High-Throughput Signal Component Separator for Asymmetric Multi-Level Outphasing Power Amplifiers

Yan Li, Zhipeng Li, Oguzhan Uyar, Yehuda Avniel, Alexandre Megretski, and Vladimir Stojanović

***Abstract*—This paper presents an energy-efficient high-throughput and high-precision signal component separator (SCS) chip design for the asymmetric-multilevel-outphasing (AMO) power amplifier. It uses a fixed-point piece-wise linear functional approximation developed to improve the hardware efficiency of the outphasing signal processing functions. The chip is fabricated in 45 nm SOI CMOS process and the SCS consumes an active area of 1.5 mm^2. The new algorithm enables the SCS to run at a throughput of 3.4 GSamples/s producing the phases with 12-bit accuracy. Compared to traditional low-throughput AMO SCS implementations, at 0.8 GSamples/s this design improves the area efficiency by 25× and the energy-efficiency by 2×. This fastest high-precision SCS to date enables a new class of high-throughput mm-wave and base station transmitters that can operate at high area, energy and spectral efficiency.**

***Index Terms*—Application specific integrated circuits (ASIC), asymmetric multi-level outphasing (AMO) power amplifier, baseband, energy efficiency, linear amplification by nonlinear component (LINC), Signal component separator (SCS), throughput.**

I. Introduction

HIGH-THROUGHPUT wireless communication systems working at the millimeter-wave (mm-wave) frequency range from 60 GHz to 90 GHz [1]–[7] have recently become the focus of research and development activity. The availability of large chunks of bandwidth and maturity of CMOS process technology provide the opportunity to address several large markets with bandwidth-demanding communication applications. Meanwhile, these mm-wave applications place great challenges on the transceiver design, due to factors such as power-amplifier (PA) efficiency and linearity, high wireless channel loss and multipath, increasing parasitics for passive components, limited amplifier gain etc. Even in cellular base stations, the drive toward flexible, multi-standard radio chips, increases the need for high-precision, high-throughput and energy-efficient backend processing. The desire to best leverage the available spectrum for these high-throughput applications, creates the demand for high-efficiency and high-linearity PAs. While these conflicting PA design requirements have been satisfied in the past at low system throughputs by designing smart digital back-ends, the multi-GSamples/s throughput required in new applications puts a significant challenge on digital baseband system design to perform the necessary modulation and predistortion operations at negligible power overhead.

This desire for high-throughput energy-efficient digital baseband becomes especially prominent for the outphasing PAs designed to improve the efficiency while satisfying the high-linearity requirements for higher-order signal constellations. At low throughputs (10–100 MSamples/s), the outphasing PAs would rely on complex digital signal processing to generate the outphasing vectors and make it possible to use simple, high-efficiency switching PAs on each path. Examples of the outphasing PAs include the linear-amplification-by-nonlinear-component (LINC) PA proposed by Cox in [8], and its more recent modification: the asymmetric-multilevel-outphasing (AMO) PA [9]–[11]. At high (multi-GSamples/s) throughputs, however, a radical redesign of the signal component separator (SCS) digital signal processing implementations is needed to prevent degradation in net power efficiency due to significant increase of digital baseband power consumption.

The conventional LINC SCS has been traditionally implemented both in analog and digital designs [12]–[14]. The analog versions of SCS are obviously not suitable for high-speed and high-precision applications, so we only consider the digital SCS implementations. The SCS decomposes the original sample signal to two signals as required by the LINC/AMO, and the decomposition involves the computations of several nonlinear functions. For digitally implemented SCS, a look-up-table (LUT) is the most common way to realize the nonlinear functions. Considering that the past signal separators mainly work below 100 MSamples/s with low to medium precision, LUT indeed is the simplest and most energy-efficient approach. Even for the recent AMO architecture, LUT is still a preferable choice for operations under 100 MSamples/s [15]. However, the traditional LUT-based function map quickly becomes infeasible when the throughput and precision requirements go up to multi-GSamples/s and more than 10-bit range. The LUT size becomes prohibitively large for on-chip implementations and gives the penalty in both area and speed. Besides, the number of LUTs used in the AMO SCS is significantly larger than in the LINC SCS, so the LUT solutions that can barely work for LINC render AMO implementations infeasible. On the other hand, at these high throughputs a direct nonlinear function synthesis through iterative algorithms like CORDIC [16] or

Manuscript received September 12, 2012; revised October 08, 2012; accepted October 17, 2012. Date of current version January 24, 2013. This paper was approved by Associate Editor Ichiro Fujimori.

The authors are with the Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA 02139 USA.

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/JSSC.2012.2229071

TABLE I
LINC AND AMO SCS EQUATIONS

LINC Equations	AMO Equations
$A=\sqrt{I^2+Q^2}, \theta=\arctan(\frac{Q}{I})$ (linc1)	$A=\sqrt{I^2+Q^2}, \theta=\arctan(\frac{Q}{I})$ (amo1)
$\alpha=\arccos(\frac{A}{2a})$ (linc2)	$\alpha_1=\arccos(\frac{a_1^2+A^2-a_2^2}{2Aa_1}), \alpha_2=\arccos(\frac{a_2^2+A^2-a_1^2}{2Aa_2})$ (amo2)
$\varphi_1=\theta+\alpha, \varphi_2=\theta-\alpha$ (linc3)	$\varphi_1=\theta+\alpha_1, \varphi_2=\theta-\alpha_2$ (amo3)
	$f(\varphi_1)=\frac{1}{1+tan(\varphi_1)}, f(\varphi_2)=\frac{1}{1+tan(\varphi_2)}$ (amo4)

nonlinear filters [17] proves to be more area compact but with prohibitive power footprint for the overall power efficiency of the PA.

In this paper, we present the function synthesis algorithms and a corresponding chip implementation, designed using an alternative approach to compute the nonlinear functions, which is both more area and energy-efficient than state-of-the-art methods like LUTs, CORDIC or nonlinear filters. The chip results demonstrate an AMO SCS working at 3.4 GSamples/s with 12-bit accuracy and over 2× energy savings and 25× area savings compared to traditional AMO SCS implementation. The new approach is based on the piece-wise linear (PWL) approximation of a nonlinear function. The approximation consists of the computations of LUT, add, and multiply. In order to minimize the computational cost while maintaining high accuracy and throughput, we propose a novel algorithm to find the fixed-point representation of the approximation. The idea of the fixed-point version of the approximation is to use as few operations as possible and minimize the number of input bits to all the operations so as to achieve high throughput. With these considerations, we are able to achieve a fixed-point representation of typical LINC or AMO nonlinear functions, which consists of one small LUT, one adder and one multiplier. The hardware architecture derived from this special algorithm achieves a nice balance among area, energy-efficiency, throughput and computation accuracy, which will be presented in details in the rest of the paper.

The paper is organized as follows. In Section II, we present the basic principles of LINC and AMO PA architectures and their corresponding SCSs. In Section III, we introduce the proposed approximation algorithm and an example to illustrate its derivations and advantages. In Section IV, we present the chip design of the digital baseband system and the microarchitecture of each block, followed by the chip measurement results. We conclude the paper in Section V.

II. SYSTEM OVERVIEW

Both LINC and AMO PAs are outphasing PA architectures and their digital basebands perform similar computations. The LINC PA architecture is proposed by Cox in [8] with the motivation to relieve the ever existing trade-off between the power efficiency and linearity performances of the PA. By decomposing the transmitted signal to two constant-amplitude signals, high-efficiency PAs can be used to amplify the two decomposed signals without sacrificing the linearity. The AMO PA architecture, proposed in [9]–[11] improves the average power efficiency further by allowing the two PAs switch among a discrete set of power supplies rather than fixing on a single supply level.

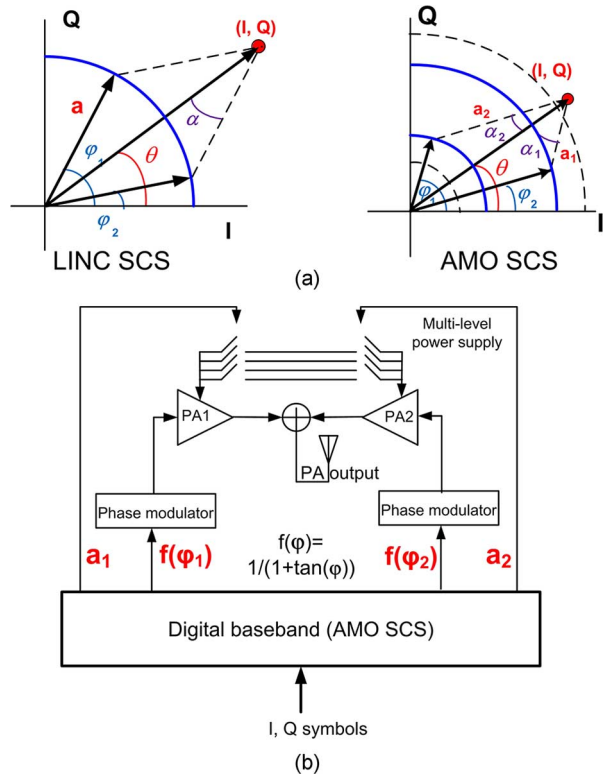


Fig. 1. (a) LINC, AMO SCS. (b) AMO PA system overview.

Fig. 1(a) shows the working schemes of LINC SCS and AMO SCS for an arbitrary IQ sample (I,Q). The SCS decomposes the (I,Q) to two signals with phases of φ_1, φ_2 and amplitudes of a_1, a_2, where for LINC $a_1 = a_2 = a$. The outphasing angles φ_1 and φ_2 for both architectures are derived from the equations summarized in Table I. In AMO equations, a_1, a_2 denote the power supplies of the two PAs respectively. a_1, a_2 are restricted to the set of $\mathcal{V} = \{V_1, V_2, V_3, V_4\}$, where $V_1 \leq V_2 \leq V_3 \leq V_4$ are the four levels of supply voltages. Equations in (amo4) of Table I are in the signal decomposition process simply due to the architecture requirement from the digital-to-RF-phase-converter (DRFPC) [18], which converts the digital outputs to RF modulated signals and takes a function of the phase $f(\varphi)$ as the input. Generally, computations in (amo4) depend on the type of the modulator and may be different than what we present here.

The typical low-throughput LINC SCS and recent AMO implementations [12]–[15], [19] usually involve the use of coordinate rotational digital computer (CORDIC) [16] and LUT map for the nonlinear functions in Table I [14], [20]. The maturity of the CORDIC algorithm and simplicity of the LUT

approach make themselves suitable for the LINC SCS applications whose throughput is below 100 MSamples/s and with low to medium resolution ($\leq$8 bits for example). However, the approaches become less attractive or even prohibitive for our target mm-wave wideband applications where the throughput is in the multi-GSamples/s range with high phase resolution ($\geq$10 bits for example). In the next section, we show our proposed solution: using fixed-point PWL approximations on the nonlinear functions which provides a balance among accuracy, power and area.

III. Proposed Piece-Wise Linear Approximation

A. Algorithm

The motivation for a new approach to the nonlinear function computation is simple: avoid and replace complex computations with simple and energy-efficient computations. For example, table look-up with LUTs of reasonable sizes, adders and multipliers are the favorable computations to perform. We also realize that all functions involved in the SCS computations are smooth in almost the whole input range. Hence, they are suitable to be approximated by functions with simple structured basis functions, such as polynomials, splines and etc. These considerations lead us to the PWL function approximation of the nonlinear functions.

Fig. 2(a) shows the general application of the PWL approximation to any smooth nonlinear function. The input x is divided into several intervals, where a linear function $y_i = a_i \times x + c_i$, $x \in [x_i, x_{i+1})$ is constructed in each interval to approximate the actual function value in that range. With this approximation, the computation of the nonlinear function only consists of the linear function computation in each interval (add and multiply), plus a relatively small LUT for the linear function parameters a_i, c_i in each interval. In terms of accuracy, for any function which has a continuous second-order derivative, the approximation error is bounded by the interval length, the second-order derivative and does not depend on higher-order derivatives, as shown in [21],

$$|error| \leq \frac{1}{8}(x_{i+1} - x_i)^2 \max_{x_i \leq x \leq x_{i+1}} |y''(x)|\,. \qquad (1)$$

Here, x_i, x_{i+1} are the boundaries of the i^{th} interval and y'' is the second-order derivative in x. We observe that the approximation error can be made arbitrarily small as we increase the number of approximation intervals. These initial examinations on the computational complexity and approximation accuracy of the piece-wise linear approximation make it an appealing alternative technique for the LINC and AMO SCS designs.

In order to benefit from the nice properties of the PWL approximation, we need to tailor it to be hardware-implementation friendly. Most importantly, all the arithmetic computations have to be converted to their fixed-point counterparts, and the question is whether the resulting fixed-point computations are able to operate at multi-GSamples/s throughputs with high accuracy. The most seemingly obvious solution is a direct quantization of the parameters in the floating-point representation of the approximation formula. However, this may not be an optimal solution if throughput is the major concern and bottleneck, because the operands of the add and multiply a_i, c_i are quantized to have

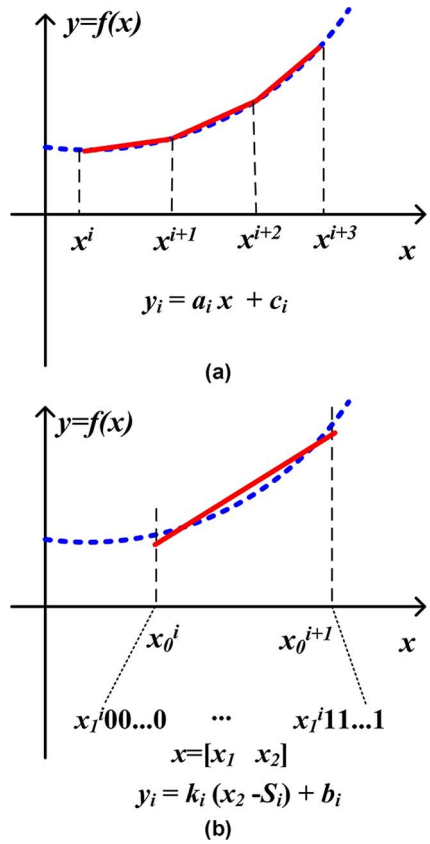


Fig. 2. (a) The general concept of PWL approximation. (b) Proposed fixed-point PWL approximation.

the same long bits as the output, and these long-bit arithmetics are likely to be in the critical timing path. Further optimization of the long multiplication would only add complexity to the design. In what follows, we present a modified formulation of the fixed-point PWL approximation and show its capability of running at a much higher throughput than the direct quantization version of the approximation.

The setup of our problem is to compute a nonlinear function of m-bit output with m-bit input $x \in [0, 1)$, using the PWL approximation. An m-bit input x can be decomposed to x_1 and x_2 as $x = [\ \underbrace{x_1}_{m_1-\text{MSB bit}}\ ,\ \underbrace{x_2}_{m_2-\text{LSB bit}}\ ]$, where $m = m_1 + m_2$. Naturally, x_1 divides the input range to 2^{m_1} intervals and it is the indexing number of those intervals. Fig. 2(b) shows an enlargement of the i^{th} interval of the approximation, where x_1 takes its i^{th} value, and x_2 takes 2^{m_2} values, ranging from 0 to $2^{m_2} - 1$. Under this setup, we have our proposed fixed-point scheme shown in (2).

$$y_i = \underbrace{b_i \cdot \mathbf{1}}_{m_1-\text{MSB bit}} + \underbrace{k_i(x_2 - S_i \cdot \mathbf{1})}_{m_2-\text{LSB bit}}, \quad i = 0, 1, \ldots 2^{m_1} - 1. \qquad (2)$$

Here, $y_i = [y([i,0]), y([i,1]), \cdots, y([i, N_2 - 1])]^{\mathrm{T}}$, $x_2 = (1/N)[0, 1, \cdots, N_2 - 1]^{\mathrm{T}}$, $\mathbf{1} = [1, 1, \cdots, 1]^{\mathrm{T}} \in \mathbb{R}^{N_2}$, $N_1 = 2^{m_1}$, $N_2 = 2^{m_2}$, $N = 2^m$, $m = m_1 + m_2$, $k_i, S_i, b_i \in \mathbb{R}$ and they are all fixed-point numbers.

The underlying idea of this formulation is to compute the m-bit output part by part. In the linear function of each interval,

we use the term b_i to represent the most significant m_1 bits of the function value, and the term $k_i \cdot (x_2 - S_i \cdot \mathbf{1})$ to achieve the lower-significant m_2 bits of accuracy. Then y_i is simply the concatenation of the two parts. The procedures to find the fixed-point representations of the three parameters k_i, S_i, b_i in (2) are described in the following steps.

Step 1: Obtain the Floating-Point Version of the PWL Approximation: The optimal real coefficients of the linear function in each interval in terms of l_2 norm can be found by least-square optimization (3), where the design variables are k_i^r and $b_i^r \in \mathbb{R}$. The superscripts denote that they are floating-point real numbers; x_2 and y_i are defined as in (2).

$$\min_{k_i^r, b_i^r} \|y_i - (k_i^r \cdot x_2 + b_i^r \cdot \mathbf{1})\|_2, \text{ for } i = 0, 1, 2, \ldots, N_1 - 1, \tag{3}$$

The approximation error bound in (1) shows that the error is proportional to $(x_{i+1} - x_i)^2$, which in the fixed-point input case, equals 2^{-2m_1}. Let $m_1 = \lceil m/2 \rceil$, then it is possible to realize the required output m-bit accuracy with only $2^{\lceil m/2 \rceil}$ intervals. Since the number of intervals determines the number of address bits of the LUT that stores the parameters of the linear function in each interval, this LUT ($2^{\lceil m/2 \rceil}$ entries) is considerably smaller than a direct map from input to output (2^m entries). The following steps determine the fixed-point parameter values, i.e., the content of the LUT.

Step 2: Obtain the Fixed-Point Value b_i: b_i can be achieved simply by quantizing the b_i^r to m_1-bit. As we mentioned before, the m-bit output is constructed part by part with b_i as the constant term in the i^{th} interval, representing the major part of the function value in that interval. As long as the functional value increment in each interval is less than 2^{-m_1}, that is, the functional derivative $|y'(x)| < 1$, it is enough to use the m_1-MSB of b_i to represent the m_1-MSB of the output.

Step 3: Obtain the Fixed-Point Value S_i: Since Step 2 yields a b_i with a maximum quantization error of 2^{-m_1}, to compensate for the accuracy loss of $b_i^r - b_i$, an extra parameter S_i^r is introduced such that $k_i^r S_i^r = b_i^r - b_i$. Its fixed-point counterpart S_i is derived as in (4)

$$S_i = \text{quantize}\left(\frac{(b_i^r - b_i)}{(k_i^r)}\right). \tag{4}$$

The number of bits of S_i is determined such that $k_i^r S_i$ has the accuracy of $m + 1$ bits. From our experience with the functions involved in the SCS design, S_i usually has the number of bits around or a few more (i.e. 2–4) bits than $m/2$, depending on the derivative k_i of the function in each interval.

Step 4: Obtain the Fixed-Point Value k_i: The slope of the function in the i^{th} interval k_i can also be obtained by simply quantizing its floating-point counterpart from the optimization procedure in Step 1. As shown in (2), the term $k_i(x_2 - S_i \cdot \mathbf{1})$ contributes to the second part of the output- the m_2 LSBs. Since $x_2 - S_i$ has an accuracy of at least m bits, k_i has to have at least m_2 bits to make the m_2 LSBs of the output.

The above procedure not only provides a way to obtain the three fixed-point parameters of the linear function in each interval, but also provides benefit in the high-throughput hardware micro-architecture design. Fig. 3(a) shows the micro-architecture of the approximation and (b) shows more clearly how

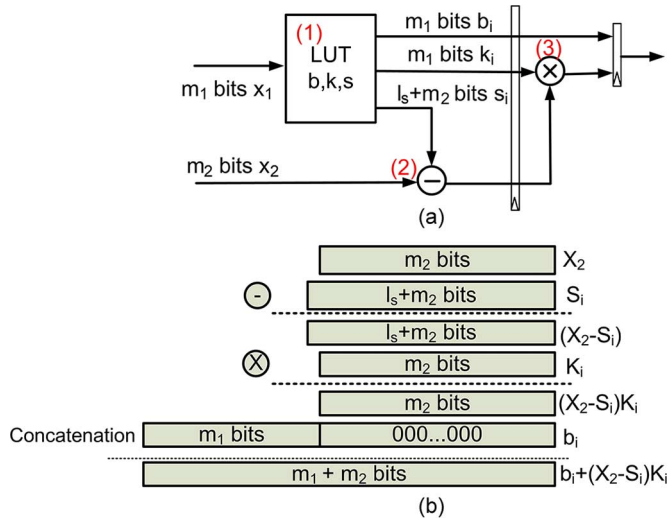


Fig. 3. (a) Micro-architecture of the PWL approximation. (b) Illustration of the computations in the PWL approximation.

the computations are carried out. There are essentially 3 arithmetic operations involved: LUT, one adder, and one multiplier. The LUT takes the m_1 MSBs of the input as the address and outputs the parameters b_i, k_i, S_i in the corresponding interval. Then the linear function computations follow accordingly. From Fig. 3(a), we notice that for all arithmetic computations, the operands have only m_1, m_2 or $l_s + m_2$ bits, but not m bits as input. As we discussed in Step 1, it is a good choice to set $m_1 = \lceil m/2 \rceil$, hence with operands of $m/2$ bits (roughly) in all computations, we are able to achieve the m-bit output.

This implies two important improvements in hardware efficiency: storage and throughput. For a direct LUT implemented function, if both the input and output have m bits, the storage required is $m \cdot 2^m$. With the proposed scheme, the storage is $(2m_2 + ls + m_1) \cdot 2^{m_1}$, which is approximately $1.5m \cdot 2^{m/2} \sim 2m \cdot 2^{m/2}$ assuming $m_1 = m_2 = m/2$ (when m is even) and l_s small ($\leq$4). A comparison on the storage usage between the direct LUT map and the fixed-point PWL approximation approach is illustrated in Table II, for practical range of m from 10 to 16. The last column of the table shows the ratio of LUT size from approximation versus the one from direct LUT map, which reflects the storage savings of 10–100$\times$ for the range of values of interest. The net area advantage of our approach versus the direct LUT will depend on the actual technology and throughput specifications, since these would dictate the type of the storage elements being used. For example, in high-throughput applications, register-based LUTs are needed while in lower throughput conditions, SRAM-based LUTs can be used. Under both types of LUT implementations, the additional area consumption brought by one adder and one multiplier is almost negligible compared to the LUT area. For example, in 45 nm SOI technology, the direct LUT implementation of a 16-bit in/out arccos function consumes an area of 19 mm^2 in register-based implementation and 0.7 mm^2 SRAM implementation. With the PWL approximation, area consumption reduces to 46200 μm^2 with register implementation and 9784 μm^2 with SRAM. The adder and multiplier consume roughly 1280 μm^2 in total, which is only a small

TABLE II
STORAGE COMPARISON EXAMPLES BETWEEN A DIRECT LUT MAP APPROACH AND FIXED-POINT PIECE-WISE LINEAR APPROXIMATION APPROACH

m	Direct LUT size L1 (bits)	Approx. LUT size L2 (bits)	Improvement ratio(L1/L2)
10	10×2^{10}	20×2^{5}	2^{4}
12	12×2^{12}	24×2^{6}	2^{5}
14	14×2^{14}	28×2^{7}	2^{6}
16	16×2^{16}	32×2^{8}	2^{7}

portion compared to the overall area consumption. Obviously, the PWL approximation has a large advantage in storage size and the advantage becomes more prominent as the input and output size increases. As for the throughput, because of the short operands and LUT address, the whole chain of operations: LUT, add and multiply can be easily pipelined into a few stages depending on the process and throughput requirement. For example, with a 45 nm SOI process, we use two pipeline stages: table lookup, adder in the first pipeline stage and multiply in the second pipeline stage, and this structure can sustain roughly a 2-GSamples/s throughput to compute a 15-bit input and output nonlinear function.

As a side note, an alternative way to write our formulation (2) is

$$y_i = k_i \cdot x_2 + (-k_i S_i \cdot \mathbf{1} + b_i \cdot \mathbf{1}) = k_i \cdot x_2 + c_i. \quad (5)$$

To compare the two formulations, we consider the following two aspects: storage size and arithmetic computation complexity. In terms of storage size, formulation (2) requires $(m_1+m_2+m_2+l_s)\cdot 2^{m_1} = (2m_2+m_1+l_s)\cdot 2^{m_1}$ bits while (5) requires $(m_1+m_2+m_2)\cdot 2^{m_1} = (2m_2+m_1)\cdot 2^{m_1}$ bits. Formulation (2) does require a little bit more storage of $l_s \cdot 2^{m_1}$ bits, however, it brings the advantage of shorter operands of the add operation. In terms of arithmetic operation complexity, formulation (2) requires an adder with $m_2 + l_s$ and m_2-bit operands, an multiplier with $m_2 + l_s$ and m_2-bit operands, while (5) requires an m-bit full adder and m_2-bit multiplier. As m gets large, the long adder in (5) may need further pipelining and complicates the design at high throughput. Furthermore, the optimization lets b_i represent the first m_1 bits while it chooses k_i and S_i in (2) so that $k_i(x_2 - S_i)$ exactly represent the rest of the m_2 bits, to avoid any overflow and an additional adder. Our design is more throughput rather than area-limited, therefore with the above considerations, we choose to use formulation (2) to achieve a higher throughput with more compact arithmetic hardware.

B. Piece-Wise-Linear Design Example

In this section, we show an example of computing a normalized 16-bit input, 16-bit output arccosine function $y = \arccos(x)/(2\pi)$ using the proposed PWL approximation approach. This function is one of the functions in the actual AMO SCS design.

First, we obtain a floating-point representation of the PWL approximation through the following least-square minimization:

$$\min_x \|Ax - \beta\|_2, \text{where}$$

$$A = \begin{bmatrix} 1, & 1, & \cdots, & 1 \\ \frac{0}{N^2}, & \frac{1}{N^2}, & \cdots, & \frac{N-1}{N^2} \end{bmatrix}^{\mathrm{T}}_{N\times 2},$$
$$x = \begin{bmatrix} b_0^r & b_1^r & \cdots & b_{N-1}^r \\ k_0^r & k_1^r & \cdots & k_{N-1}^r \end{bmatrix}_{2\times N},$$
$$\beta = \begin{bmatrix} y_{0,0} & \cdots & y_{N-1,0} \\ y_{0,1} & \cdots & y_{N-1,1} \\ \vdots & \ddots & \vdots \\ y_{0,N-1} & \cdots & y_{N-1,N-1} \end{bmatrix}_{N\times N}. \quad (6)$$

Here, $N = 8$, half of the number of input bits; $y_{i,j} = y([i,j]) = \arccos((2^N i + j)/2^{2N})/(2\pi)$, $i, j = 0, 1, \ldots N-1$, and i acts as the address for the LUT. The optimal floating-point parameters b^r, k^r yield a maximum absolute error $< 2^{-16}$ for the input range $x \in [0, 0.963]$. For input $x \in (0.963, 1]$, the PWL approximation does not behave as well because of the large derivative value when the input approaches 1. However, this case only happens when the input sample vector nearly aligns with the two decomposed vectors, namely A is approaching $a_1 + a_2$ and α_1, $\alpha_2 \to 0$. One solution is to redefine the threshold values such that those samples use a set of higher level of power supplies so as to avoid the situations of $\alpha_1, \alpha_2 \to 0$.

Then, we quantize the terms b^r and k^r to 8 bits, and use (4) to obtain the offset S. It turns out that the offset parameter uses 11 bits. And the resulting accuracy after all the quantization is $< 2^{-15}$ in terms of maximum absolute error.

Table III shows the place and route results of the hardware implementation with the proposed approximation approach, as well as other approaches as comparisons. There are two versions of the approximation approach shown there with different ways of handling the LUT: one version has the LUT programmable and the other version has it hardwired. The approaches shown there as comparisons include CORDIC and a 6th order polynomial approximation. CORDIC [22] is a general iterative approach to implement the trigonometric functions. However, due to its general purpose, it is much less energy-efficient and with lower throughput compared to our PWL approximation. The polynomial approximation, as another alternative to approximate the nonlinear functions, requires much more multipliers than the PWL approximation, hence is also less energy-efficient. As a summary, the proposed PWL approximation provides 6–20× improvement in energy-efficiency with significant area savings over the competing approaches.

IV. CHIP IMPLEMENTATION

A. Overall Chip Design

The baseband design uses the 64-QAM modulation scheme and has the target symbol throughput of 1–2 GSym/s. The system has an oversampling rate of 4 or 2, resulting in a system sample throughput of 4 GSam/s. The baseband needs to provide at −60 dB adjacent channel power ratio (ACPR). In order to meet this specification while overcoming the nonlinearity in the phase modulator DAC [18], the baseband is designed to achieve −65 dB ACPR with 12-bit phase quantization.

The baseband system has a block diagram as shown in Fig. 4. It includes two parts of the design: supporting blocks and AMO SCS. The supporting blocks upsample and pulse-shape

TABLE III
COMPARISON BETWEEN PWL, CORDIC IMPLEMENTATIONS OF THE 16-bit INPUT, OUTPUT FUNCTION $y(x) = cos^{-1}(x)$

	Minimal clock period(ps)	Power consumption (mW) (post-extraction simulation)	Area (μm $\times\mu$m), Density (%)	Energy per operation (pJ/op)
Proposed PWL (hardwired LUT)	792	3.24 (at 1GHz)	80×60, 80%	3.24
Proposed PWL (programmable LUT)	856	7.23 (at 1GHz)	250×240, 77.5%	7.23
Unrolled radix-4 CORDIC	2600	63.1 (at 400MHz)	220×200, 81.4%	157.75
6th order polynomial	250	42 (at 1GHz)	200×200, 70%	42

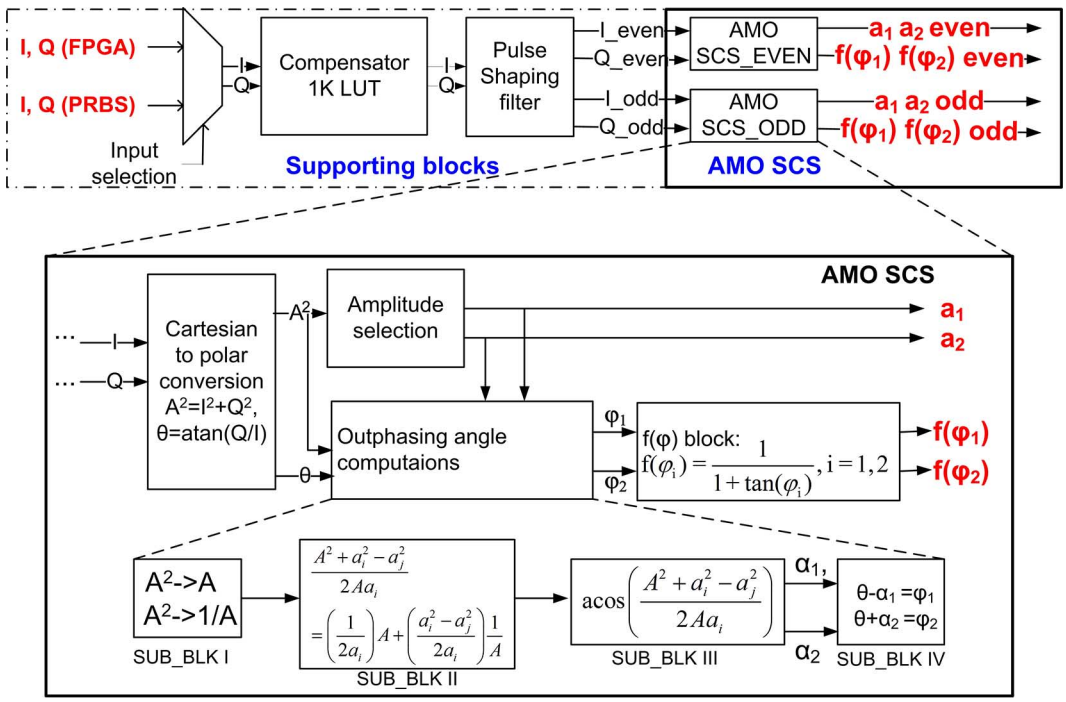


Fig. 4. The block diagram of the chip.

the input symbol sequence from the 64-QAM constellation to appropriate sample sequences, which are then fed to the AMO SCS blocks. Shown in Fig. 4, the 3-bit I and Q symbols first pass through a LUT-based nonlinear predistorter with a size of $(2^{10}) \times 24$ and produce I/Q symbols with 12-bit accuracy in each dimension. The system is not designed to have a powerful nonlinear predistorter, so this simple predistortion table is added only for preliminary symbol-space predistortion. The table size is chosen such that the predistorter has some memory while fitting in the die area. Then the 12-bit I and Q symbols pass through a pulse shaping filter which oversamples the symbols and produces 12-bit I and Q samples with shaped spectrum. Interleaving is explored here to achieve even higher throughput. The shaping filter produces one sample at the positive edge of the clock and another at the negative edge. Therefore, two copies of the AMO SCS blocks follow the even and odd outputs of the filter.

The AMO SCS part, the zoomed-in part in the bottom of Fig. 4, consists of four main sub-blocks: the *Cartesian-to-polar* block, *Amplitude-selection* block, *Outphasing-angle-computation* block, and the angle function $f(\varphi)$ block. The *Cartesian-to-polar* block computes the amplitude square and the angle of the I/Q samples in polar coordinates, corresponding to equation (amo1) in Table I.

The *Amplitude-selection* block then takes the value of amplitude square and selects the pair of power supplies for the PAs in the two paths. Recall that the initial motivation to modify the LINC architecture to the AMO architecture is to introduce more supply levels to minimize the combiner loss especially when the outphasing angle is large. Therefore, the choice of the power supplies directly affects the average power efficiency. According to the Wilkinson combiner's efficiency [9] at sample amplitude A and two PA's supply voltages a_i, a_j

$$\eta_c(A, a_i, a_j) = \left(\frac{A}{\frac{(a_i+a_j)}{2}}\right)^2 \left(\frac{2(\frac{a_i+a_j}{2})^2}{a_i^2+a_j^2}\right), \qquad (7)$$

we design the criterion shown in Table IV to select the pair of power supplies, where

$$[th_1, th_2, \cdots, th_7] = [(2V_1)^2, (V_1+V_2)^2, (2V_2)^2, (V_2+V_3)^2, (2V_3)^2, (V_3+V_4)^2, (2V_4)^2], \qquad (8)$$

and $V_1 \le V_2 \le V_3 \le V_4$ are the four available power supply levels. The criterion is designed to maximize the combiner's efficiency (7) by using the smallest pair of power supplies while still the power levels are large enough to form the transmitted sample. Obviously, there are more than the 7 levels used here

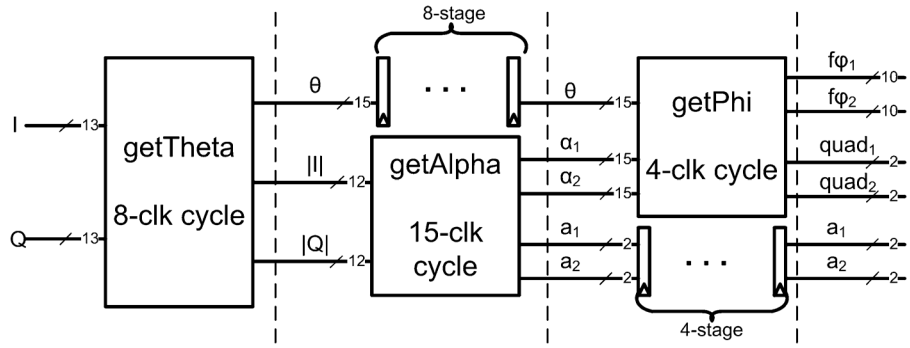


Fig. 5. The hardware block diagram of the SCS system.

TABLE IV
CRITERION FOR POWER SUPPLY PAIR SELECTION. ($A^2 = I^2 + Q^2$)

a_1, a_2	Criterion
V_1, V_1	$A^2 \le th_1$
V_1, V_2	$th_1 < A^2 \le th_2$
V_2, V_2	$th_2 < A^2 \le th_3$
V_2, V_3	$th_3 < A^2 \le th_4$
V_3, V_3	$th_4 < A^2 \le th_5$
V_3, V_4	$th_5 < A^2 \le th_6$
V_4, V_4	$th_6 < A^2 \le th_7$

TABLE V
SUMMARY OF ARITHMETIC OPERATIONS IN EACH FUNCTIONAL BLOCK OF THE AMO SCS

Functional block		**Arithmetic operations**
Cartesian-to-polar		multiply, division, arctan
Amplitude selection		Comparator
Outphasing angles	SUB_BLK I	square-root, inversion of square-root
	SUB_BLK II	multiply, add
	SUB_BLK III	arccos
	SUB_BLK IV	add
$f(\varphi)$ block		$\frac{1}{1+tan(\varphi)}$

that can be designed from 4 supply levels. An important factor that motivates the choice of the 7 levels is the consideration of minimizing the number of switching events with each of the power supply. Power supply switching is accompanied by ringing and slewing, which introduce nonlinear and memory effects into the system and cause the spectrum outgrowth and degradation in the linearity performance of the overall transmitter. The rules in (8) make only one adjacent power supply change when the sample amplitude jumps from one region to an adjacent region. This is what happens most of the time because the pulse-shaping filter smooths the I/Q symbol transitions and limits the jumps between I/Q samples.

The *Outphasing-angle-computation* block computes the two angles between the decomposed and transmitted vectors, corresponding to equations (amo2) and (amo3) in Table I. The steps of the computations are divided into four sub-blocks in Fig. 4. Sub-blocks I and II compute the argument of the arccosine function $(A^2 + a_i^2 - a_j^2)/(2Aa_i)$, including square-root, inverse of square-root and summation operations. The terms $1/2a_i$ and $(a_i^2 - a_j^2)/(2a_i)$ in sub-block II are two programmable constants and selected after the determination of two supply levels. Then sub-block III computes the arccosine function and IV computes the final outphasing angles.

The last block of $f(\varphi)$ computation prepares the input signals for the phase modulator we use, which take the form of $1/(1 + tan(\varphi))$. The LUT used in this block can also be programmed to compensate the static nonlinearity of the phase modulator DAC.

As a summary, Table V lists the arithmetic operations for each functional block.

B. SCS Blocks Design

In this section, we show details of the micro-architecture of each block in the SCS system. Fig. 5 shows the overall pipelined hardware block diagram. It is roughly a direct translation from the conceptual block diagram in Fig. 4. The I/Q samples generated by the shaping filter first pass through the *getTheta* block and produce the θ and $|I|$, $|Q|$. The following *getAlpha* block then takes $|I|$ and $|Q|$, selects the two power supplies and computes the angles α_1 and α_2. This roughly corresponds to the *Amplitude-selection* and *Outphasing-angle-computation* blocks in Fig. 4. The angles α_1 and α_2, together with θ, are inputs to the *getPhi* block, which computes the function $1/(1 + tan(\varphi))$ on the outphasing angles φ_1, φ_2. This represents the $f(\varphi)$ block in Fig. 4. The final outputs of the SCS system are $f\varphi_1$, $f\varphi_2$, $quad_1$, $quad_2$, and a_1, a_2. Here, $quad_1$ and $quad_2$ are quadrant indicators of φ_1 and φ_2, respectively; $f\varphi_1$, $f\varphi_2$ are computed with φ_1 φ_2 converted to the first quadrant; a_1 and a_2 are the digital codes that control the PA power supply switches. Next, we see how each sub-block accomplishes its tasks.

1) getTheta Block: Fig. 6(a) shows the micro-architecture of the *getTheta* block, which has two main operations as division and arctan. With the PWL approximation algorithm discussed in Section III-A, both functions can be realized with the micro-architecture in Fig. 3. Before applying the approximation, it is important to carefully examine the input and output range of the function, because of the nature of the fixed-point computation. In order to have a good accuracy with the approximation, it is desirable to have an input range where the function behaves smoothly and has a nicely bounded derivative. Consider as an example the division function. The division function Q/I has two input variables, while the presented algorithm assumes a single variable function. So the computation of Q/I is divided into $1/I$, followed by $Q \times (1/I)$. The inversion function $1/I$ has a discontinuity at $I = 0$ and its derivative $-1/I^2$ becomes large as $|I|$ approaching zero. In order to use the PWL approximation with good accuracy, several preprocessing steps are necessary

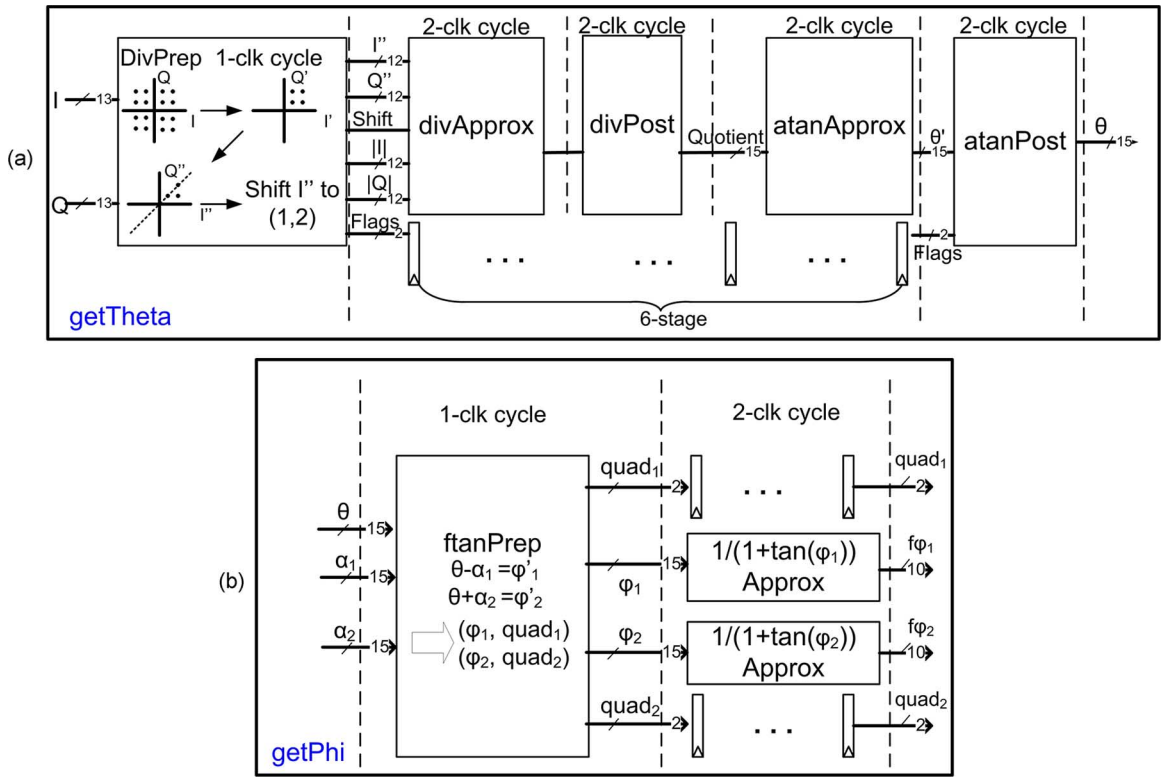


Fig. 6. (a) The hardware block diagram of the *getTheta* block. (a) The hardware block diagram of the *getPhi* block.

to massage the input before doing the approximation of the inversion function $1/I$. We implement the following treatments on the input, corresponding to the *divPrep* block in Fig. 6(a):

- Step (1): (I, Q) are first transformed to the first quadrant as (I', Q') where $I' = |I|$ and $Q' = |Q|$. Use a flag of two bits to indicate whether the current sample (I, Q) is actually negative or not.
- Step (2): Swap I' and Q' if $Q' > I'$, so the resulting (I'', Q'') satisfies $Q''/I'' \in (0, 1)$. The boundary values of 0 and 1 are computed as special cases separately. Again, use a flag to indicate whether the swap is performed on the current sample.
- Step (3): Shift the input I'' such that $I'' \in (1, 2)$. The shift operation is always valid because the shaping filter coefficients are programmable and can be designed such that $I, Q \in [0, 1]$. This step just means shifting the bits in I'' to the left until the MSB is 1. Record the shifted number of bits for each sample I''.

Although it is obvious that after the transformations, Q''/I'' is different from the desired output Q/I, these preprocessing steps can be compensated. Specifically, the swap in Step (2) and the absolute operation in Step (1) are taken care of after the computation of θ; and the shift operation in Step (3) are taken care of after the computation of $Q'' \times (1/I'')$.

- Step (1): Shift back accordingly after the computation of $Q'' \times (1/I'')$. This is an operation included in the block of *divPost*, together with the multiplication $Q'' \times (1/I'')$.
- Step (2): After the computation of θ', for values whose flag indicating a swap operation has happened, $\theta = \pi/2 - \theta'$, otherwise $\theta = \theta'$. This is included in the *atanPost* block in Fig. 6(a).
- Step (3): After Step (2), we need to check further if quadrant change has happened to the current sample, and adjust the θ accordingly. This is also a part of *atanPost* block.

With properly designed preprocessing, the input of inversion function $1/x$ takes the range of (1, 2), and the input of function $\arctan(x)$ takes the range of (0, 1). In these ranges, the functions have nicely bounded derivatives, enabling them to be suitable for the fixed-point PWL approximation. The two function's approximation computations are represented by the blocks *divApprox* and *atanApprox* in Fig. 6(a), whose micro-architecture follows the one in Fig. 3(a). The overall *getTheta* block is able to achieve a throughput of 2 GSamples/s in the place and route timing analysis. The look-up tables that store the b, S, and k for the two functions have sizes as summarized in the first two lines in Table VI. The table also gives a size comparison to the LUTs which are used directly to map the nonlinear functions. There, we can see orders of magnitude of LUT size saved by using our fixed-point PWL approximation approach. The accuracy column also shows that an output accuracy of 14 bit is achieved.

2) getAlpha Block: Fig. 7 demonstrates the detailed micro-architecture of the *getAlpha* block of Fig. 5, also corresponding to the conceptual sub-blocks I, II and III of the *Outphasing-angle-computation* part in Fig. 4. The α_1 and α_2 computations include two parts: obtain the argument to the $\arccos$ function and calculate the $\arccos$ function itself. In order to obtain the argument $(a_i^2 + A^2 - a_j^2)/(2Aa_i)$, we rearrange the terms as

$$\frac{a_i^2 + A^2 - a_j^2}{2Aa_i} = c_1 A + c_2 \frac{1}{A}, \text{ and } c_1 = \frac{1}{2a_i}, c_2 = \frac{a_i^2 - a_j^2}{2a_i}, \tag{9}$$

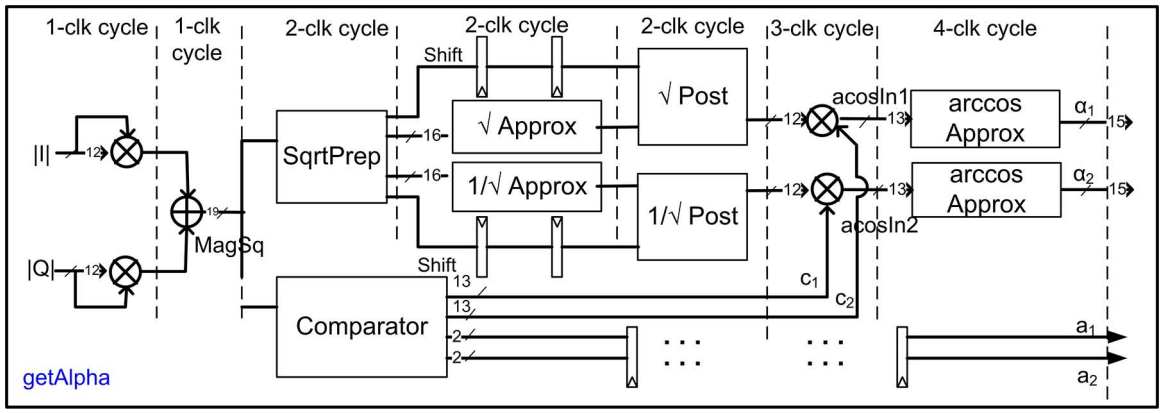


Fig. 7. The hardware block diagram of the *getAlpha* block.

TABLE VI
SUMMARY OF ACCURACY AND LUT SIZE OF THE PWL APPROXIMATED FUNCTION BLOCKS

	max |error|	PWL LUT size	Direct LUT size	Improvement ratio
$1/x$	7e-5	30×2^7	15×2^{12}	4
$\arctan(x)$	6e-5	25×2^7	15×2^{15}	128
$\sqrt{x}$	2.3e-5	30×2^7	12×2^{19}	1638
$1/\sqrt{x}$	8.2e-5	30×2^7	12×2^{19}	1638
$\arccos(x)$	2.4e-5	30×2^7	15×2^{15}	128
$1/(1 + \tan(x))$	1.6e-5	26×2^7	10×2^{15}	100

where constants c_1 and c_2 are programmable values and are selected according to the selection of power supplies. The problem with using the original formula $(a_i^2 + A^2 - a_j^2)/(2Aa_i)$ is the long-bit division, whose inputs are on the same order of A^2. On the other hand, (9) involves no computations with inputs on the order of A^2.

The computations to obtain the terms A, $1/A$ in (9) include approximations of the functions $\sqrt{x}$ and $1/\sqrt{x}$, whose inputs are the sum of $|I|^2$ and $|Q^2|$. Similarly as we discussed for the division computation, certain input preprocessing is necessary to avoid the large derivatives near discontinuity point at 0. The *SqrtPrep* block of Fig. 7 serves this purpose by scaling the input to the range of [1/4, 1), namely shifting two bits at a time either to the left or right until the input fits to the range. Then the approximations to the two functions are performed and followed by the postprocessing parts that compensate for the shifting operations done to the inputs. With two more multipliers and one adder, the computations of (9) are now accomplished. Then the function $\arccos(x)$ takes the input arguments and obtain angles α_1, α_2, which is already shown in the previous example. For the three functions, The LUT sizes and accuracy for the three functions are summarized in Table VI.

3) getPhi Block: Shown in Fig. 6(b) and as the final block in Fig. 5, *getPhi* takes the outputs α_1, α_2 and θ from the previous *getAlpha* and *getTheta* blocks and produces the final outphasing angles $f\varphi_1$ and $f\varphi_2$. The *getPhi* block first computes the outphasing angles φ_1, φ_2 in the sub-block *ftanPrep*, then $1/(1 + tan(\varphi))$ block computes the final outputs. Nominally, the digital baseband SCS's tasks end after the *ftanPrep*, delivering the outphasing angles themselves. However, there may be additional signal processing task at the interface between the digital baseband and the DRFPC phase modulator. In our case, the phase modulator we intend to use requires such a function on the outphasing angle as input.

After obtaining the outphasing angles as $\varphi_1 = \theta - \alpha_1$ and $\varphi_2 = \theta + \alpha_2$, we convert them to the first quadrants and use 2-bit flags $quad_1$ and $quad_2$ to indicate the quadrants. This conversion is necessary both for the sake of the phase modulator input requirement, as well as acting as a preprocessing step for the following functional approximation. By limiting the input to the first quadrant, the function $1/(1 + tan(\varphi))$ has nicely bounded derivative as $-1/(1 + sin(2\varphi))$ in the range of $[0, \pi/2]$. Otherwise, the function has a discontinuity at $3\pi/4$. So it is suitable to apply the PWL approximation on this function as well. The hardware cost in terms of the LUT size is again summarized in Table VI.

C. Experimental Results

With all nonlinear functions properly approximated and parameters quantized, the tested SCS output produces the signal spectrum as shown in Fig. 8(a). Compared with the spectrum at the shaping filter's output, the SCS block reduces the ACPR by 2 dB, from 67 dB to 65 dB, due to the approximation and quantization errors. Fig. 8(b) shows the 64 QAM constellation diagram between SCS output and ideal input, illustrating that the SCS introduces EVM of 0.08%.

The digital AMO SCS system is fabricated in a 45 nm SOI process, with 448578 gates occupying the area of 1.56 mm^2. The chip runs up to 1.7 GHz (3.4 Gsample/s) at 1.1 V supply. As shown in the shmoo plot of Fig. 9, lowering the power supply voltage decreases the dynamic power of the SCS digital system until it hits the minimum-energy point at lower throughput, where leakage energy takes over. The minimum-energy point of 58 pJ per sample or 19 pJ per bit in 64-QAM transmission (assuming $2\times$ oversampling) is measured at 800 MSamples/s throughput. For typical PA efficiency of 40% and throughput of 800 MSamples/s, at peak output power level of 1.8 W, the total peak PAE is affected by less than 1% (46 mW/(46 mW + 1.8 W/0.4)) by this 64-QAM capable AMO SCS backend.

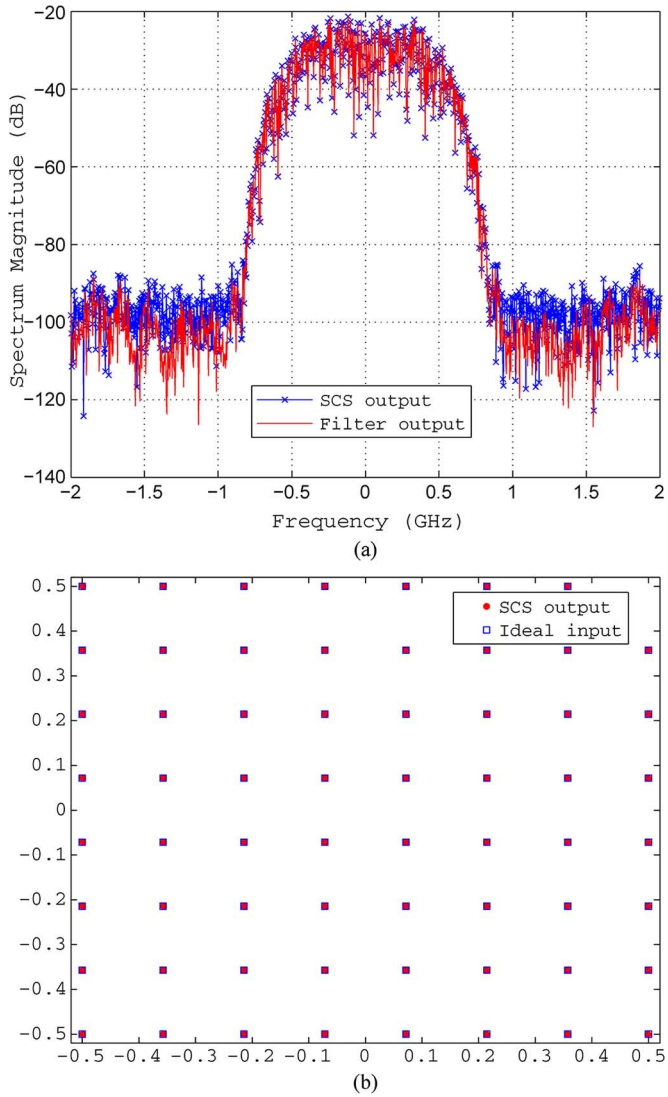


Fig. 8. Spectrum and EVM of the SCS. (a) Spectrum comparison of the SCS output and shaping filter output. (b) EVM comparison of the SCS output and ideal input.

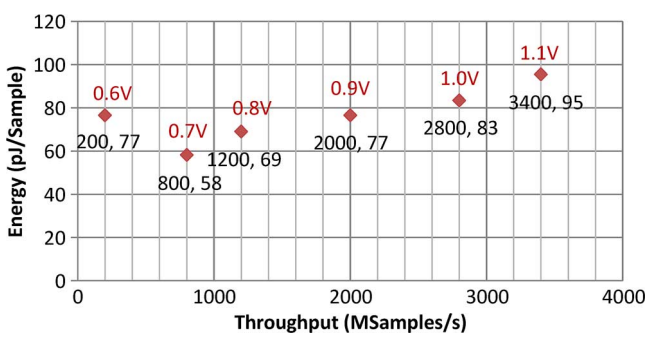


Fig. 9. Throughput and energy with supply scaling for AMO SCS.

The chip photograph is shown in Fig. 10, with annotated blocks and sizes. The power breakdown of the AMO SCS is illustrated in Fig. 11(a). Based on the reported post-place and route power estimation values, the estimated contribution to the total AMO SCS power at 2 GHz operation is shown. The

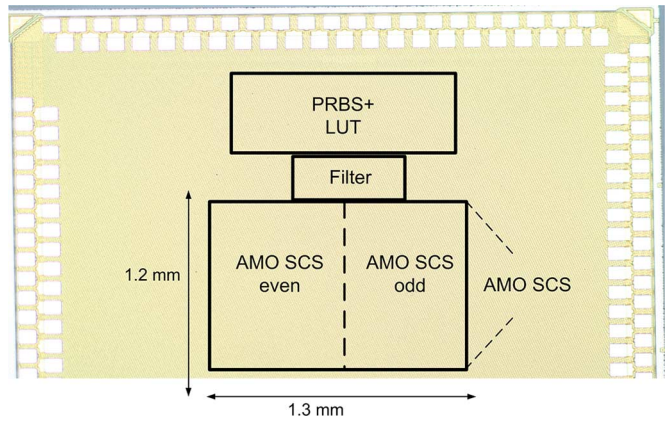


Fig. 10. Chip photograph.

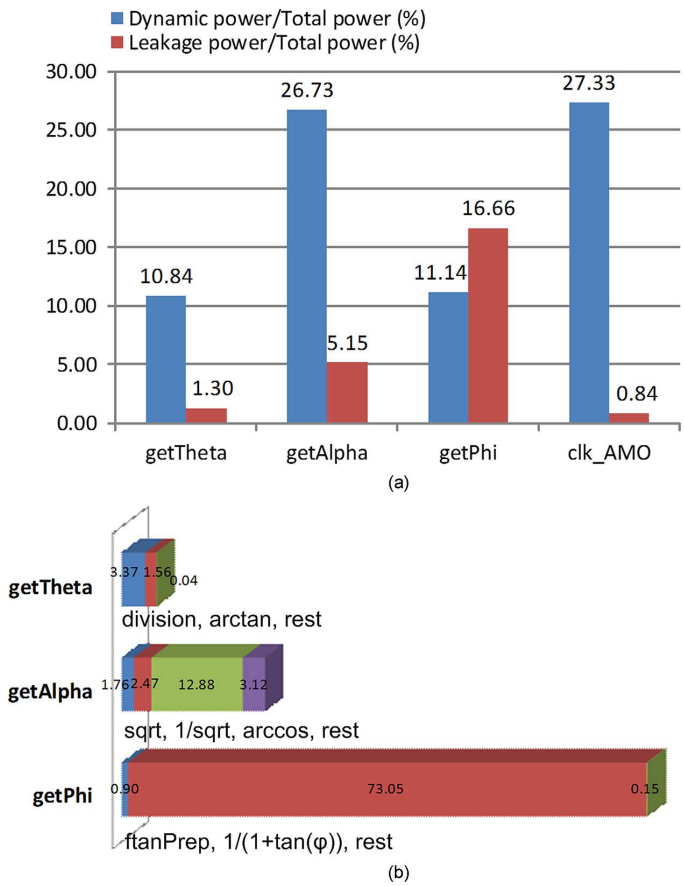


Fig. 11. (a) Power breakdown of the AMO SCS design. (b) Area breakdown of the AMO SCS design.

large proportion of the clocking power is in part due to the latency-matching register stages on amplitude paths required to compensate for the depth of the phase computations, and the leakage power of the *getPhi* block is due to its programmable LUT of the $f(\varphi)$ function. The area breakdown of the AMO SCS is illustrated in Fig. 11(b), which shows the areas of major functional blocks of the three main functions of the SCS. The computation of the function of $f(\varphi)$ takes over two thirds of the area due to its programmable LUTs. A comparison of our work with other digital/analog implementations of LINC/AMO SCS is summarized in the first 5 columns of the Table VII. Our work

TABLE VII
COMPARISON WITH OTHER WORKS

	This work	[13]	[23]	[19]	[15]	[15]	[15]	[15]
Analog/Digital	Digital	Analog	Analog	Digital	Digital	Digital	Digital	Digital
Functionality	AMO	LINC	LINC	LINC	AMO	AMO	AMO	AMO
Technology	45nm SOI CMOS	0.25μm CMOS	0.35μm CMOS	90nm CMOS	90nm CMOS	90nm CMOS	**Scaled to 45nm CMOS**	**Scaled to 45nm CMOS**
Throughput	**3.4GSam/s, 0.8GSam/s**	20MSam/s	1.5MSam/s	50MSam/s	40MSam/s	40MSam/s	40MSam/s	**Scaled to 0.8GSam/s**
Phase Resolution	12-bit	N/A	N/A	8-bit	8-bit	**Scaled to 12-bit**	**Scaled to 12-bit**	**Scaled to 12-bit**
Power	323mW, 46mW	45mW	80mW	0.95mW	0.36mW	8.64mW	4.32mW	86.4mW
Energy/Sample	95pJ/Sam, 58pJ/Sam	2250pJ/Sam	5333pJ/Sam	19pJ/Sam	8.9pJ/Sam	212pJ/Sam	106pJ/Sam	106pJ/Sam
Area	1.5mm^2	0.1mm^2	0.61mm^2	0.06mm^2	0.34mm^2	8.16mm^2	2.04mm^2	40.8mm^2

demonstrates a design with the highest throughput and phase accuracy to date. To show a more fair comparison with other digital AMO SCS work, we scaled the design in [15] to provide the same phase accuracy, technology node and throughput. The scaled performances are summarized in the last 3 columns of the Table VII, and our design shows more than 2× improvement in energy-efficiency and 25× improvement in area. As a general guideline, for applications with low/medium accuracy (e.g. less than 8-bit phase resolution) requirement and low/medium throughput (e.g. up to hundreds of MSamples/s), LUT is still a good design choice because of its low energy-efficiency, reasonable size and low design complexity. On the other hand, our proposed approach is more suitable for applications with high accuracy (e.g. greater than 10-bit phase resolution) and high throughput (e.g. around GSamples/s) requirements.

V. CONCLUSION

In this paper, we present a chip design of a high-throughput (3.4 GSamples/s) SCS for the AMO PA architecture. In order to achieve energy- and area-efficient high-throughput operation, we developed a new fixed-point piece-wise linear approximation algorithm for the computations of the nonlinear functions in SCS design. This new algorithm and the corresponding implementation achieve over 2× improvement in energy efficiency and 25× improvement in area efficiency over the traditional AMO SCS implementations. The algorithm has nice properties of few and simple arithmetic operations, short arithmetic operands and small-sized look-up tables, and can be easily pipelined to run at multi-GSamples/s throughputs. Designed in 45 nm SOI technology, this SCS implementation is the fastest SCS implementation demonstrated to date. Though we demonstrate the application of the approximation algorithm with the AMO SCS, the approximations are directly applicable to LINC SCS, and enable a new class of wideband wireless mm-wave communication system designs with high energy and spectral efficiency.

ACKNOWLEDGMENT

The authors would like to thank Mark M. Tobenkin, and Joel Dawson, David Ricketts, Wei Tai, Zhen Li, and Gilad Yahalom from the MIT-CMU ElastX team, for their useful discussions and kind help.

REFERENCES

[1] J. Laskar, S. Pinel, S. Sarkar, B. Perumana, and P. Sen, "The next wireless wave is a millimeter wave," *Microwave J.*, vol. 50, no. 8, pp. 22–36, Aug. 2007.

[2] S. Pinel, P. Sen, S. Sarkar, B. Perumana, D. Dawn, D. Yeh, F. Barale, M. Leung, E. Juntunen, P. Vadivelu, K. Chuang, P. Melet, G. Iyer, and J. Laskar, "60 GHz single-chip CMOS digital radios and phased array solutions for gaming and connectivity," *IEEE J. Sel. Areas Commun.*, vol. 27, no. 8, pp. 1347–1357, Oct. 2009.

[3] C. Marcu, D. Chowdhury, C. Thakkar, L.-K. Kong, M. Tabesh, J.-D. Park, Y. Wang, B. Afshar, A. Gupta, A. Arbabian, S. Gambini, R. Zamani, A. Niknejad, and E. Alon, "A 90 nm CMOS low-power 60 GHz transceiver with integrated baseband circuitry," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2009, pp. 314–315.

[4] M. Tabesh, J. Chen, C. Marcu, L. Kong, S. Kang, E. Alon, and A. Niknejad, "A 65 nm CMOS 4-element sub-34 mW/element 60 GHz phased-array transceiver," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2011, pp. 166–168.

[5] S. Nicolson, K. Yau, S. Pruvost, V. Danelon, P. Chevalier, P. Garcia, A. Chantre, B. Sautreuil, and S. Voinigescu, "A low-voltage SiGe BiCMOS 77-GHz automotive radar chipset," *IEEE Trans. Microw. Theory Tech.*, vol. 56, no. 5, pp. 1092–1104, May 2008.

[6] R. Ben Yishay, R. Carmon, O. Katz, and D. Elad, "A high gain wideband 77 GHz SiGe power amplifier," in *Proc. IEEE Radio Frequency Integrated Circuits Symp. (RFIC)*, May 2010, pp. 529–532.

[7] A. Arbabian, B. Afshar, J.-C. Chien, S. Kang, S. Callender, E. Adabi, S. Toso, R. Pilard, D. Gloria, and A. Niknejad, "A 90 GHz-carrier 30 GHz-bandwidth hybrid switching transmitter with integrated antenna," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2010, pp. 420–421.

[8] D. Cox, "Linear amplification with nonlinear components," *IEEE Trans. Commun.*, vol. COM-22, no. 12, pp. 1942–1945, Dec. 1974.

[9] S. Chung, P. Godoy, T. Barton, E. Huang, D. Perreault, and J. Dawson, "Asymmetric multilevel outphasing architecture for multi-standard transmitters," in *Proc. IEEE Radio Frequency Integrated Circuits Symp.*, Jun. 2009, pp. 237–240.

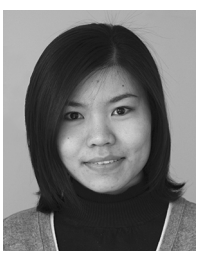
[10] P. Godoy, S. Chung, T. Barton, D. Perreault, and J. Dawson, "A 2.5-GHz asymmetric multilevel outphasing power amplifier in 65-nm CMOS," in *Proc. IEEE Topical Conf. Power Amplifiers for Wireless and Radio Applications (PAWR)*, Jan. 2011, pp. 57–60.

[11] J. Hur, O. Lee, K. Kim, K. Lim, and J. Laskar, "Highly efficient uneven multi-level LINC transmitter," *Electron. Lett.*, vol. 45, no. 16, pp. 837–838, 30 2009.

[12] B. Shi and L. Sundström, "A 200-MHz IF BiCMOS signal component separator for linear LINC transmitters," *IEEE J. Solid-State Circuits*, vol. 35, no. 7, pp. 987–993, Jul. 2000.

[13] L. Panseri, L. Romano, S. Levantino, C. Samori, and A. Lacaita, "Low-power signal component separator for a 64-QAM 802.11 LINC transmitter," *IEEE J. Solid-State Circuits*, vol. 43, no. 5, pp. 1274–1286, May 2008.

[14] W. Gerhard and R. Knoechel, "LINC digital component separator for single and multicarrier W-CDMA signals," *IEEE Trans. Microw. Theory Tech.*, vol. 53, no. 1, pp. 274–282, Jan. 2005.
[15] T.-W. Chen, P.-Y. Tsai, D. De Moitie, J.-Y. Yu, and C.-Y. Lee, "A low power all-digital signal component separator for uneven multi-level LINC systems," in *Proc. ESSCIRC*, Sep. 2011, pp. 403–406.
[16] J. E. Volder, "The Cordic trigonometric computing technique," *IRE Trans. Electron. Comput.*, vol. EC-8, no. 3, pp. 330–334, Sep. 1959.
[17] M. Schetzen, *The Volterra and Wiener Theories of Nonlinear Systems*. Malabar, FL: Krieger Publishing Co., 2006, vol. 1.
[18] P. Eloranta, P. Seppinen, S. Kallioinen, T. Saarela, and A. Parssinen, "A multimode transmitter in 0.13 μm CMOS using direct-digital RF modulator," *IEEE J. Solid-State Circuits*, vol. 42, no. 12, pp. 2774–2784, Dec. 2007.
[19] T.-W. Chen, P.-Y. Tsai, J.-Y. Yu, and C.-Y. Lee, "A sub-mW all-digital signal component separator with branch mismatch compensation for OFDM LINC transmitters," *IEEE J. Solid-State Circuits*, vol. 46, no. 11, pp. 2514–2523, Nov. 2011.
[20] C. Conradi, J. McRory, and R. Johnston, "Low-memory digital signal component separator for LINC transmitters," *Electron. Lett.*, vol. 37, no. 7, pp. 460–461, Mar. 2001.
[21] K. Eriksson and D. Esten, *Applied Mathematics: Body and Soul: Volume 2: Integrals and Geometry in* $\mathbb{R}^n$. New York: Springer, 2010, vol. 2.
[22] P. Meher, J. Valls, T.-B. Juang, K. Sridharan, and K. Maharatna, "50 years of cordic: Algorithms, architectures, and applications," *IEEE Trans. Circuits Syst. I: Reg. Papers*, vol. 56, no. 9, pp. 1893–1907, Sep. 2009.
[23] B. Shi and L. Sundstrom, "An IF CMOS signal component separator chip for LINC transmitters," in *Proc. IEEE Custom Integrated Circuits Conf. (CICC)*, 2001, pp. 49–52.

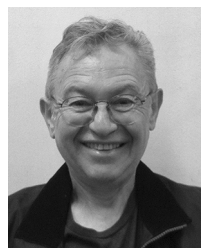


Yan Li received the B.E. degree in electrical engineering from University of Science and Technology of China in 2004, and the M.A.Sc. degree in electrical engineering from McMaster University, Canada in 2006. She is currently pursuing the Ph.D. degree in electrical engineering and computer science from the Massachusetts Institute of Technology (MIT), Cambridge, MA.

Her current research interests include the design of high-speed energy-efficient digital systems, nonlinear systems modeling and compensation, and optimization for analog integrated circuits.

Zhipeng Li received the S.B. in physics, S.B. in electrical engineering, M.Eng., and Electrical Engineer degrees from the Massachusetts Institute of Technology (MIT), Cambridge, MA, in 2009, 2009, 2010, and 2011, respectively, where he is currently pursuing the Ph.D. degree. His research interests include design and implementation of energy-efficient digital circuits.

Oguzhan Uyar was born in Yozgat, Turkey, in 1986. He received the M.S. degree in electrical engineering from Massachusetts Institute of Technology (MIT), Cambridge, MA, in 2011 and the B.S. degree in electrical engineering from Bogazici University in 2009. His main areas of interest are high-speed mixed signal circuits, gigabit serial links, equalization and wireless transceivers.



Yehuda Avniel received the Ph.D. degree in electrical engineering and computer science from Massachusetts Institute of Technology (MIT), Cambridge, MA, in 1985 in the area of control. He is a Research Affiliate at MIT and a contributor in numerous DARPA initiatives in the applications of robust and hierarchical optimization in nanotechnology, integrated circuit design, and optimization-based non-linear model reduction.

Alexandre Megretski is currently a Professor of Electrical Engineering and Computer Science with the Laboratory for Information and Decision Systems, Massachusetts Institute of Technology (MIT), Cambridge, MA. He was a Researcher with both the Royal Institute of Technology, Stockholm, Sweden, and the University of Newcastle, NSW, Australia, and a Faculty Member with Iowa State University, Ames. His current research interests include nonlinear dynamical systems (identification, analysis, and design), validation of hybrid control algorithms, optimization, applications to analog circuits, control of animated objects, and relay systems.

Vladimir Stojanović received the Ph.D. degree in electrical engineering from Stanford University, Stanford, CA, in 2005, and the Dipl. Ing. degree from the University of Belgrade, Serbia, in 1998.

He is the Emanuel E. Landsman Associate Professor of Electrical Engineering and Computer Science at Massachusetts Institute of Technology (MIT), Cambridge, MA. He was with Rambus, Inc., Los Altos, CA, from 2001 through 2004. His research interests include design, modeling and optimization of integrated systems, from CMOS-based VLSI blocks and interfaces to system design with emerging devices like NEM relays and silicon-photonics. He is also interested in design and implementation of energy-efficient electrical and optical networks, and digital communication techniques in high-speed interfaces and high-speed mixed-signal IC design.

Prof. Stojanović received the 2006 IBM Faculty Partnership Award, and the 2009 NSF CAREER Award as well as the 2008 ICCAD William J. McCalla, 2008 IEEE TRANSACTIONS ON ADVANCED PACKAGING, and 2010 ISSCC Jack Raper best paper awards. He is an IEEE Solid-State Circuits Society Distinguished Lecturer for the 2012–2013 term.