Low-Power Sign-Magnitude FFT Design for FMCW Radar Signal Processing

Oğuz Meteer
o.meteer@utwente.nl
University of Twente
Enschede, The Netherlands

Marco J.G. Bekooij
NXP Semiconductors
Eindhoven, The Netherlands
University of Twente
Enschede, The Netherlands
marco.bekooij@nxp.com

ABSTRACT

Fully integrated CMOS frequency-modulated continuous-wave radar ICs are under development, in which computing FFTs cost a significant amount of energy.

In this paper we introduce a power-efficient FFT solution which exploits that intermediate results of FFT computations typically have small amplitudes in FMCW radar systems. We propose using the sign-magnitude number representation combined with a custom, unsigned Booth multiplier that does not generate negative numbers internally, significantly decreasing switching activity.

RTL power-simulation results show up to 46.45% less power usage with our sign-magnitude radix-2 FFT implementation compared to a two's complement design, while only having a 6.67% lower maximum clock speed.

CCS CONCEPTS

• **Hardware → Digital signal processing**; **Circuits power issues**; **Application specific integrated circuits**.

KEYWORDS

Low-power design, Signal processing systems, FFT, Automotive, FMCW

ACM Reference Format:
Oğuz Meteer and Marco J.G. Bekooij. 2021. Low-Power Sign-Magnitude FFT Design for FMCW Radar Signal Processing. In *Workshop on Design and Architectures for Signal and Image Processing (14 th edition) (DASIP '21), January 18–20, 2021, Budapest (initially), Hungary.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3441110.3441145

1 INTRODUCTION

Frequency-modulated continuous-wave (FMCW) radar CMOS ICs are being developed to be used in autonomous cars as cheap distance and speed sensors. In such ICs, the distance and speed of targets are calculated by applying Fast Fourier Transforms (FFT) on the digitized data coming from an RF front-end. Attaining usable accuracy requires a high amount of digital signal processing (DSP), leading to a high power usage and thus excessive heat dissipation. An important issue is that in cars, these ICs are placed in difficult to cool places, and therefore have to be duty-cycled to reduce heat dissipation.

In this paper we present techniques that reduce the power usage of FFTs in FMCW radar systems. We analyze several key properties of FMCW radar, the FFT, and number representations. By combining these properties, we implement a fixed-point, radix-2 hardware FFT using sign-magnitude encoding resulting in much lower power usage than comparable two's complement implementations. These power savings occur despite needing extra hardware for converting between number formats to efficiently add and subtract numbers. We also apply several architectural optimizations to reduce the critical path. In our gate level power simulations, our proposed design of the FFT butterfly logic has an up to 46.45% lower power usage.

The paper is organized as follows. In Section 2 we first give a brief overview of FMCW radars. Then in Section 3 we describe the basic idea of our proposed solution. In Section 4 we analyze how numbers are processed in FFTs, and give an overview of the two's complement and sign-magnitude number formats. We also describe the structure of array and modified Booth multipliers. In Section 5 we propose several solutions and describe our implementations. Next, in Section 6, we evaluate all implementations, perform static power analysis, and list the power usage, area, and maximum frequencies. Finally, we conclude the paper in Section 7.

2 FREQUENCY-MODULATED CONTINUOUS-WAVE RADAR

Radar systems transmit a signal, receive it reflecting off of objects, and measure the delay between transmission and reception of the signal (Δt) which is proportional to the distance between the radar and those objects. FMCW radar systems transmit signals with increasing frequency over time (a *chirp*), which in addition to measuring Δt, also allows measuring Δf. This is shown in Fig. 1(a). Demodulation is performed by mixing the transmitted and received signals, resulting in a sinusoidal signal called the *beat signal*. This signal contains components with different frequencies, corresponding to the different reflections of each object and their distance R. The FFT is then applied to extract these frequencies.

The proportion of the transmitted and received signal powers is [6]:

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
DASIP '21, January 18–20, 2021, Budapest (initially), Hungary
© 2021 Association for Computing Machinery.
ACM ISBN 978-1-4503-8901-3/21/01…$15.00
https://doi.org/10.1145/3441110.3441145

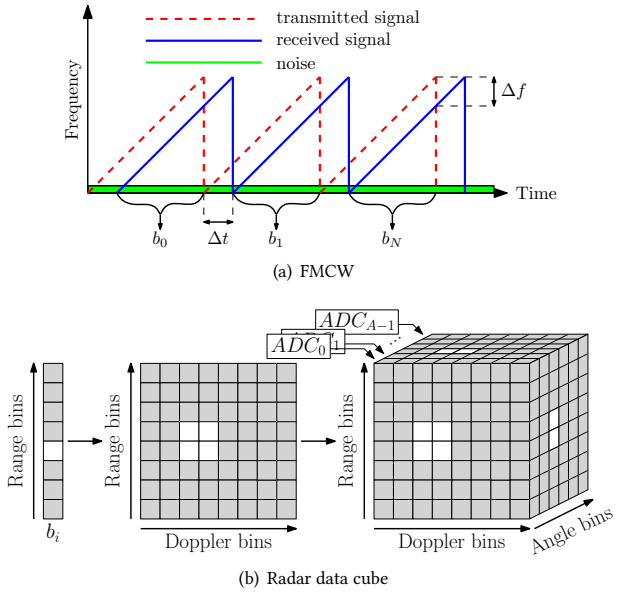


Figure 1: Basic workings of FMCW radar, and the obtained radar data cube with range, Doppler, and angle information.

$$P_{rx} \propto \frac{1}{R^4} P_{tx} \tag{1}$$

where P_{rx} is the power of the received signal, P_{tx} is the peak transmission power, and R is the target range. Although P_{rx} has a high dynamic range, one important aspect is that the power of the received signal is usually much smaller than the transmitted signal due to the sharp decrease in signal power when increasing the distance.

2.1 Radar Data Cube

Range, Doppler, and angle information can be obtained by applying a 3D FFT on the beat signals as shown in Fig. 1(b). First we take the FFT of a single beat signal b_i which yields a column with the ranges of objects. Since beat signals consist of sums of sinusoidal signals, they are typically not small in amplitude. The input to the range FFT can be large when for example there are always strong reflections due to the radar dome or a bumper. Despite this, the output of the range FFT is sparse and yields low signal power in most bins as they contain noise and very small input signals (indicated by a grey color in Fig. 1(b)). Close examination reveals that this also holds for the internal values of FFTs, even when the amplitude of the input signal is large.

Next, we repeat this for all N beat signals and then take the FFT of each row, resulting in the range-Doppler plot. Finally, we compute the range-Doppler plot for each receiver antenna and take the FFT in the Z direction to obtain angle information. Both the Doppler and angle FFTs mostly process very small signals, especially for range bins that correspond to targets at a large distance from the radar.

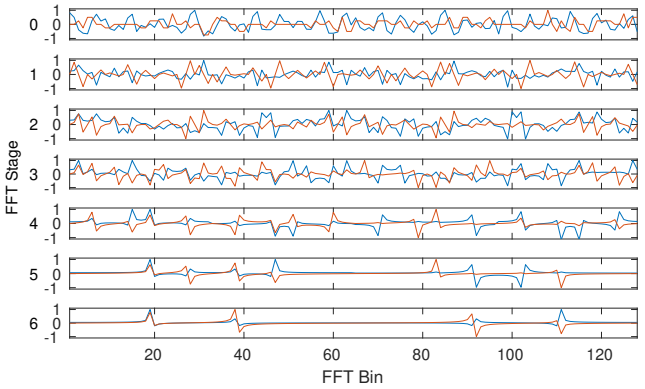


Figure 2: Normalized bin amplitudes after each stage of a 128-bin FFT. Real and imaginary parts are drawn separately.

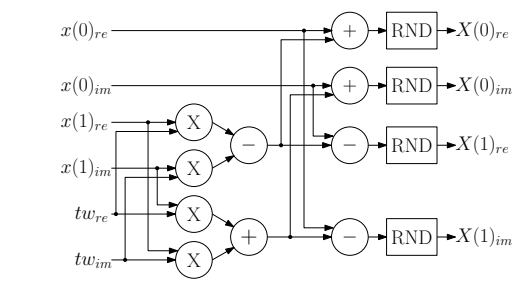
3 BASIC IDEA

To decrease dynamic power usage, we choose to exploit the sign-magnitude number format to reduce the switching activity. We justify our choice due to the following observations.

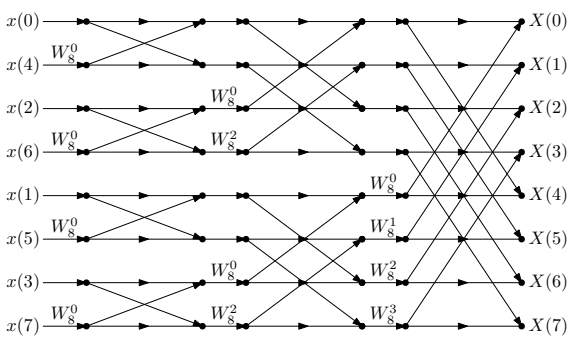
First, a wide data path is necessary because the input signals have a high dynamic range and we want to exploit the processing gain of the FFT to extract very small signals that are hidden in the noise. The range FFTs produce sparse outputs with most bins containing noise and very small signals. The Doppler and angle FFTs therefore process mostly small inputs. Additionally, the internal values of all FFTs overwhelmingly have small amplitudes. Fig. 2 shows the normalized bin amplitudes after each stage for a 128-bin FFT. In this example, the input consists of the sum of Gaussian noise and two cosine waves with amplitudes 0.25 and frequencies that do not fall exactly in the center of a bin, smearing the signal power over multiple bins. We see that the amount of bins with large amplitudes decreases quickly, meaning that many internal values are small. This effect increases with larger FFTs.

Second, using the sign-magnitude number format for low-power designs is not new, but is often discouraged for arithmetic applications, as addition/subtraction is more complex [3] compared to two's complement. The overhead of converting to and from two's complement often nullifies the gains of a lower switching activity. Instead, other number formats [10], mixed number representations [16], and sign bit reduction techniques [12][15] are much more studied. However, evaluation of using the sign-magnitude number format in the FMCW radar FFT context is new, and as our results show in Section 6.1, can lead to significant power-efficiency improvements despite the conversion overhead.

Third, since multipliers are usually the source of the highest switching activity in FFT implementations, power-efficiency improvement is dependent on the used multiplier type. In the case of array multipliers, there is a large difference in switching activity between just multiplying small unsigned numbers compared to small, signed two's complement numbers. This difference however is much smaller when using a modified Booth multiplier [9], because even if all inputs are unsigned numbers, it can still generate negative numbers internally, switching most higher bits. However, Booth multipliers are typically used because of the lower delay



(a) Basic radix-2 FFT butterfly unit.



(b) Structure of an 8-bin radix-2 FFT.

Figure 3: Overview of a radix-2 FFT butterfly unit, and an 8-bin radix-2 FFT example.

resulting in a higher maximum clock frequency. We therefore propose an unsigned Booth multiplier that does not generate negative numbers internally, and thus significantly decreases the switching activity.

4 BACKGROUND

In this section we first give a brief description of the discrete Fourier transform (DFT) and the fast Fourier transform (FFT), the inherent processing gain, and how numbers are processed internally. We then describe the two's complement and sign-magnitude number formats, and elaborate on their advantages and disadvantages.

4.1 Discrete Fourier Transform

For N-periodic discrete signals, the DFT extracts the discrete frequency components and their respective amplitudes. It can be seen as applying a band-pass filter for each output [8] which represents a bin for a specific frequency range. It is defined as:

$$X[k] = \sum_{n=0}^{N-1} x[n] \cdot W_N^{kn} \tag{2}$$

where $x[n]$ is the nth complex input sample, $X[k]$ is the kth transformed sample, and $W_N^{kn} = e^{-i\frac{2\pi kn}{N}}$ is called the *twiddle factor* [5] and is the principal Nth complex root of unity.

4.2 Fast Fourier Transform

The most famous FFT uses the Cooley-Tukey algorithm [4] which reduces the complexity of the DFT from $O(N^2)$ to $O(Nlog_2(N))$, by exploiting the symmetry of the twiddle factors and recursively decomposing the DFT into smaller parts. Given two complex input samples $x[i]$ and $x[j]$, the radix-2 FFT is defined as:

$$\begin{aligned} X[i] &= x[i] + x[j] \cdot W \\ X[j] &= x[i] - x[j] \cdot W \end{aligned} \tag{3}$$

Fig. 3(a) shows the radix-2 FFT *butterfly* unit, which performs (3) and results in a 2-bin FFT. Larger, N-bin FFTs can be calculated by applying (3) in $log_2(N)$ stages. An example of an 8-bin FFT is shown in Fig. 3(b).

4.3 Bit-widths, Rounding, and DC-Bias

The number of bits of the data path determines how accurate numbers can be represented, where wider data paths also require more hardware and generally consume more power.

Results of multiplications require the added sizes of the input operands to guarantee that the result is accurate. To prevent a costly, ever growing data path, truncation is applied after operations to keep the data path width constant.

Truncation typically has two negative effects. First, depending on the used number representation, it can introduce a bias and affect the accuracy of the results. In an FFT, this bias appears in the first bin as it represents the DC-bias in the input signal. In such cases, unbiased rounding is required. Second, truncation can also introduce *quantization* noise due to the finite precision of fixed point representations.

4.4 Processing Gain

The magnitude of a bin containing the signal is proportional to the FFT length N, whereas the magnitude of noise is proportional to $\sqrt{N}$. This interesting property, known as the *processing gain*, is defined as [8]:

$$SNR_N = SNR_{N'} + 10 \cdot log_{10}\left(\frac{N}{N'}\right) \tag{4}$$

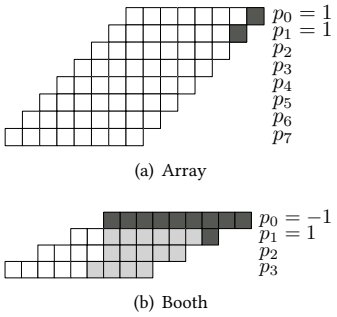
where N and N' are FFT lengths. To increase the SNR, we simply choose a bigger FFT length, i.e. $N > N'$.

When processing beat signals, far away objects have a higher frequency and as such are placed into higher bins. As a consequence of (1), their magnitudes are therefore much smaller, and can even be smaller than noise. The processing gain aids in extracting those small signals.

However, exploiting the processing gain comes at a cost as it requires more input samples and more operations to be executed. Also, the data path needs to be wide enough to store the larger values to not lose accuracy, increasing the number of logic gates and area of the implementation.

4.5 Number Encodings

4.5.1 Two's Complement. An N-bit two's complement number has a range of $[-(2^{N-1}), 2^N - 1]$. A negative number is encoded by applying the two's complement operation on the positive value,



(a) Array

(b) Booth

Figure 4: Example of multiplier power usage approximation. Generated partial products and carry bits are shown dark grey and light grey respectively.

i.e. inverting all bits and adding one. This causes a high switching activity when changing signs. They also inherently round towards $-\infty$, so truncation introduces a bias therefore requiring rounding for sensitive applications.

4.5.2 Sign-Magnitude. For an N-bit sign-magnitude number, the MSB represents the sign, and the remaining bits represent the magnitude. The range is $[-(2^{N-1}) - 1, 2^{N-1} - 1]$. This encoding has low switching activity when changing signs of small values. However, two downsides are the two encodings of 0, namely +0 and −0, and the complexity of adding two numbers with opposite signs [3]. The latter requires converting between the two number representations by only applying the two's complement operation for negative numbers.

4.6 Multipliers

In this section we describe the basic structures of array and Booth multipliers. We then give a general approximation of how values with different magnitudes for the multiplier and multiplicand affect the power usage.

4.6.1 Sign-Magnitude Multiplication. Multiplication of sign-magnitude numbers involves an unsigned multiplication of the magnitudes and an XOR of the sign bits.

4.6.2 Array Multiplier. An AxB array multiplier is used for unsigned numbers. It generates rows of partial products for each bit B_i. When B_i is 1, the partial product of row i is the value of A shifted left by i bits, and is zero otherwise. Each partial product p_{yx} is the logical AND of A_x and B_y. Fig. 4(a) shows the basic structure of an 8x8-bit array multiplier. The Baugh-Wooley multiplier [1] modifies this multiplier such that it also works with two's complement numbers.

4.6.3 Booth Multiplier. The modified Booth multiplier [9] is used for two's complement numbers, and uses a radix-4 scheme to generate half the partial products of array multipliers. It encodes control signals for each overlapping group of three bits of B. Each group of control signals is then decoded to generate one partial product, which ranges between $A \cdot [-2, -1, 0, 1, 2]$. For our reference design, we use the modified Booth multiplier as proposed by Kuang, Wang, and Guo in [7]. Fig. 4(b) shows the basic structure of an 8x8-bit modified Booth multiplier. Using higher radices decreases the number of partial products further, but their implementations can end up being slower than radix-4 [13].

4.6.4 Power Usage. An example of power usage approximation of array and modified Booth multipliers is shown in Fig. 4. Both multipliers perform 1×3, but generate $p_0 + (p_1 \times 2^1) = 1 + (1 \times 2)$ and $p_0 + (p_1 \times 2^2) = -1 + (1 \times 4)$ for the array and Booth multiplier respectively. Even though both inputs are positive, the latter generates a negative number internally and causes carry signals to be propagated, leading to a much higher switching activity.

5 SOLUTIONS

Our proposals are two-fold. First we propose an unsigned Booth multiplier which, for sign-magnitude numbers, has a much lower switching activity than the modified Booth multiplier. Second, we show a hybrid radix-2 FFT butterfly unit that uses both sign-magnitude and two's complement numbers for multiplication and addition/subtraction respectively.

5.1 Unsigned Booth Multiplier

The modified Booth multiplier can internally generate negative numbers even if the inputs are positive, resulting in a high switching activity of the higher bits. We therefore propose an unsigned Booth multiplier that internally only generates positive numbers, thereby decreasing the switching activity when small amplitude numbers are used.

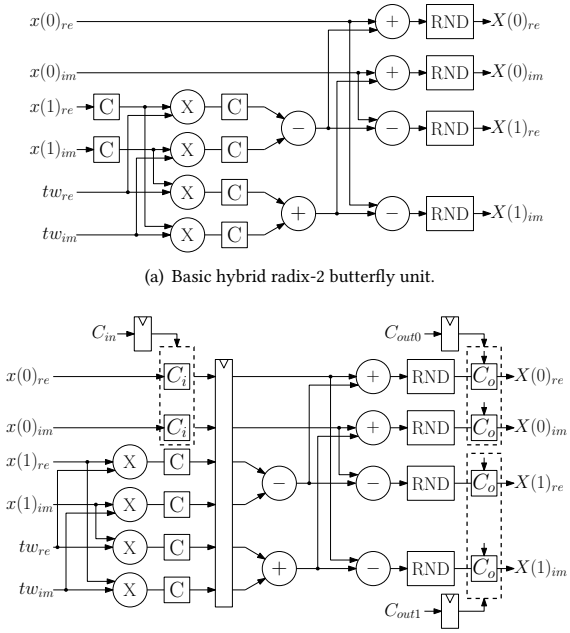
We implemented a radix-4 unsigned Booth multiplier whose partial products can generate 0, A, $2A$, and $3A$. The latter is a hard multiple because it is not trivial to generate with just shifting.

One way to calculate $3A$ is to add A and $2A$. Although optimized adders for this purpose exist [11], it still increases the critical path. We chose an alternative method, and implemented a simplified version of a redundant binary multiplier [2], where we instead generate two partial products per encoder. The first partial product generates $A \cdot [0, 1, 2]$ and the second one generates $A \cdot [0, 1]$. While this almost doubles the number of partial products, the depth of the Wallace tree only increases from six to eight with a 32x32 multiplier, limiting the increase to the critical path.

5.2 Hybrid Butterfly Unit

The basic structure of our hybrid radix-2 butterfly unit is shown in Fig. 5(a). To efficiently add and subtract, it converts between the two number encodings by XORing all bits with the MSB (except the MSB itself), and then adding the MSB to the XORed result. This implementation also correctly treats +0 and −0 as 0.

The pre-computed twiddle factors use sign-magnitude encoding, and the input samples use two's complement encoding. The $x(1)$ inputs are converted to sign-magnitude, and converted back to two's complement after multiplication. The 62-bit results of the multipliers are added and subtracted, and then truncated to 33 bits. The rounding hardware (RND) adds the MSB of the truncated bits to the final result if it is negative, effectively rounding towards zero and removing bias.



(a) Basic hybrid radix-2 butterfly unit.

(b) Retimed and pipelined hybrid radix-2 butterfly unit. C_i and C_o inside dashed boxes indicate conditional converters.

Figure 5: High-level overview of the basic and optimized versions of our hybrid radix-2 sign-magnitude butterfly unit.

5.3 Optimizations

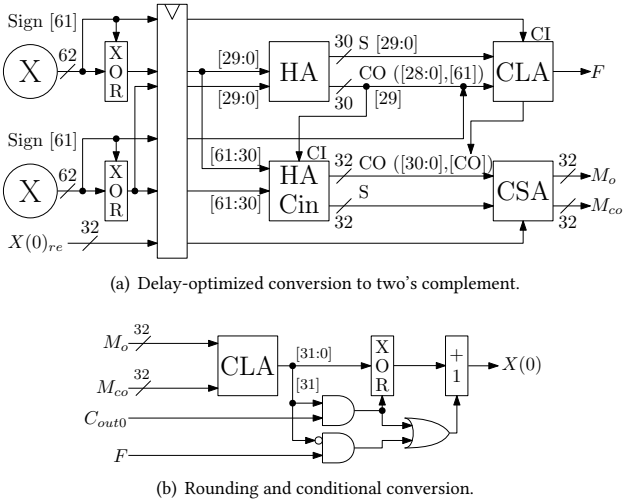
In this section, several optimizations are described, and their impact on power and latency can be found in Section 6.1.

5.3.1 Sign-Change. Subtracters are adders, where the B input is inverted and the carry-in is asserted. We invert the sign bit of the twiddle factor going to the multiplier that computes $x(1)_{im} \cdot tw_{im}$, and change the subtracter that follows to an adder. This eliminates all the inverters of the B input of the following subtracter.

5.3.2 Retiming. XOR gates are known to cause glitches [14]. The converters can therefore also cause glitches throughout the multipliers and adders, increasing unwanted switching activity. To avoid this, we *retimed* and pipelined the design. Both the twiddle factors and input data now use sign-magnitude encoding.

The retimed and pipelined hybrid radix-2 butterfly unit is shown in Fig. 5(b). These implementations have the same four converters after the multipliers, and six *conditional* converters indicated by the dashed lines around them (C_{in}, C_{out0}, and C_{out1}). The conditional converters are similar to the other converters, except the MSB is replaced with a logical AND of the MSB and a selector signal. Pipeline registers are placed after the converters of the multipliers.

Conversion is only needed when a sample needs to be multiplied. C_{in} will only convert all x_0 inputs in the first round as all x_1 inputs are always multiplied. C_{out0} and C_{out1} only convert back to sign-magnitude if the next stage will have a multiplication. Therefore, three control signals (C_{in}, C_{out0}, and C_{out1}) have to be generated for each butterfly.



(a) Delay-optimized conversion to two's complement.

(b) Rounding and conditional conversion.

Figure 6: Detailed view of conversion to two's complement, rounding, and conditional conversion to sign-magnitude.

5.3.3 Output Stage Adders. Instead of using a 62-bit CLA after multiplication, we use CSAs for the most significant 32 bits and a 30-bit CLA for the lower bits as shown in Fig. 6(a). The MSB of the this CLA is the fraction bit F that is used for unbiased rounding. In our proposed implementation, the +1 adders of the converters after the multipliers are replaced with CSAs as well. Incorporating the +1 addition into the CSAs shortens the critical path.

5.3.4 Rounding and Conversion. Both the rounding and conditional converter blocks contain a conditional +1 adder, and the activation of both adders is mutually exclusive. Therefore one adder can be removed, which is shown in Fig. 6(b), where C_{out0} is the conditional conversion signal.

6 EVALUATION

In this section we evaluate the two's complement reference and our proposed implementation using synthetic data and a full frame from an actual FMCW radar. The reference two's complement design uses the modified Booth multiplier as proposed by Kuang, Wang, and Guo in [7], and an output stage with Carry-Save (half)adders similar to our proposed design as described in Section 5.3.3. We demonstrate that our proposed design has a significantly lower power usage. Both the reference design and our proposed design use Q2.30 fixed-point numbers, 12-bit input samples, and divide the output by two every two stages, effectively performing $\sqrt{N}$ division on the final results of the FFT. This limits the dynamic range of processed data by keeping the noise amplitude constant, and keeping the growth of signals limited to $\sqrt{N}$.

We have synthesized both implementations using the TSMC 40nm LP standard cell library and Synopsys tools with *high* synthesis and mapping effort. A typical-typical corner was used with a supply voltage of $V_{dd} = 1.1$ V.

Furthermore, we wrote software implementations that model both of the FFT designs, and have verified the correctness of our

Table 1: Synthesis results for circuit area and maximum frequency.

Butterfly Implementations	Max. Freq. MHz (Δ)	Area @ Max. Freq. μm^2 (Δ)	Area @ 500 MHz μm^2 (Δ)
Reference (2C Booth)	714.3 (-)	43532 (-)	33420 (-)
Proposed (SM Booth)	666.7 (-6.67%)	64815 (+48.89%)	45356 (+35.72%)

software FFTs with the built-in FFT implementation in Matlab. Our software FFTs also generate files for stimulus and expected values which was then used by the test bench to verify that our hardware implementations produce the exact same output, and that the results are correct.

Both implementations were tested with two synthetic input signals, by applying a 1024-bin FFT using a single radix-2 butterfly unit in a sequential manner. Both signals contain two bits of noise with normal distribution:

(1) *Weak*: sine wave with amplitude $\frac{1}{4096} \approx 2.44 \cdot 10^{-4}$.
(2) *Strong*: sine wave with amplitude $\frac{4000}{4096} \approx 9.7 \cdot 10^{-1}$.

All implementations were also tested with a full radar frame from an actual FMCW radar by applying a 2D FFT to obtain a range-Doppler frame. The input consists of 512 chirps of 1024 samples each, and therefore the range FFTs perform 512 1024-bin FFTs. Since the input is a real-valued signal, the upper half of range FFT results were discarded because of symmetry, meaning that the Doppler FFTs consist of 512 512-bin FFTs.

6.1 Results

The synthesis results are shown in Table 1. In terms of maximum clock frequency, our proposed design is 6.67% slower than the reference design. There are two reasons that cause the lower maximum clock frequency. First, the Wallace tree in our proposed unsigned Booth multiplier has two more levels compared to the modified Booth multiplier in the reference design. Second, the multiplier results have to be converted to two's complement. This requires one level of XOR gates and one level of half adders to incorporate the +1 addition that is necessary for conversion after multiplication.

Our proposed design typically is 35.72% larger compared to the reference design, and 48.89% larger when both designs are synthesized at maximum clock frequency. This is mostly due to our unsigned Booth multiplier having more partial products, a larger Wallace tree, and hardware for converting to two's complement. However, in a fully integrated CMOS chip where the analog front-end is much larger, this increase in area is typically not an issue.

The power results for both designs using synthetic data can be seen in Fig. 7. Compared to the reference design using only two's complement numbers, our proposed design shows a substantial decrease of 58.18% and 57.18% in power usage for our *weak* and *strong* synthetic inputs respectively. Fig. 8 shows power results for both designs when processing a real radar frame. Our proposed design shows a significant decrease of 40.84% and 46.45% in power usage for the range and Doppler inputs when performing a 2D FFT.

Fig. 9 shows the power usage of just the multipliers when processing the same radar frame. Our proposed unsigned booth multiplier shows a substantial decrease in power usage of up to 66.27%. The difference in power usage can be mostly attributed to the higher bits of the modified Booth multiplier having a much higher switching activity compared to our proposed unsigned Booth multiplier.

To demonstrate the source of this difference in power usage, we simulated both designs with uniformly distributed 12-bits and 32-bits of random data instead of sinusoidal signals, and the results are shown in Fig. 10. Since the FFTs process uncorrelated data and we use $\sqrt{N}$ scaling, the signal power will be smeared out over all bins, and the dynamic range of the intermediate and output data will more or less stay the same as that of the input. Therefore the multipliers will process values with the same dynamic range in every stage.

When given 12-bits of random data, the multipliers of our proposed design use up to 42.68% less power than those of the reference design, despite our proposed multiplier generating almost double the amount of partial products. The only other source that can cause a higher switching activity in the multipliers is that of the higher bits when using two's complement numbers. This effect is clearly reflected in the power results when 32-bits of random data is given to both designs. In this scenario, the multipliers of our proposed design now use up to 35.41% more power than the multipliers of the reference design.

When comparing each multiplier with 12-bits and 32-bits random data, then our proposed unsigned Booth multiplier uses between 88.00% and 114.31% more power with 32-bits random data compared to 12-bits random data. For the modified Booth multiplier this difference is between 1.53% and 3.15%. These results confirm the overwhelming sparsity of the internal values and outputs of FFTs when processing FMCW radar signals.

7 CONCLUSION

In this paper we have proposed a low-power FFT implementation for FMCW radars. Our proposed, hybrid design uses sign-magnitude encoding for multiplication and two's complement encoding for addition and subtraction. This design uses our proposed, unsigned Booth multiplier that does not generate negative numbers internally.

Evaluation shows that our proposed design uses up to 46.45% less power than a comparable two's complement implementation when processing data from a real FMCW radar. Area and maximum clock frequency wise, the design is 35.72% larger and only 6.67% slower. The results clearly demonstrate the significant benefits of using sign-magnitude numbers in FFTs for FMCW radar systems in terms of power-efficiency, with a very small increase in the critical path.

ACKNOWLEDGMENTS

This work is part of the research program Perspectief ZERO with project number P15-06 Project 3, which is (partly) financed by the Dutch Research Council (NWO).

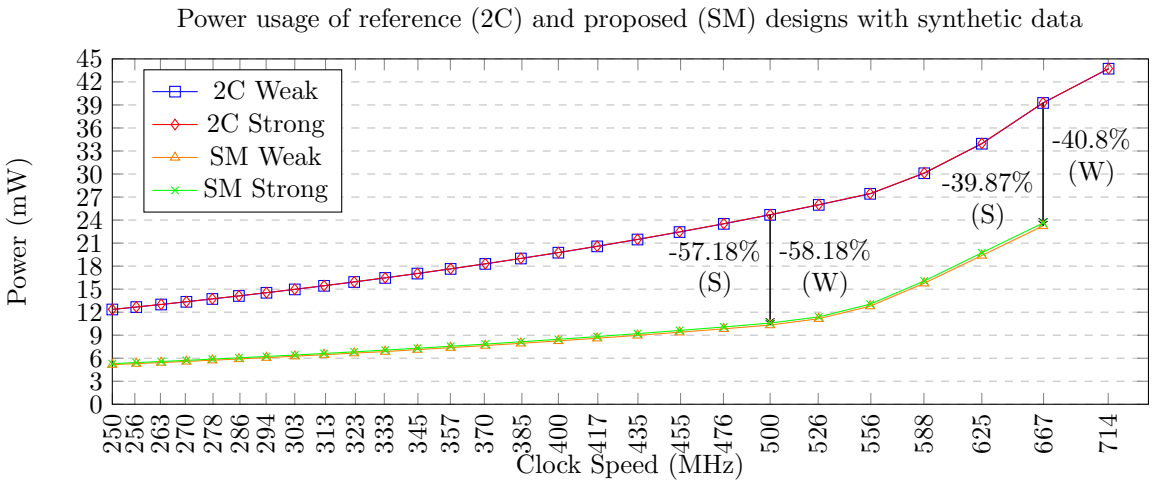


Figure 7: Power usage of reference (2C) and proposed (SM) designs using synthetic data. The (W) and (S) indicators denote the power usage decrease percentage for weak and strong signals respectively.

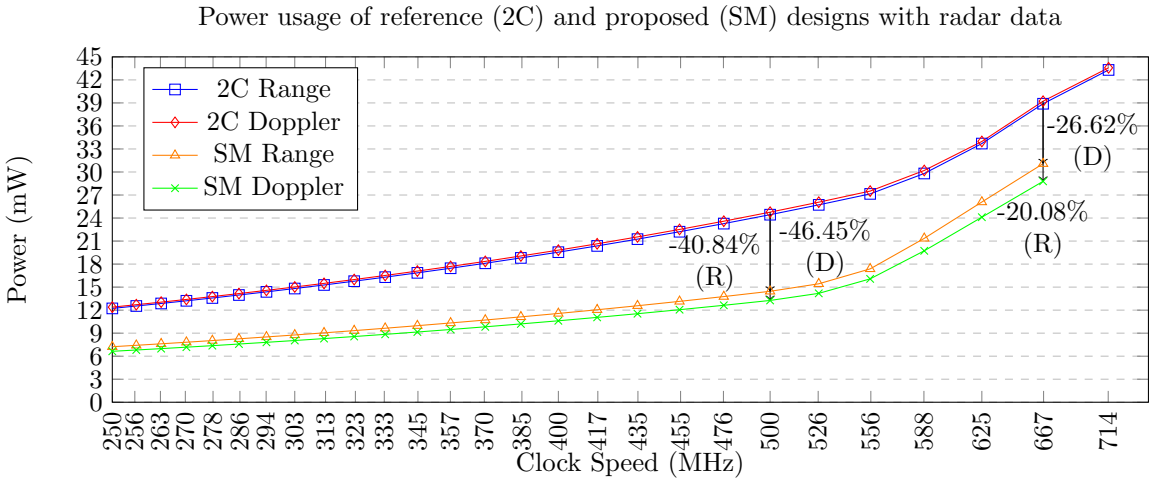


Figure 8: Power usage of reference (2C) and proposed (SM) designs using a real radar frame. The (R) and (D) indicators denote the power usage decrease percentage for range and Doppler data respectively.

REFERENCES

[1] Charles R. Baugh and Bruce A. Wooley. 1973. A Two's Complement Parallel Array Multiplication Algorithm. *IEEE Trans. Computers* C-22, 12 (Dec 1973), 1045–1047.

[2] Gary Wayne Bewick. 1994. *Fast Multiplication: Algorithms and Implementation.* Ph.D. Dissertation. Stanford, CA, USA.

[3] Anantha P. Chandrakasan and Robert W. Brodersen. 1995. *Low Power Digital CMOS Design.* Kluwer Academic Publishers.

[4] James Cooley and John Tukey. 1965. An Algorithm for the Machine Calculation of Complex Fourier Series. *Math. Comp.* 19, 90 (1965), 297–301.

[5] W. Morven Gentleman and G. Sande. 1966. Fast Fourier Transforms: For Fun and Profit. In *Proceedings of the November 7-10, 1966, Fall Joint Computer Conference* (San Francisco, California) *(AFIPS '66 (Fall)).* Association for Computing Machinery, New York, NY, USA, 563–578. https://doi.org/10.1145/1464291.1464352

[6] Simon Kingsley and Shaun Quegan. 1999. *Understanding Radar Systems.* SciTech Publishing, 11.

[7] Shiann-Rong Kuang, Jiun-Ping Wang, and Cang-Yuan Guo. 2009. Modified Booth Multipliers With a Regular Partial Product Array. *IEEE Transactions on Circuits and Systems II: Express Briefs* 56, 5 (2009), 404–408.

[8] Richard G. Lyons. 2011. *Understanding digital signal processing, 3/E.* Pearson Education India.

[9] Olin Lowe Macsorley. 1961. High-Speed Arithmetic in Binary Computers. *Proc. IRE* 49, 1 (Jan 1961), 67–91.

[10] Chetana Nagendra, Robert Michael Owens, and Mary Jane Irwin. 1995. Unifying Carry-Sum and Signed-Digit Number Representations for Low Power. In *Proc. ISLPED.* ACM, 15–20.

[11] Gustavo A Ruiz and Mercedes Granda. 2008. Efficient implementation of 3X for radix-8 encoding. *Microelectronics Journal* 39, 1 (2008), 152–159.

[12] Mohsen Saneei, Ali Afzali-Kusha, and Zainalabedin Navabi. 2005. Sign bit reduction encoding for low power applications. In *Proc. DAC.* 214–217.

[13] Kelly Liew Suet Swee and Lo Hai Hiung. 2012. Performance comparison review of radix-based multiplier designs. In *ICIAS2012,* Vol. 2. IEEE, 854–859.

[14] Yibin Ye, Kaushik Roy, and Rolf Drechsler. 1999. Power consumption in XOR-based circuits. In *Proc. ASP-DAC.* 299–302 vol.1.

[15] Zhan Yu, Meng-Lin Yu, Kamran Azadet, and Alan N. Willson. 2001. A low power FIR filter design technique using dynamic reduced signal representation. In *2001 VLSI-TSA. Proc. of Technical Papers.* 113–116.

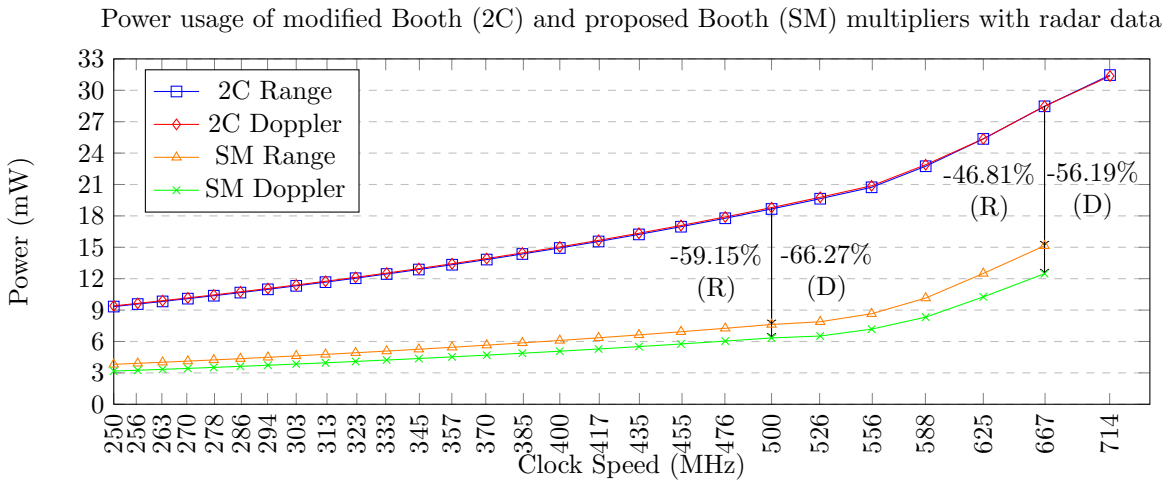


Figure 9: Power usage of the multipliers in the reference (2C) and proposed (SM) designs using a real radar frame. The (R) and (D) indicators denote the power usage decrease percentage for range and Doppler data respectively.

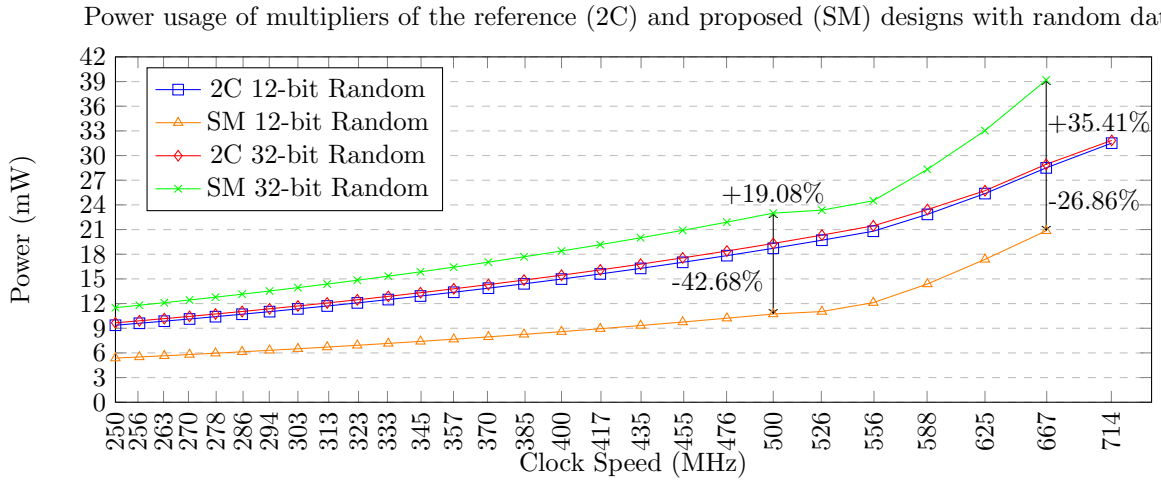


Figure 10: Power usage of the multipliers in the reference (2C) and proposed (SM) designs when processing random input. Percentages indicate the decrease or increase of power usage of our proposed multiplier compared to the reference multiplier.

[16] Menghui Zheng and Alexander Albicki. 1995. Low power and high speed multiplication design through mixed number representations. In *Proc. ICCD VLSI in Computers and Processors*. 566–570.