74-dBc SFDR 71-MHz Four-Stage Pipeline ROM-Less DDFS Using Factorized Second-Order Parabolic Equations

Chua-Chin Wang, Pang-Yen Lou, Tsung-Yi Tsai, and Hsiang-Yu Shih

***Abstract*— In this brief, a four-stage pipeline read only memory (ROM)-less direct digital frequency synthesizer (DDFS) with equal division interpolation is proposed. To attain higher spurious-free dynamic range (SFDR) and faster clock rate, the hardware cost and delay using different segments with various interpolation equations are analyzed systematically to explore the optimal solution. The second-order parabolic equations with proper coefficients and factorized operation orders based on optimized hardware cost and delay are finally utilized to enhance SFDR. The proposed design is demonstrated by the physical implementation using the TSMC 0.18-μm CMOS technology cell library and on-silicon measurements, where the maximum SFDR is 74 dBc, 0.018-mW/MHz power dissipation, and the maximal clock frequency is 71.9 MHz.**

***Index Terms*— Direct digital frequency synthesizer (DDFS), frequency synthesizer, parabolic polynomial interpolation, pipeline structure, spurious-free dynamic range (SFDR).**

I. Introduction

Direct digital frequency synthesizer (DDFS) is well known to adopt mathematical manipulations and directly synthesize the sinusoidal output, where the implementation can be realized by either the memory-based phase-to-sine mapping or an algorithm-based phase-to-sine mapping (or called conversion). Since there is no need of any feedback loop and voltage-controlled oscillator (VCO), DDFSs are able to achieve fast frequency switching and the wide output range more easily than the phase-locked loop (PLL)-based solutions.

Many studies have also reported read only memory (ROM)-less DDFS designs (see [1]–[3]), where different algorithms were used instead of ROM tables to realize the phase-to-sine mapper (PSM). A typical solution was proposed by Sodagar and Lahihi [4] using the second-order parabolic approximation. However, to reduce the conversion error, a scaling table and an error correction table (or generator) are needed. It not only deteriorates the speed but also reduces the resolution of the output word length. In addition, any polynomial whose order is higher than three was found impractical to be implemented if considering the feasibility of hardware realization [3]. Besides, the spurious-free dynamic range (SFDR) of sinusoidal output will be limited. Recently, several ROM-less DDFSs [5]–[8] have been developed to realize the PSM, i.e., the critical path of DDFS. We proposed a pipeline solution in [9], which is a two-page report to show the basic simulations without any theoretical analysis or implementation. In short, these DDFSs based on the second-order polynomials still cannot achieve high SFDR performance without sacrificing the speed.

To increase SFDR and reduce power dissipation, the second-order polynomial with equal division in a pipeline architecture is proposed and realized in this brief. Most important of all, different orders of mathematical operators of the second-order polynomial are fully analyzed to find out the optimal SFDR, output frequency, and power consumption.

Manuscript received December 27, 2018; revised April 25, 2019 and June 15, 2019; accepted June 17, 2019. This work was supported by the Ministry of Science and Technology under Grants MOST 106-2221-E-110-058, MOST 108-2218-E-110-002, and MOST 107-2218-E-110-016- *(Corresponding author: Chua-Chin Wang.)*

The authors are with the Department of Electrical Engineering, National Sun Yat-sen University, Kaohsiung 80424, Taiwan (e-mail: ccwang@ee.nsysu.edu.tw).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TVLSI.2019.2925574

II. High-SFDR DDFS Design Using Factorized Second-Order Parabolic Equations

Apparently, the sine wave is governed by the equation $f(x) = \sin(x)$. Theoretically, synthesizing a full sinusoidal wave $(0 \sim 2 \cdot \pi)$ can be achieved by synthesizing one quarter of the full cycle, namely $(0 \sim (1/2) \cdot \pi)$. DDFS means to take advantage of digital logic and algorithms to generate a waveform close to the real sinusoidal wave. Thus, the interpolation approach using the low-order polynomials (or called equations) is widely adopted to speed up the sine wave generation.

A. Analysis of Interpolation Schemes for DDFS

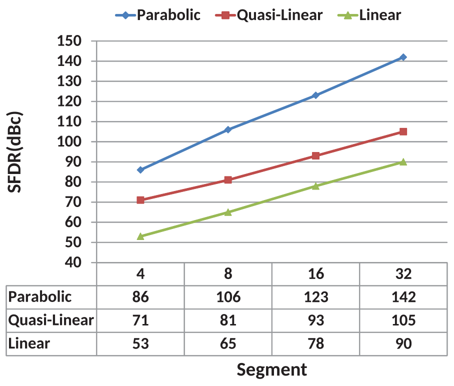
Three common interpolation schemes using low-order polynomials are linear interpolation (first-order polynomials), quasi-linear interpolation (combination of first-order and second-order polynomials), and parabolic interpolation (second-order polynomials) [10]. The linear interpolation apparently has the edge of simplicity with poor accuracy. By contrast, the parabolic interpolation is taking advantage of the similarity between $\sin(x)$ and the parabolic function. However, the price to pay is high computation complexity. As for the quasi-linear interpolation, the synthesis of the sine wave close to the origin is based on linear polynomials, and the synthesis of the region close to $(1/2)\pi$ is based on the parabolic polynomials. However, this scheme needs more sophisticated logic control and region division. The features of the mentioned three interpolation schemes are summarized as follows.

1) *Linear Interpolation:* First-order equations are used to attain lower complexity and high speed but result in the lowest SFDR as well.
2) *Quasi-Linear Interpolation:* Utilizing the second-order equation to minimize the error caused by the first-order equations, where the best partition of these two types of equations was found at $(\pi/8)$ [3], [8].
3) *Parabolic Interpolation:* Only the second-order equations are used to approximate the entire sine wave, which has the better SFDR and moderate complexity.

Notably, although more segments are utilized and synthesized, which will result in higher accuracy, the design complexity will increase, and the speed will be reduced. To explore the scenarios that are given different interpolation approaches and different segmentations, SFDR and error simulations by MATLAB are summarized in Table I. The SFDR and maximum error results of parabolic interpolation with eight segments are found to be better than those of quasi-linear interpolation with 32 segments. Besides, if the design complexity, area overhead, and switching speed are taken into consideration, the parabolic interpolation with eight segments seems to be a better option than others.

TABLE I
PERFORMANCE GIVEN DIFFERENT SEGMENTATION AND INTERPOLATION

	Parabolic interpolation		Quasi-linear interpolation		Linear interpolation	
#Seg.	Max. Error	SFDR	Max. Error	SFDR	Max. Error	SFDR
4	4.97×10^{-4}	86	1.28×10^{-3}	71	6.29×10^{-3}	53
8	$\mathbf{6.30 \times 10^{-5}}$	106	4.67×10^{-4}	81	1.59×10^{-3}	65
16	7.92×10^{-6}	123	1.35×10^{-4}	93	4.01×10^{-4}	78
32	9.97×10^{-7}	142	3.61×10^{-4}	105	1.00×10^{-4}	90



	4	8	16	32
Parabolic	86	106	123	142
Quasi-Linear	71	81	93	105
Linear	53	65	78	90

Fig. 1. SFDR of three interpolation methods by MATLAB simulations.

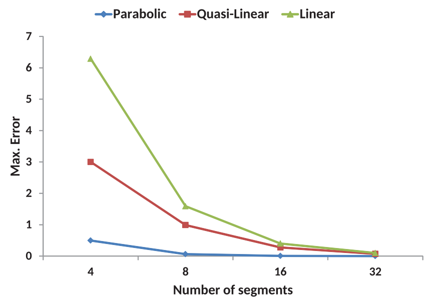


Fig. 2. Maximum error of three interpolation methods by MATLAB simulations.

When it comes to the performance priority, SFDR is well recognized to be more important than the maximum error and many others. Therefore, all the SFDR entries in Table I are plotted in Fig. 1 to highlight the difference among these selections. Similarly, the maximal errors are also shown in Fig. 2.

Referring to Table I, the parabolic interpolation method is apparently better than the other two methods when it comes to the consideration of SFDR. Notably, although 32-segment partition has the minimal error distribution, 8-segment partition with over 100 dBc SFDR is much more easy to be realized on silicon. Thus, the parabolic interpolation with eight-segment partition is chosen to be realized in this brief.

B. Operation Order and Selection of Parabolic Polynomials

As mentioned in Section II-A, the complexity can be reduced by selecting proper second-order parabolic equations, provided that the parabolic interpolation scheme is used. The most popular second-order polynomials are as follows:

$$f_1(x) = ax^2 + bx + c \quad (1)$$
$$f_2(x) = ax^2 + c \quad (2)$$
$$f_3(x) = a(x+b)^2 + c \quad (3)$$

TABLE II
DIFFERENT PARABOLIC EQUATION COMPARISONS

	Eqn. (1)	Eqn. (2)	Eqn. (3)	Eqn. (4)
Eqn.	ax^2+bx+c	ax^2+c	$a(x+b)^2+c$	$(ax+b)x+c$
Major Hardware Blocks	24-bit ADD 25-bit ADD 14×14 MUL 11×14 MUL	24-bit ADD 14×14 MUL	17-bit ADD 26-bit ADD 17×17 MUL	11-bit ADD 24-bit ADD 11×14 MUL
SFDR	Eqn.(1) $\approx$ Eqn.(3) $\approx$ Eqn.(4) $\gg$ Eqn.(2)			
Complexity	Eqn.(1) $>$ Eqn.(3) $>$ Eqn.(4) $>$ Eqn.(2)			

*simulation control factor : 32-bit FCW and 24-bit output resolution
*ADD=adder, MUL=multiplexer

$$f_4(x) = (ax+b)x + c. \quad (4)$$

Equation (1) is the general form of the parabolic equation. The other three equations are either simplified form or factorized form. Apparently, the implementation of (1) needs at least one multiplier and one squarer, where the squarer is basically a multiplier, besides two adder/subtractors. Although the implementation is straightforward, the hardware complexity is very high. By the perspective for SFDR, the missing coefficient "b" in (2) makes a single coefficient, "a," very hard to fit both curvature and the slope at the same time, which is why it attains the worst SFDR. Although the hardware complexity for (3) and (4) looks similar, the factorized $(ax+b)x+c$ has a very important advantage, which is that the truncation of ax will reduce the word length of later operations. By contrast, (3) will execute $(x+b)$ first so that word length of later operations will be longer.

To justify the theoretical analysis of the hardware complexity versus SFDR, 32-bit frequency control word (FCW) input and a 24-bit resolution output DDFS designs using the above-mentioned four parabolic equations are simulated by MATLAB, as tabulated in Table II. As expected, (2) attains the least complexity. Equation (1) is the most complicated one, because one more multiplexer is needed. Equation (4) is proven to attain the advantage of hardware complexity. Therefore, DDFS implemented using (4) is selected based on the simulation outcome.

C. Parabolic Polynomial Interpolation Derivation

A quadrant of sinusoid is partitioned into i segments, where every segment is approximated by the second-order parabolic equation, and each segment has m sampling points, as shown in the following equation:

$$y_{ij}(x_{ij}) = (a_i x_{ij} + b_i)x_{ij} + c_i, \quad i = 1 \sim 8, \;\; j = 1 \sim m. \quad (5)$$

According to the calculation method in [8], the least square method is used to attain the coefficients, and we differentiate (5) to get the following equation:

$$y'_{ij}(x) = 2a_i x_{ij} + b_i, \quad i = 1 \sim 8, \;\; j = 1 \sim m. \quad (6)$$

Equation (6) can be re-organized into a matrix expression as

$$\begin{bmatrix} y'_{i1} \\ y'_{i2} \\ y'_{i3} \\ \vdots \\ y'_{im} \end{bmatrix} = \begin{bmatrix} x_{i1} & 1 \\ x_{i2} & 1 \\ x_{i3} & 1 \\ \vdots & \\ x_{im} & 1 \end{bmatrix} \begin{bmatrix} 2a_i \\ b_i \end{bmatrix} \quad (7)$$

which is used to derive the least squares approximate solutions $[a_i, b_i]$ as shown in

$$\begin{bmatrix} x_{i1} & 1 \\ x_{i2} & 1 \\ x_{i3} & 1 \\ \vdots & \\ x_{im} & 1 \end{bmatrix}^T \begin{bmatrix} y'_{i1} \\ y'_{i2} \\ y'_{i3} \\ \vdots \\ y'_{im} \end{bmatrix} = \begin{bmatrix} x_{i1} & 1 \\ x_{i2} & 1 \\ x_{i3} & 1 \\ \vdots & \\ x_{im} & 1 \end{bmatrix}^T \begin{bmatrix} x_{i1} & 1 \\ x_{i2} & 1 \\ x_{i3} & 1 \\ \vdots & \\ x_{im} & 1 \end{bmatrix} \begin{bmatrix} 2a_i \\ b_i \end{bmatrix}$$

$$\begin{bmatrix} 2a_i \\ b_i \end{bmatrix} = \left(\begin{bmatrix} x_{i1} & 1 \\ x_{i2} & 1 \\ x_{i3} & 1 \\ \vdots & \\ x_{im} & 1 \end{bmatrix}^T \begin{bmatrix} x_{i1} & 1 \\ x_{i2} & 1 \\ x_{i3} & 1 \\ \vdots & \\ x_{im} & 1 \end{bmatrix} \right)^{-1} \begin{bmatrix} x_{i1} & 1 \\ x_{i2} & 1 \\ x_{i3} & 1 \\ \vdots & \\ x_{im} & 1 \end{bmatrix}^T \begin{bmatrix} y'_{i1} \\ y'_{i2} \\ y'_{i3} \\ \vdots \\ y'_{im} \end{bmatrix}. \quad (8)$$

Certainly, (5) can also be reorganized, which is used to find the least squares approximate solutions of $[c_i]$, as shown in the following equation:

$$\begin{bmatrix} 1 \\ c_i \end{bmatrix} = \left(\begin{bmatrix} (a_i x_{i1} + b_i) x_{i1} & 1 \\ (a_i x_{i2} + b_i) x_{i2} & 1 \\ (a_i x_{i3} + b_i) x_{i3} & 1 \\ \vdots & \\ (a_i x_{im} + b_i) x_{im} & 1 \end{bmatrix}^T \begin{bmatrix} (a_i x_{i1} + b_i) x_{i1} & 1 \\ (a_i x_{i2} + b_i) x_{i2} & 1 \\ (a_i x_{i3} + b_i) x_{i3} & 1 \\ \vdots & \\ (a_i x_{im} + b_i) x_{im} & 1 \end{bmatrix} \right)^{-1}$$

$$\times \begin{bmatrix} (a_i x_{i1} + b_i) x_{i1} & 1 \\ (a_i x_{i2} + b_i) x_{i2} & 1 \\ (a_i x_{i3} + b_i) x_{i3} & 1 \\ \vdots & \\ (a_i x_{im} + b_i) x_{im} & 1 \end{bmatrix}^T \begin{bmatrix} y_{i1} \\ y_{i2} \\ y_{i3} \\ \vdots \\ y_{im} \end{bmatrix}. \quad (9)$$

D. Stage of Pipelining

Pipelining is a well-known method to enhance clock rate and throughput. However, how many stages of pipelining are needed in the eight-segment DDFS design using the second-order parabolic polynomial, namely (5), is the next issue to be resolved.

1) *Two Stages:* This is the simplest pipelining structure, where one register is used after the 32-bit FCW counting, and the other is added before the generation of the 24-bit output. The clock rate is 62 MHz.
2) *Three Stages:* The multiplier is certainly the critical path of the entire data flow. An extra register stage is then added before the multiplier such that the clock rate is increased to 83 MHz at the expense of 7% hardware cost.
3) *Four Stages:* To further increase the clock rate, another register is proposed to be added after the multiplier. The clock rate is raised up to 100 MHz, which is 61% higher than that of the two-stage structure. However, the hardware cost is increased by 12% thereof.

Although higher order of pipelining is feasible, the increase of clock rate becomes very obscure and the hardware cost turns into quite significant. Therefore, a four-stage pipeline structure is considered as the optimal solution to carry out the DDFS design, as shown in Fig. 3.

Notably, the first two most significant bit (MSB) of phase accumulator output are used as the control bits to generate all the four quarters of a full sine wave, where MSB1 is for the up and down of the sinusoidal outputs, and MSB2 is for the left and right of the sinusoidal outputs. Because the proposed design is realized by an equal eight-segment partition, MSB3, MSB4, and MSB5 are used to control the coefficients a_i, b_i, and c_i, respectively. For example, when (MSB3, MSB4, MSB5) is (0,0,0), the coefficients a_1, b_1, and c_1 are selected. If (MSB3, MSB4, MSB5) is (0, 1, 0), a_3, b_3, and c_3 are selected.

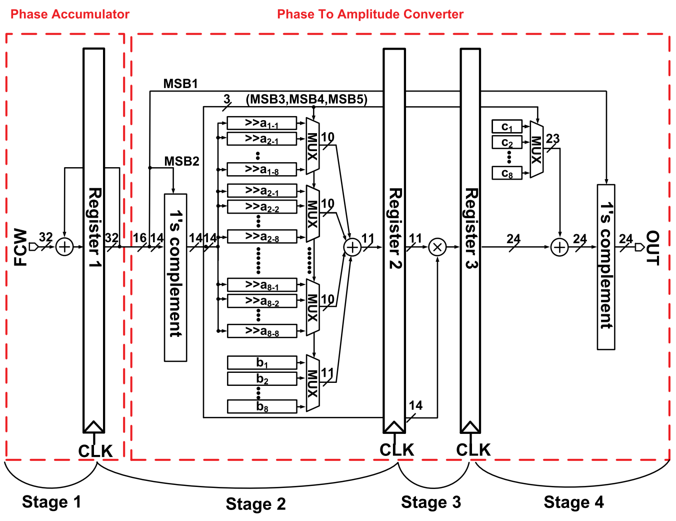


Fig. 3. Block diagram of the proposed DDFS.

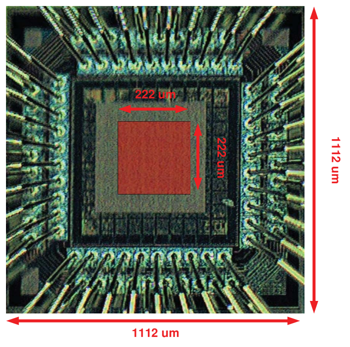


Fig. 4. Die photograph of the proposed DDFS.

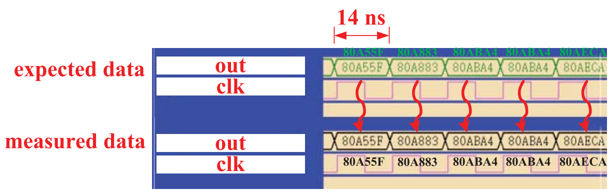
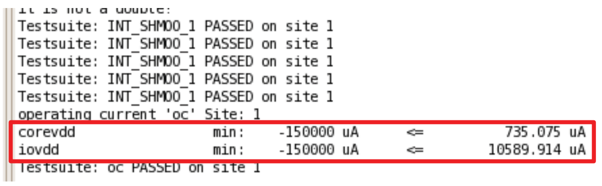


Fig. 5. Functional test.



```
It is not a double:
Testsuite: INT_SHMOO_1 PASSED on site 1
Testsuite: INT_SHMOO_1 PASSED on site 1
Testsuite: INT_SHMOO_1 PASSED on site 1
Testsuite: INT_SHMOO_1 PASSED on site 1
Testsuite: INT_SHMOO_1 PASSED on site 1
operating current 'oc' Site: 1
corevdd                  min:    -150000 uA      <=          735.075 uA
iovdd                    min:    -150000 uA      <=        10589.914 uA
Testsuite: oc PASSED on site 1
```

Fig. 6. Current measurement.

III. Implementation and Measurement

This brief is realized using the TSMC 0.18-μm mixed-signal CMOS process cell library. The core area is only 0.222×0.222 mm^2,

TABLE III
PERFORMANCE COMPARISON OF DDFS

	[8]	[5]	[6]	[7]	[11]	[12]	This work
year	ISIC 2011	EMICC 2013	ICEE 2014	IFCS 2016	EuMIC 2017	ICMDCS 2017	2018
Process (μm)	0.35	0.13	0.35	0.18	0.065	0.18	0.18
FCW	32 bits	32 bits	9 bits	17 bits	24 bits	16 bits	32 bits
Output Resolution	24 bits	12 bits	8 bits	10 bits	10 bits	16 bits	24 bits
Power (mW / MHz)	0.31	1.07	0.186	0.027	0.036	0.024	0.018
Normalized Area ($*10\text{mm}^2$)	11.85	18.93	1.20	0.52	25.44	1.28	1.52
SFDR (dBc)	68	60	55	52.47	56.5	49.1	74
Verification	Sim.	Meas.	Sim.	Meas.	Meas.	Sim.	Meas.
Clock(MHz)	50	650	1000	100	2000	1000	71.9
Max. Output Freq. (MHz)	25	155	240	24	1000	500	17
FOM†	5.85	5.87	7.25	7.80	7.47	8.61	8.19

*Sim.=Simulation, Meas.=Measurement

$$^{\dagger}\text{FOM} = \log_{10}\left(\frac{\text{SFDR}*\text{FCW}*\text{Output resolution}*\text{Max.Clock}}{\text{Power}*\text{Normalized Area}}\right)$$

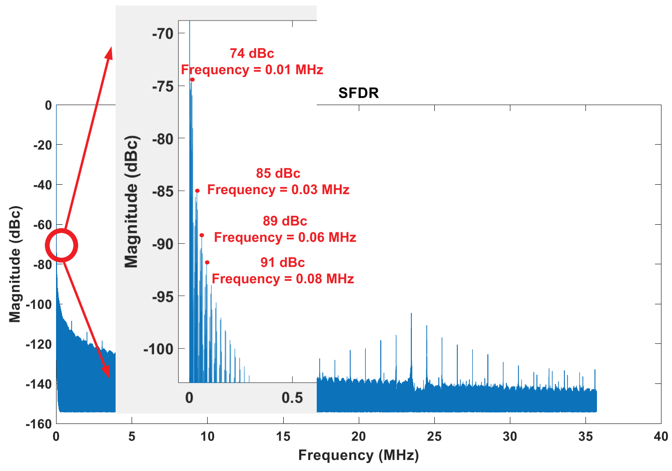


Fig. 7. SFDR measurement.

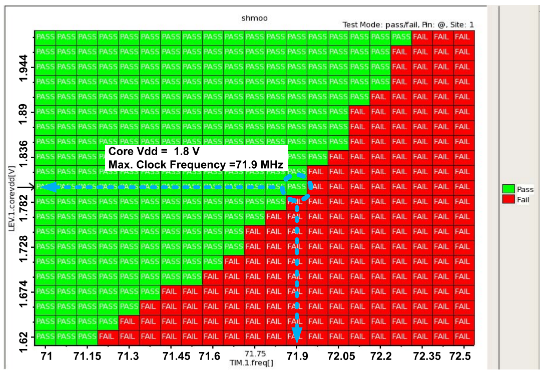


Fig. 8. Shmoo plot.

and the overall chip size is 1.112×1.112 mm^2. The die photograph is shown in Fig. 4.

The prototype chip of the proposed DDFS is tested by ADVANTEST V93000 PS1600 provided by Taiwan Semiconductor Research Institute (TSRI), Taiwan. The testing procedure consists of continuity check, standby current measurement, functional test,

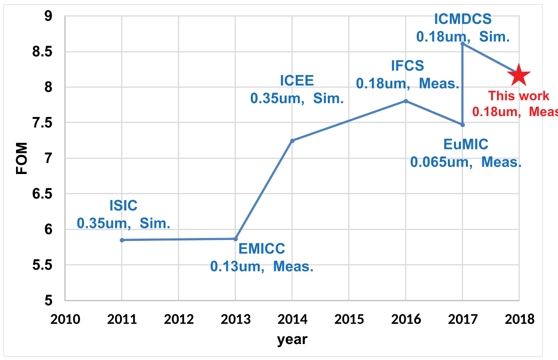


Fig. 9. Roadmap of the ROM-less DDFS designs.

operation current measurement, and shmoo plot. Fig. 5 shows a snapshot of functional test. The highest clock rate is 71.9 MHz at 1.8-V system voltage. The power is found as shown in Fig. 6, where the core current is 735.05 μA, and the PAD current is 10589.914 μA. This concludes that the power of the proposed DDFS is 1.8 V $\times$ 735.05 μA = 1.3231 mW at 71.9 MHz.

As for the most important measure of DDFS, the SFDR measurement is shown in Fig. 7, where 74 dBc is verified. Fig. 8 shows the shmoo plot, where the clock rate is 71.9 MHz at 1.8-V VDD.

Table III shows the performance comparison with prior ROM-less DDFS designs. Not only our design attains the highest SFDR but also has the second best FOM. Although [12] has the best FOM, it is only based on the simulations. Besides, the SFDR of [12] is only 49.1 dBc. Our major impact is the low power and low complexity contributed by the factorized operation of the parabolic polynomial, as shown by the roadmap in Fig. 9.

IV. CONCLUSION

This brief demonstrated a pipeline ROM-less DDFS with optimally selected segments and factorized second-order parabolic interpolation equations. By physical measurement on silicon, this brief attained the significant SFDR 74 (dBc) and achieved the second best FOM, given FCW (32 bits) and output resolution (24 bits).

Acknowledgment

The authors would like to express their deepest gratefulness to the Taiwan Semiconductor Research Institute (TSRI) of Nation Applied Research Laboratories (NARL), Taiwan.

References

[1] C.-C. Wang, Y.-L. Tseng, H.-C. She, C.-C. Li, and R. Hu, "A 13-Bit resolution ROM-less direct digital frequency synthesizer based on a trigonometric quadruple angle formula," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 13, no. 9, pp. 1096–1098, Sep. 2005.

[2] Y. Song and B. Kim, "A 250 MHz direct digital frequency synthesizer with delta-sigma noise shaping," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, vol. 1, pp. 472–509, Feb. 2003.

[3] C.-C. Wang, C.-H. Hsu, C.-C. Lee, and J.-M. Huang, "A ROM-less DDFS based on a parabolic polynomial interpolation method with an offset," *J. Signal Process. Syst.*, vol. 64, no. 3, pp. 351–359, May 2010.

[4] A. M. Sodagar and G. R. Lahihi, "A novel architecture for ROM-less sine-output direct digital, frequency synthesizers by using the 2^{nd}-order parabolic approximation," in *Proc. IEEE/EIA Int. Freq. Control Symp. Exhib.*, Jun. 2000, pp. 284–289.

[5] J. Cali, X. Geng, F. Zhao, M. Pukish, F. Dai, and A. Aklian, "A 650 MHz DDFS for stretch processing radar in 130 nm BiCMOS process," in *Proc. Eur. Microw. Integr. Circuit Conf. (EMICC)*, Oct. 2013, pp. 33–36.

[6] M. Padash, S. Toofan, and M. Yargholi, "A 9-bit, 1-giga samples per second sine and cosine direct digital frequency synthesizer," in *Proc. 22nd Iranian Conf. Elect. Eng. (ICEE)*, May 2014, pp. 438–442.

[7] P. R. B. de Carvalho, J. A. A. Palacio, and W. Van Noije, "Area optimized CORDIC-based numerically controlled oscillator for electrical bio-impedance spectroscopy," in *Proc. IEEE Int. Freq. Control Symp. (IFCS)*, May 2016, pp. 1–6.

[8] C.-H. Hsu, Y.-C. Chen, and C.-C. Wang, "ROM-less DDFS using non-equal division parabolic polynomial interpolation method," in *Proc. Int. Symp. Integr. Circuits (ISIC)*, Dec. 2011, pp. 59–62.

[9] T.-Y. Tsai, H.-Y. Shih, and C.-C. Wang, "A pipeline ROM-less DDFS using equal-division interpolation," in *Proc. Int. SoC Design Conf. (ISOCC)*, Nov. 2017, pp. 19–20.

[10] X. Geng, F. F. Dai, J. D. Irwin, and R. C. Jaeger, "An 11-bit 8.6 GHz direct digital synthesizer MMIC with 10-bit segmented sine-weighted DAC," *IEEE J. Solid-State Circuits*, vol. 45, no. 2, pp. 300–313, Feb. 2010.

[11] A. M. Alonso, X. Yuan, M. Miyahara, and A. Matsuzawa, "A 2 GS/s 118 mW digital-mapping direct digital frequency synthesizer in 65 nm CMOS," in *Proc. 12th Eur. Microw. Integr. Circuits Conf. (EuMIC)*, Oct. 2017, pp. 228–231.

[12] R. Suryavanshi, S. Sridevi, and B. Amrutur, "A comparative study of direct digital frequency synthesizer architectures in 180 nm CMOS," in *Proc. Int. Conf. Microelectron. Devices, Circuits Syst. (ICMDCS)*, Aug. 2017, pp. 1–5.