT processors," IEEE J. Solid-State Circuits **42** (2007) 2034 (DOI: 10.1109/JSSC.2007.903039).
[14] G. Zhong, *et al.*: "A power-scalable reconfigurable FFT/IFFT IC based on multi-processor ring," IEEE J. Solid-State Circuits **41** (2006) 483 (DOI: 10.1109/JSSC.2005.862344).
[15] M. Seok, *et al.*: "A 0.27 V 30 MHz 17.7 nJ/transform 1024-pt complex FFT core with super-pipelining," IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. (2011) 342 (DOI: 10.1109/ISSCC.2011.5746346).
[16] W.-C. Yeh and C.-W. Jen: "High-speed and low-power split-radix FFT," IEEE Trans. Signal Process. **51** (2003) 864 (DOI: 10.1109/TSP.2002.806904).
[17] L. Jia, *et al.*: "A new VLSI-oriented FFT algorithm and implement," Proc. 11th Annu. IEEE Int. ASIC Conf. (1998) 337 (DOI: 10.1109/ASIC.1998.723029).
[18] J. Garcia, *et al.*: "VLSI configurable delay commutator for a pipeline split radix FFT architecture," IEEE Trans. Signal Process. **47** (1999) 3098 (DOI: 10.1109/78.796442).
[19] Y. Jung, *et al.*: "New efficient FFT algorithm and pipeline implementation results for OFDM/DMT applications," IEEE Trans. Consum. Electron. **49** (2003) 14 (DOI: 10.1109/TCE.2003.1205450).
[20] Y.-W. Lin, *et al.*: "A 1-GS/s FFT/IFFT processor for UWB applications," IEEE J. Solid-State Circuits **40** (2005) 1726 (DOI: 10.1109/JSSC.2005.852007).
[21] C. Yu and M.-H. Yen: "Area-efficient 128- to 2048/1536-point pipeline FFT processor for LTE and mobile WiMAX systems," IEEE Trans. Very Large Scale Integr. (VLSI) Syst. **23** (2015) 1793 (DOI: 10.1109/TVLSI.2014.2350017).
[22] T. Cho and H. Lee: "A high-speed low-complexity modified radix-25 FFT processor for high rate WPAN applications," IEEE Trans. Very Large Scale Integr. (VLSI) Syst. **21** (2013) 187 (DOI: 10.1109/TVLSI.2011.2182068).
[23] M. Ayinala and K. K. Parhi: "FFT architectures for real-valued signals based on radix-23 and radix-24 algorithms," IEEE Trans. Circuits Syst. I, Reg. Papers **60** (2013) 2422 (DOI: 10.1109/TCSI.2013.2246251).
[24] K.-J. Yang, *et al.*: "MDC FFT IFFT processor with variable length for MIMO-OFDM systems," IEEE Trans. Very Large Scale Integr. (VLSI) Syst. **21** (2013) 720 (DOI: 10.1109/TVLSI.2012.2194315).
[25] O. Sarbishei and K. Radecka: "Analysis of mean-square-error (MSE) for fixed-point FFT units," Proc. IEEE Int. Symp. Circuits Syst. (2011) 1732 (DOI: 10.1109/ISCAS.2011.5937917).
[26] O. Sarbishei and K. Radecka: "On the fixed-point accuracy analysis and optimization of FFT units with CORDIC multipliers," Proc. IEEE Symp. Comput. Arithmetic (ARITH) (2011) 62 (DOI: 10.1109/ARITH.2011.17).
[27] W.-H. Chang and T. Q. Nguyen: "On the fixed-point accuracy analysis of FFT algorithms," IEEE Trans. Signal Process. **56** (2008) 4673 (DOI: 10.1109/TSP.2008.924637).
[28] C.-Y. Wang, *et al.*: "Hybrid word length optimization methods of pipelined FFT processors," IEEE Trans. Comput. **56** (2007) 1105 (DOI: 10.1109/TC.2007.1059).
[29] C. Yang, *et al.*: "New quantization error assessment methodology for fixed-point pipeline FFT processor design," IEEE System-on-Chip Conference (SOCC) (2014) 299 (DOI: 10.1109/SOCC.2014.6948944).
[30] A. V. Oppenheim and C. J. Weinstein: "Effects of finite register length in digital filtering and the fast Fourier transform," Proc. IEEE **60** (1972) 957 (DOI: 10.1109/PROC.1972.8820).
[31] M. Ayinala, *et al.*: "Pipelined parallel FFT architectures via folding transformation," IEEE Trans. Very Large Scale Integr. (VLSI) Syst. **20** (2012) 1068 (DOI: 10.1109/TVLSI.2011.2147338).