Enabling Software-defined PHY for Backscatter Networks

Fengyuan Zhu
Shanghai Jiao Tong University
jsqdzhufengyuan@sjtu.edu.cn

Mingwei Ouyang
Shanghai Jiao Tong University
1999mrou@sjtu.edu.cn

Luwei Feng
Shanghai Jiao Tong University
yundanfengqing@sjtu.edu.cn

Yaoyu Liu
Shanghai Jiao Tong University
lyyu19@sjtu.edu.cn

Xiaohua Tian*
Shanghai Jiao Tong University
xtian@sjtu.edu.cn

Meng Jin
Shanghai Jiao Tong University
jinm@sjtu.edu.cn

Dongyao Chen
Shanghai Jiao Tong University
chendy@sjtu.edu.cn

Xinbing Wang
Shanghai Jiao Tong University
xwang8@sjtu.edu.cn

ABSTRACT

In this paper, we for the first time show how to enable software-defined PHY (SD-PHY) to achieve agile reprogrammability in wireless backscatter networks. This can facilitate innovations in this field by relieving researchers from unnecessary engineering work. With SD-PHY, the tag's PHY-layer behavior can be neatly defined by configuring a set of parameters, which allows the common hardware to generate backscattered signals complying with various wireless protocols. The SD-PHY architecture is based on the key insight that the tag's PHY-layer behavior is essentially determined by reflection coefficient sequence.

The SD-PHY is factually to instruct the hardware: How to generate various reflection coefficient sequences that meet different protocols' requirements at the right time; under what clock rate to feed the generated sequence into RF switches on the tag. We abstract such instructions into a set of parameters. To make different parameter values take effect in the runtime, innovative designs of the generic wake-up receiver, baseband modulator, and clock signal generator on the tag, which are responsible for executing those parameterized instructions. We design and implement a general hardware platform to support the SD-PHY software. Moreover, we demonstrate that under the unified SD-PHY framework how the same tag can generate different kinds of backscatter signals, which could obey standardized protocols such as Wi-Fi (11b/g), BLE, LoRa, and LTE, as well as highly customized protocols such as OFDMA backscatter and NetScatter. Experimental results show that the system presents similar performance no matter it is realized with universal SD-PHY or a dedicated design approach.

*Corresponding author

MobiSys '22, June 25-July 1, 2022, Portland, OR, USA

ACM ISBN 978-1-4503-9185-6/22/06…$15.00
https://doi.org/10.1145/3498361.3538927

CCS CONCEPTS

• **Networks** → **Network design principles**; **Network architectures**;

KEYWORDS

Software-defined, Backscatter, PHY

ACM Reference Format:
Fengyuan Zhu, Mingwei Ouyang, Luwei Feng, Yaoyu Liu, Xiaohua Tian, Meng Jin, Dongyao Chen, and Xinbing Wang. 2022. Enabling Software-defined PHY for Backscatter Networks. In *The 20th Annual International Conference on Mobile Systems, Applications and Services (MobiSys '22), June 25-July 1, 2022, Portland, OR, USA.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3498361.3538927

1 INTRODUCTION

The past decade has seen numerous backscatter networking techniques proposed, which provide μW-level IoT connectivity while complying with a variety of wireless protocols [1–8]. However, obstacles still remain in the path of existing backscatter systems toward practical usage, issues include but are not limited to the inability to work with COTS receivers, support longer communication range, higher data rate, and concurrency [9–14]. Looking back into those existing work is necessary for this research area to move forward, where the inevitable process is to reproduce both hardware and software of previously proposed backscatter systems for comparison. The engineering efforts incurred however hinder researchers from quickly verifying new ideas about backscatter communications.

The labor work can be significantly mitigated if the backscatter tag can be quickly reprogrammed. Imagine that the tag's PHY layer behavior can be software defined with only writing a few parameters in the processor, just like calling a program function, and then the corresponding baseband design could take effect. All we need to do is calculating the values of these parameters according to the backscatter modulation process desired.

In this paper, we propose software-defined PHY (SD-PHY), a universal design approach that can achieve the goal above. The essence of the backscatter communication is that the tag modulates the excitation signal through changing the impedance. The choice of impedance determines the reflection coefficient of the antenna, which realizes backscattering. Our key insight in this paper is that the PHY-layer behavior of the tag can be defined by the sequence

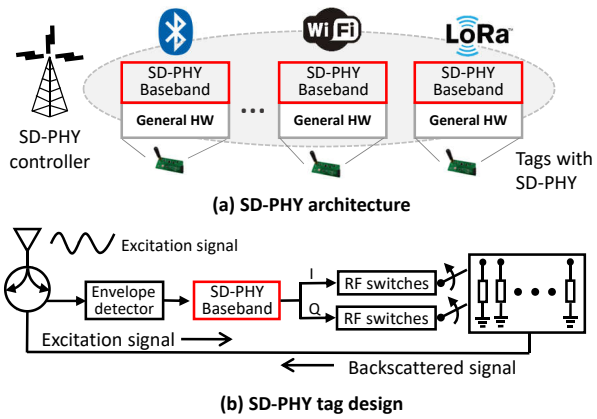

Figure 1: SD-PHY for backscatter.

$\{\Gamma_1, \Gamma_2, ...\Gamma_i, ...\Gamma_n\}$, where Γ_i denotes the reflection coefficient provided by the hardware. Each of the elements in the sequence Γ_i corresponds to a change in the amplitude, phase, or frequency of the excitation signal. Switching the tag circuit's load impedance according to a certain sequence produces corresponding waveforms of the backscatter signal. Each of the existing backscatter network designs factually uses the fixed Γ sequence to generate predefined backscatter signals.

The SD-PHY enables reprogrammability of backscatter networks through software defining the Γ sequence. As shown in Fig. 1, we abstract the sequence computing functionality as the *SD-PHY baseband*, which is independent of the general hardware providing the impedance. The SD-PHY controller remotely manipulates the SD-PHY baseband so that the tag could generate appropriate backscatter signals according to designated wireless protocols as instructed. We manage to let the SD-PHY baseband produce desired reflection coefficient sequence through a set of PHY control parameters, which avoids the process of burning binary files into an expensive FPGA chip after mass production. This key advantage further paves the path for reconfigurable backscatter ASIC, which fits heterogeneous industrial IoT applications.

Materializing the architecture as shown in Fig. 1 requires the backscatter system to own new capabilities under the unified PHY-layer design: 1) Capable of recognizing various excitation signals from different kinds of RF sources so that the tag could generate right Γ sequence at right time; 2) able to produce various Γ sequences different in lengths and elements, in order to accommodate different protocols; 3) could feed various Γ sequence into RF switches in different clock rates according to sampling rates of corresponding standards; 4) capable of controlling the tag's PHY-layer behavior with respect to 1)-3) by only configuring a few parameters, in order to achieve agile reprogrammability.

In addressing the issues above, we make following **technical contributions:**

- We present a parameterized design of SD-PHY for backscatter networks (**§Section 3**). The core of the SD-PHY is the SD-PHY baseband, which contains the generic wake-up receiver, baseband modulator, and clock signal generator. Such components are innovatively designed to obtain the new capabilities required for reprogramming. We abstract the PHY control scheme into a set of parameters stored in a centralized control plane, which enables remotely controlling waveform generation in the tag with agility.
- We show how to realize representative backscatter systems with SD-PHY design approach. We illustrate how to configure those control-plane parameters, in order to make the same tag capable of generating backscatter signals complying with standardized protocols such as Wi-Fi (11b/g), BLE, LoRa, and LTE under the unified SD-PHY framework. We also show that the SD-PHY could realize highly customized backscatter systems including OFDMA backscatter [15] and NetScatter [16] (**§Section 5**).
- We design and implement a prototype of the SD-PHY tag, with which we conduct experiments to evaluate performance of the backscatter system under different configurations (**§Section 6**). We test the system with respect to communication throughput/range and power consumption. Experimental results show that the SD-PHY tag presents the same or close performance compared with the dedicated designs in terms of data rate and distance.

Platform availability. The SD-PHY hardware platform, including schematics, layout, and the software is released for academic use, and available online [17].

2 PRELIMINARIES

2.1 Primer for Backscatter Networks

The backscatter network normally consists of three components: the excitation signal transmitter, tags, and the receiver. Existing designs can be classified according to whether the excitation signal transmitter could be the legacy infrastructure or must be the dedicated plug-in device, which are termed as *compatible* and *dedicated* backscatter networks respectively in the remainder of the paper.

Compatible backscatter networks such as ambient backscatter [18], Wi-Fi backscatter [1], and BackFi [19] leverage ambient RF signals from TV tower or Wi-Fi AP as the source of power and carrier signal, which however suffer from self-interference thus results in very limited data rate and communication range. HitchHike [3] presents a novel idea of *codeword translation* to resolve the self-interference issue, which is further developed by FreeRider [4]. The translation is realized by switching the phase of the excitation signal on the tag, where the modulation is carried out by the tag's performing translation or not. The backscattered signal is then moved to a channel that is other than the excitation signal's channel but still in the range of the receiver. The receiver decodes the message conveyed from the tag by XORing the productive excitation signal from the AP and backscattered signal from the tag with both following 802.11b. FreeRider streamlines the codeword translation technique to support the OFDM Wi-Fi, Bluetooth, and ZigBee, where the codeword translation is realized by modifying amplitude, phase and frequency of the excitation signal.

Dedicated backscatter networks such as passive Wi-Fi [2], NetScatter [16], OFDMA backscatter and DigiScatter [20] adopt the dedicated plug-in RF source, where the transmitted excitation signal mainly consists of the single tone. Such systems directly generate desired wireless signals through manipulating and reflecting the single tone, which provides higher flexibility to realize more complicated backscatter networking schemes. For example, NetScatter

tag could directly generate chirp signals with different frequency shifts, and OFDMA backscatter manages to achieve strict time and frequency domain synchronization through directly generating OFDM subcarriers in tags. In the dedicated backscatter network, the modulation operation in the tag is more like the traditional wireless transmitter, where the single tone acts as the pure carrier signal to be modulated by the local baseband signal. The modulation process also yields a frequency shift of the backscatter signal thus avoiding self-interference with the excitation signal.

The compatible system is more convenient for deployment, which leverages legacy infrastructure; the dedicated system provides higher flexibility, which makes it possible for the backscatter network to act more like the traditional network while maintaining ultra-low level power consumption. Moreover, existing designs present diverse capabilities in communication throughput/range as summarized in [12], which suit various scenarios. It is hard to pick anyone from those existing backscatter systems fitting all the scenarios. Practical IoT connectivities are usually heterogeneous [21], thus numerous and varied tags will coexist. The deployment and maintenance cost will be considerable if tags are designed in an ad hoc manner.

2.2 PHY Description

The on-tag operations vary in different backscatter networks to comply with corresponding wireless PHY standards. In particular, the tag of each system must backscatter predefined signal waveform, so that the receiver obeying corresponding PHY standard could successfully decode the information conveyed. However, we are to present a general paradigm below, showing that the essence for controlling the behavior of the tag in all the backscatter networks is to control how the baseband signal is generated.

The following equation can be used to describe how the ideal backscatter signal is generated:

$$S_{recv} = S_{base} \cdot S_{src},$$

where S_{recv}, S_{base} and S_{src} denote the received signal at the receiver, the baseband signal at the tag and the RF source signal from the excitation signal transmitter, respectively. For any specific backscatter network, the excitation signal S_{src} is given, and the core of the system is to appropriately generate S_{base} on the tag, so that the backscatter signal is equal to the desired S_{recv}. This indicates that as long as we could software define S_{base}, we are able to software define the behavior of the tag to accommodate different standards. That is

$$S_{base} = \frac{S_{recv}}{S_{src}}.$$

Take LoRa backscatter and NetScatter for example, the desired received signal and the dedicated excitation signal are as shown in (1), which are the chirp signal and the pure tone, respectively. Then the baseband signal on the tag is supposed to be as shown in (2), which is a chirp signal with a frequency shift Δf as mentioned in Section 2.1.

$$\begin{cases} lS_{recv\_CSS} = cos[2\pi(f_c + \Delta f)t] \cdot e^{j2\pi(a_2t^2 + a_1t)}, \\ S_{src\_CSS} = cos(2\pi f_c t) \cdot [1, 1, \ldots 1]. \end{cases} \quad (1)$$

$$S_{base\_CSS} = cos(2\pi\Delta f t) \cdot e^{j2\pi(a_2t^2 + a_1t)}. \quad (2)$$

We could always figure out what baseband signal the tag is supposed to generate as long as the PHY-layer protocol is given. If we could manipulate the baseband signal generation, we will be able to switch the backscatter network's working protocols, for example, switching from HitchHike to LoRa backscatter only requires the tag to generate a different kind of baseband signal. While in quite different forms, after quantization process in the tag, the baseband signal is produced by switching among different impedance provided by the tag circuit. The baseband signal generation process is eventually transformed into the sequence generation, and the behavior of the tag is essentially determined by the reflection coefficient sequence in the form of $\{\Gamma_1, \Gamma_2, \ldots\Gamma_i, \ldots,\Gamma_n\}$.

3 DESIGN

3.1 Challenges

The PHY layer of the existing backscatter designs could be described using the Γ sequence. The crux for realizing agile reprogrammability is how to realize re-configurable PHY software for the tag. Ideally, we can simply instruct the tag what Γ sequence to generate to achieve reprogrammability. However, we have to overcome the following challenges before real software-defined PHY could be achieved.

- **C1: When to generate the Γ sequence?** The backscatter system differs from the traditional wireless network in that the reflecting device is triggered by the excitation signal, instead of actively transmitting local information. For compatible backscatter systems, tags need to synchronize with ambient excitation signal. The excitation signal waking up the tag in existing backscatter systems is diverse: HitchHike wakes up the tag with a rising edge of the ambient signal [3] while FreeRider with a special OOK sequence [4]. In order to achieve reprogrammability, the tag must be able to recognize various excitation signals, which enables triggering the tag to generate appropriate Γ sequence under corresponding configuration.
- **C2: How to generate various Γ sequences?** Various backscatter designs need the tag to generate different sequences with various lengths and elements. For example, passive Wi-Fi and HitchHike need the tag to produce the 11-bit Barker codewords, while NetScatter needs the tag to produce a sequence containing 2^{SF} zero or one bits, where SF is the spreading factor for CSS modulation.
- **C3: Under what clock rate to feed Γ sequence into RF switches?** The Γ sequence is factually the XOR result of the shift frequency and the baseband signal after ADC, which needs different clocks to drive. The frequency shifting operation is widely adopted in existing backscatter systems to avoid self-interference, where the value of the frequency shift varies. By default, the frequency shift Δf should satisfy $\Delta f \geq BW$, where BW is the bandwidth of the excitation signal. Similarly, different protocols require different baseband rates. For instance, baseband rate (chip rate) of 802.11b DSSS modulation is 11 MHz, while that for LoRa CSS modulation can be 125kHz/250kHz /500kHz [22], orders lower than DSSS. This means that the tag must be able to feed different kinds of Γ sequences into RF switches in different rates required.
- **C4: Which strings to pull so that the tag could generate desired Γ sequence?** The centralized controller and the tag should agree on how to reprogram the PHY behavior. In particular, there

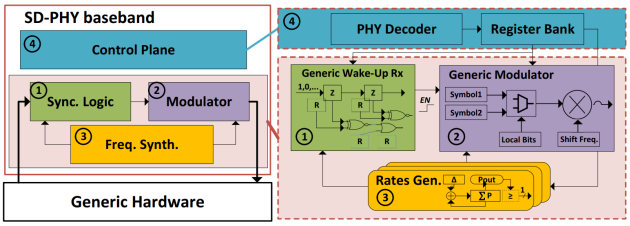

Figure 2: Structure of the SD-PHY baseband.

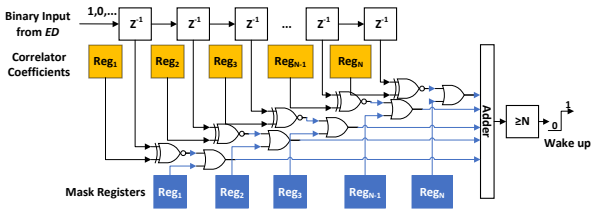

Figure 3: Correlator of the generic wake-up receiver.

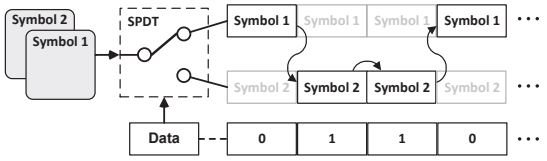

Figure 4: Framework of the generic modulator.

should be an instruction set that is able to tailor PHY characteristics as desired by simple operations such as assigning values to some PHY parameters. This requires abstracting common properties of different backscatter designs to create a new PHY description language.

3.2 SD-PHY Baseband

This section presents the SD-PHY baseband design as shown in Fig. 2, which contains 4 modules with each resolving one of the challenges mentioned above correspondingly. The SD-PHY baseband is the software part over the general hardware of the tag. As shown in the left part of Fig. 2, the synchronization logic is responsible for synchronizing the excitation signal transmitter and the tag to wake up the tag. The modulator module generates the desired baseband signal in the form of the Γ sequence. The frequency synthesizer module generates clocks to drive the two modules mentioned above. The control plane module is for configuring the other 3 modules, so that the entire PHY can be software defined. The key design of the SD-PHY baseband is illustrated in the right part of Fig. 2, which is elaborated below.

3.2.1 Generic Wake-up Receiver. The synchronization logic module is essentially a generic wake-up receiver, which could identify all the constant-envelope excitation signal as used in existing backscatter networks. In particular, when the excitation signal arrives at the tag, the analog envelope detector circuit in the tag hardware will forward the binary envelope samples to a correlator with both coefficients and depth reconfigurable. The generic wake-up receiver consists of the circuit and the correlator, which triggers the streaming-out process of the Γ sequence.

Triggering the tag is factually a cross-correlation process. A widely adopted correlation logic in existing tag designs is the finite-depth digital correlator as shown in the upper part of Fig. 3. We can see that there are N_s one-bit registers for storing the predefined sequence (coefficients), and the sequence abstracted from the envelope detector circuit correspondingly contains N_s bits. Those two fixed-length sequences are correlated; if match, then the tag is triggered and responds to the excitation signal.

After the correlator is implemented in the FPGA, the length of coefficients is fixed and all the coefficients will be involved in the correlation operation, which is unable to support multiple kinds of excitation signals. For example, the correlator only needs to store two coefficients '10' for the triggering rising-edge but needs to store more for the OOK excitation signal such as '100100100'. To support switching among the two kinds of excitation signals, the correlator should have 9 registers at least, so that the longer one can be stored. If switching to the '10' case, the two bits have to be stored in those 9 registers, but bits to be stored in the rest of the 7 registers are not defined. In this case, the correlator may not be able to recognize the triggering rising-edge, because the FPGA (or ASIC) is unable to eliminate the rest 7 registers on site and add them back when needed next time.

To address the issue, we design a **mask logic** to control the correlation depth as shown in the lower part of Fig. 3. The logic gates in blue can bypass the XNOR result if the mask register stores '1', and the XNOR result will be unaffected if the mask register stores '0'. By manipulating the values stored in mask registers, the actual correlation length and depth can be tailored. Continue the example, when switching to the '10' case, we could apply a mask sequence '001111111' meaning that the correlation depth is 2, and all the correlation taps after are bypassed. Then we could elegantly set the correlation registers as '10*******' where '*' means "do not care".

3.2.2 Generic Baseband Modulator. Recall the analysis in Section 2.2, the Γ sequence is factually the XOR result of the baseband signal and the frequency shifting signal after quantization. This section presents our generic baseband modulator design, and the frequency shifting signal sequence generation is to be explained in Section 3.2.3.

The length and elements of the baseband signal sequence are diverse, which is determined by the modulation scheme adopted by the system. The generic baseband modulator should be able to generate all kinds of baseband signal sequences to support reprogrammability. This could be naively realized by pre-storing all possible baseband signal sequences in the tag. Figure 4 shows a 2-symbol case. In particular, the modulated wireless signal is factually made up of a limited number of symbols such as the barker code in 11b. We only need to pre-store those symbols following the Nyquist sampling rate, and the modulated signal sequence given the local data stream could be generated by combining corresponding symbols. Both the tag side and the receiver side will have a symbol codebook, which contains S_{recv} and S_{base} respectively. As long as the symbol S_{base} can be discretely quantized by Γ values, tags can switch to run any modulation scheme. In this manner, the modulation scheme of the backscatter signal can be software defined.

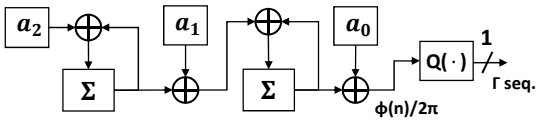

Figure 5: Computing $\phi(n)$ and Γ sequence.

Challenge: How to produce PSK, FSK and even CSS signals in a both unified and memory-efficient way? While the symbol pre-storage mode is simple and effective for realizing modulated signals containing limited samples per symbol (SPS), it confronts the challenge when dealing with the low-rate wide-area wireless protocol. In particular, the modulated signal in LoRa is the chirp signal containing 2^{SF} samples per symbol, with SF could vary between 7 and 12. This means that we will have to store 2^{12} samples for a single symbol in the worst case, which consumes too much memory resource. Similar problem occurs when we want to generate an over sampled FSK or PSK signal.

We find that the modulation schemes mentioned above all produce constant-envelope signals, which can be expressed in the following form:

$$BB(n) = sign(cos(\phi(n))),$$

where the phase term can be expressed with Taylor's 2^{nd}-order expansion as following:

$$\phi(n) = 2\pi \cdot (\frac{1}{2}a_2 n^2 + a_1 n + a_0), \tag{3}$$

$$BB(n) = sign(cos(2\pi(\frac{1}{2}a_2 n^2 + a_1 n + a_0))). \tag{4}$$

We have n start from 0 to $N_s - 1$ with N_s the number of samples contained in a symbol.

Insight : The 2^{nd}-order expansion already realizes PSK, FSK and CSS with only four parameters. 1) To realize PSK: we set parameter $a_2 = 0$ and $a_1 = 0$, then we embed payload on a_0, e.g., $a_0 = \frac{0}{2\pi}$ and $a_0 = \frac{\pi}{2\pi}$. 2) To realize FSK: we set $a_2 = 0$ and leave a_0 fixed, then we embed payload on a_1. 3) To realize CSS: we set a_2 to be the chirp rate, a_1 to be the starting frequency and a_0 fixed. Parameter N_s can be configured according to the symbol rate. Then symbols containing a large number of samples can be described using parameters a_2, a_1, a_0, and N_0, thus we only need 4 registers to store those symbols and the storage resource required is significantly mitigated.

However, directly applying Eq. (4) will incur unaffordable power consumption due to the use of multipliers. To address this problem, we replace multipliers with adders by improving the digital circuit design, which empirically could save 80 − 90% power consumption. To convert multiplication to addition, we differentiate Eq. (4) as follows.

$$\phi(n) - \phi(n-1) = 2\pi \cdot (\frac{1}{2}a_2(2n-1) + a_1), \tag{5}$$

$$(\phi(n) - \phi(n-1)) - (\phi(n-1) - \phi(n-2)) = a_2. \tag{6}$$

By twice differentiating, we eliminate multipliers. The final structure for computing phase $\phi(n)$ is shown in Fig. 5. The first part of the modulator calculates the total frequency at a specific sampling point based on Eq. (6). The second part calculates the total phase using Eq. (5). The third part performs output waveform value decision based on previously obtained total phase.

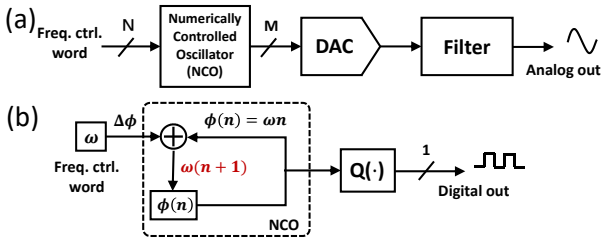

Figure 6: Rate generation module. (a) Traditional DDS. (b) Proposed low-power digital frequency synthesizer.

We now show how to calculate the exact values of parameters a_2, a_1 and a_0 using the chirp signal generation as an example. Other schemes such as PSK and FSK can share the criteria by configuring $a_2 = 0$. We could substitute the sample index n into Eq. (3) with $n = \frac{t}{T_s} = tf_s$, then we have:

$$\phi(t) = 2\pi \cdot (\frac{1}{2}a_2 f_s^2 t^2 + a_1 f_s t + a_0),$$

where the starting frequency and rate of the chirp are:

$$\frac{1}{2\pi}(\frac{d\phi}{dt})_{t=0} = a_1 f_s, \tag{7}$$

$$\frac{1}{2\pi}(\frac{d^2\phi}{dt^2}) = a_2 f_s^2, \tag{8}$$

respectively. Those parameters are expressed in 16-bit fixed-point numbers, with values can be calculated as following:

$$a_0 = round(\phi_0/2\pi \cdot 2^{16}), \tag{9}$$

$$a_1 = round(f_0/f_{uplink} \cdot 2^{16}), \tag{10}$$

$$a_2 = round(ChirpRate \cdot 1/f_{uplink}^2 \cdot 2^{16}). \tag{11}$$

With the first kind of symbol described with a_0, a_1, and a_2, we can calculate the second kind of symbol in the same way, which could be described with b_0, b_1, and b_2. Then we can obtain a code book containing two parameterized symbols.

Remarks: Note that the proposed design based on Taylor's expansion is not limited to CSS modulation. Since the coefficients contain the first order and the constant value, it can also support FSK/PSK modulation. Note that Taylor's expansion brings zero precision loss to the symbol generation if we could store and compute infinitely precise parameter numbers. This is because FSK/PSK/CSS symbols contain no high-order (> 2) terms in definition. The precision loss caused by fixed-point number will be evaluated in Section 6.

We could adopt a hybrid method for generating the baseband waveform sequence. That is, the controller could choose between the pre-storage mode or the Taylor's expansion mode. This is because we use fixed-point numbers to represent $\phi(t)$ coefficients in logic circuits. To achieve high precision, a 16-bit word length is usually needed for each coefficient. Then if there are fewer than $16 \times 3 = 48$ samples per symbol like DSSS/CCK, the first mode can be more efficient.

3.2.3 Generic Clock Rates Generation. The generic clock rates generation scheme is responsible for synthesizing multiple clock rates in order to drive the correlator and the baseband modulator as

shown in Fig. 2. In order to support reprogrammability, the generator is supposed to support a wide tuning range and fine-grained frequency resolution to accommodate diverse requirements.

An analog way to realize clock rate re-adaptation is to reconfigure divider words and reboot the phase-locked loop (PLL) module in FPGA. A PLL module would synthesize the desired frequency via generating a much higher frequency using the voltage-controlled oscillator (VCO) and then divide it by an integer divider. Since the output signal would be compared with an accurate crystal unit and there exists negative feedback to keep the output frequency on track, the output is usually very stable and accurate. However, using PLL in the scenario under study would encounter two issues as following: **1) Non-trivial configuration overheads.** Output frequency of the PLL is configured via changing the dividers in the loop. However, too many possible combinations exist and complicated rules should be followed when choosing divider values. Commercial PLLs are typically configured with the assistance of EDA tools [23, 24]. This is unsuitable for quickly configuring many tags. **2) High VCO frequency results in high power consumption.** A PLL first generates a high-frequency clock that is integer multiple of the reference clock in the VCO, and then divides it with an integer to reach the target frequency. The high VCO frequency leads to high power consumption even if the target output frequency is low.

Solution: We design a low-power and easy-to-configure digital frequency synthesizer to produce configurable clocks. Our design borrows the idea of direct digital synthesis (DDS) as shown in Figure 6(a). DDS is a hybrid digital-analog frequency synthesis method that can generate arbitrary waveform without using the VCO or PLL. To reduce power consumption, we make a purely digital design of the synthesizer as shown in Fig. 6(b), avoid using DAC and filter. Our key observation here is that we do not need to generate an analog sinusoid like a function generator. Instead, we only need to produce a square wave that can drive a digital circuit. Thus, we could replace the DAC with a quantization logic outputting a square wave. We set the frequency control registers to have N_s bits, with N_s equals 16 for the backscatter baseband and 10 for WuRX. Larger N_s can bring higher frequency resolution $\Delta f = \frac{f_s}{2^N}$ while linearly increases the power consumption since more registers are used in IC. Take uplink clock generation for example, we use a $40MHz$ reference clock and control the output clock rate by writing different rate control words. The rate value can be obtained as:

$$rate\ value = round(f/40MHz \cdot 2^{16}) \tag{12}$$

3.2.4 Parameterized PHY Control. The PHY control scheme involves the SD-PHY controller and the SD-PHY tags. The controller sends control messages to the tag, which are processed by the control plane as shown in Fig. 2. The control plane stores all necessary parameters that could completely describe the behavior of the PHY, including the wake-up receiver registers, baseband modulator coefficients, and rate control words. Such parameters are physically stored in FPGA registers to form a register bank. The controller could dynamically change values in the register bank to configure PHY. In particular, the register bank is directly interfaced by a PHY decoder, which will write new PHY parameters after the configuration frame is validated. All the necessary parameters for PHY

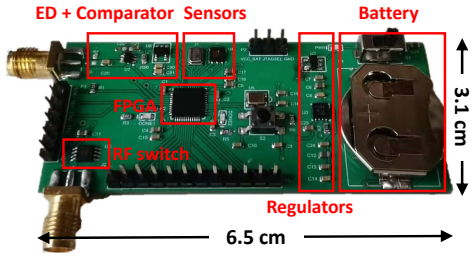

Figure 7: SD-PHY hardware prototype.

configuration are tabulated in Table 1. Each parameter is uniquely identified by a 4-bit command type.

In particular, "Mode" is to instruct the tag's modulator working in the symbol pre-storage mode or the Taylor's expansion mode as described in Section 3.2.2. $Symbol_1$ and $Symbol_2$ are the symbols to be stored by the modulator for the former mode, while $\{a_i\}$ and $\{b_i\}$ are polynomial coefficients describing the two symbols for the latter mode. The "$Shift\_freq_{1,2}$" specifies two frequency shift values for the two symbols, respectively. "Synch. pattern, mask" specifies the triggering signal for the tag. "Sync. clk rate" and "Uplink rate" instruct the generic clock generator how to generate the clock signal for downlink synchronization and uplink backscattering communication, respectively. Note that "N_s" specifies the number of samples in a symbol regardless the mode chosen, thus the symbol duration can be configured with N_s and "Uplink rate", i.e., symbol duration equals $\frac{N_s}{Uplinkrate}$. "Reset ID" is used to write a new ID into the tag.

Table 1: PHY parameters.

Parameter	Cmd. type	Bits
Mode	0111	1
$Symbol_1$, $Symbol_2$	1100 - 1101	64, 64
a_0, a_1, a_2	0100 - 0110	16, 16, 16
b_0, b_1, b_2	0001 - 0011	16, 16, 16
N_s	1111	10
$Shift\_freq_{1,2}$	0000	16, 16
Sync. pattern, mask	1001	16, 16
Sync. clk rate	1010	10
Uplink rate	1011	16
Reset ID	1110	16

We now present a go-through example to show how to configure PHY with those parameters. Suppose that the controller intends to configure two tags to work in the passive Wi-Fi mode. After setting the central frequency of the excitation signal, say $2.412GHz$, we could assign values to those parameters tabulated in Table 1 as follows. 1) Consider the Γ sequence in this case, which consists of two aspects: a single tone signal with shift frequency and the baseband chips. To avoid self-interference, the shift frequency is set as $20MHz$ using command type '0000' with the value $fi(2^{15}, 0, 16, 0)$ [1] according to Eq. 12. For the baseband Γ sequence itself, since 802.11b DSSS modulation adopts the Barker code, there are two kinds of symbols need to be stored, '10110111000' and '01001000111'.

[1] $fi(v, s, w, f)$ represents a fixed-point number with the value v, signedness s, word length w and fraction length f [25].

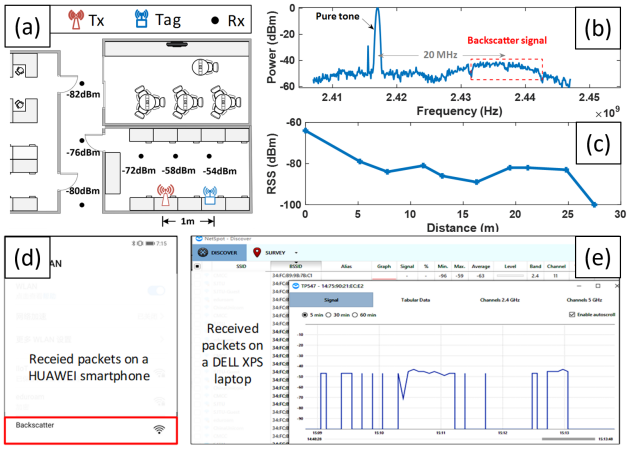

Figure 8: SD-PHY realizing passive Wi-Fi.

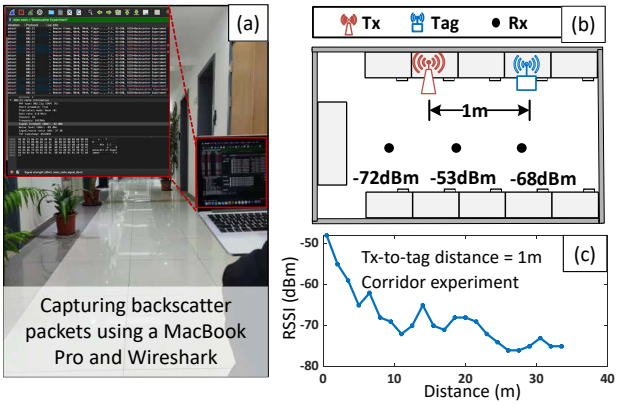

Figure 9: SD-PHY realizing codeword translations.

As the length of each symbol (11 bits) is short, we could use the pre-storage mode of the modulator. This choice is realized by sending the SD-PHY command with the type '0111'. 2) Now we specify the feed-in rate for the baseband sequence, which is $11MHz$ according to the protocol. We send a command with type '1011' and value $fi(18022, 0, 16, 0)$ according to Eq. 12.

4 HARDWARE PLATFORM

We implement a generic hardware prototype to accommodate the SD-PHY framework as shown in Figure 7. The digital circuit of SD-PHY is written in Verilog and deployed in a low-power flash-based FPGA GW1N-LV4 [26]. The tag can be powered by a standard 3V coin battery widely used in wearable devices. The voltage regulators are connected to the battery and power the remaining circuit. An envelope detector IC along with a comparator works as the analog receiver and provides input for the generic WuRX. An ADG902 RF switch controlled by the FPGA is for backscatter. Two sensors are also available on the tag: a TI HDC2080 temperature and humidity sensor, and a TDK INMP621 low-power microphone.

5 CASE STUDIES

In this section, we show how to use the SD-PHY design framework to realize backscatter systems complying with both standardized and highly-customized wireless protocols.

5.1 Standardized Protocols

5.1.1 Wi-Fi backscatter. We first verify the go-through example mentioned in Section 3.2.4 with the prototype. Figure 8(a) shows the experiments environment and corresponding results. In realizing passive Wi-Fi, we make the excitation signal (single tone) generated in channel 2, and the tag shifts the single tone to channel 6 for backscatter. We use a HUAWEI MATE 30 smartphone and a DELL XPS 13 laptop to demodulate the passive Wi-Fi packets. We also use a cross-platform software NetSpot [27] to report measured RSSI. The results are presented in Fig. 8.

We next show how to make the SD-PHY tag that already works in passive Wi-Fi mode switch to the HitchHike/FreeRider mode. To this end, we need to modify the correlator coefficients because HitchHike/FreeRider needs to synchronize with the excitation signal. For synchronization, we adopt the rising edge '100...111' with mask '001...111' as explained in Section 3.2.1. Alternatively, we can follow the packet length modulation by translating it into an equivalent OOK sequence. Since FreeRider [4] does not elaborate on the sequence, we simply use the sequence '0001110000011111'. To perform codeword translation with two phases, we set $Symbol_1$ as '0' and $Symbol_2$ as '1' with $1/(4\mu s * 4) = 62.5kHz$ symbol rate. We note that although 802.11g has a $250kHz$ symbol rate, the scrambling, encoding and interleaving process is performed over the bits across 4 symbols [4]. To validate our design, we send Wi-Fi beacons periodically over channel 2 and receive the backscatter signal with a Macbook Pro laptop in channel 6. The packets are captured using Wireshark [28]. We present the spectrum of the backscattered Wi-Fi signal and the received packets in Figure 9.

5.1.2 BLE backscatter. We realize the BLE backscatter scheme presented in [8] with the SD-PHY method, which turns out to be non-straightforward due to the unusual BLE modulation scheme. There are two types of modulation techniques in general: memoryless modulation and modulation with memory. The difference lies in whether the modulated symbol is independent of previous symbols. In most wireless protocols like 802.11, memoryless modulation techniques (BPSK and QAM) are used. However, the modulation technique used in BLE is GFSK, a modulation technique with memory. It differs from the classic FSK modulation in that the phase is continuous and the frequency transitions are Gaussian filtered. This problem is also introduced in [10]. Exactly realizing GFSK is impractical due to the Gaussian filtering is outside the scope of our baseband framework. However, we can best approximate the GFSK modulation with continuous-phase FSK (CPFSK) with the Taylor's series computing.

Recall that the $\phi(n)$ and Γ sequence are computed using integral structure as shown in Fig. 5. The second accumulator stores the total phase during the phase calculation. To provide the function of phase continuity, we avoid clearing the phase inside the second accumulator in Fig. 5 when two initial phases (a_0 and b_0) are the same.

Now we introduce how we set SD-PHY parameters to realize BLE backscatter where the tag performs CPFSK. We let the transmitter send a single-tone carrier in 2400 MHz, which is 2MHz

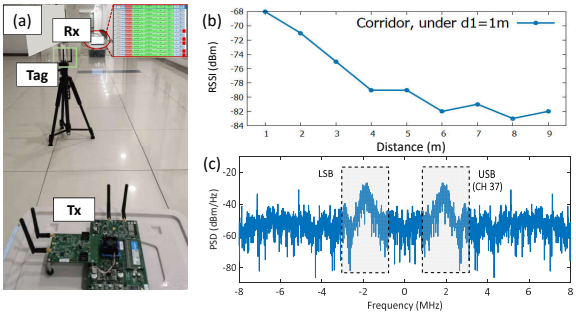

Figure 10: SD-PHY tag realizing BLE backscatter.

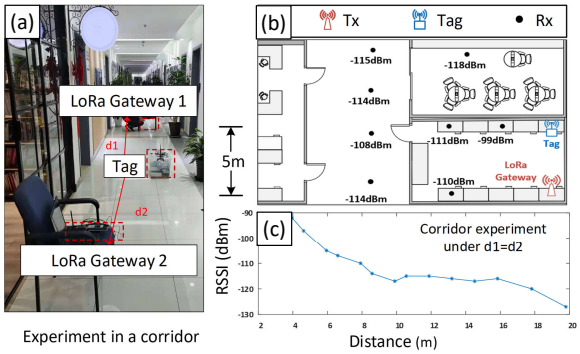

Figure 11: SD-PHY tag realizing XORLoRa backscatter.

aside from the CH 37 (2402 MHz) channel in BLE. We note that we choose CH 37 because it is an advertising channel that can broadcast ADV_NON_CONN_IND packets that can be captured by COTS devices. To send data '1' and '0', the tag should provide $2MHz + 185kHz$ and $2MHz - 185kHz$ square wave, respectively. To realize it, the uplink rate value is calculated as 13107 (8MHz); a_1 and b_1 parameters are set as 14868 and 17900, respectively. The shift frequency and a_2, a_0, b_2, b_0 values are set as zeros. Figure 10(a) shows our experiment setup, the tag is 1m away from Tx, and Rx is a TI CC2540 dongle connected to a laptop. We use SmartRF Sniffer software to capture the BLE packets on CH 37 and filter out backscatter packets based on the advertiser address. The RSSI with distance is plotted in Fig. 10(b). The spectrum of the CPFSK tag signal is presented in Fig. 10(c), where the upper side-band (USB) can be demodulated by the COTS receiver.

5.1.3 LoRa backscatter. We use SD-PHY to realize XORLoRa [14], which is a streamlined version of PLoRa [7]. Though sharing the same design philosophy, XORLoRa improves PLoRa by allowing COTS LoRa receiver to decode the backscattered signal, which facilitates verifying our unified design. To realize XORLoRa design under the SD-PHY framework, we make the following configurations. We let $Shift\_freq_1 = 3MHz$ and $Shift\_freq_2 = 0MHz$. The choice of symbol duration is non-straightforward. We note that due to existence of data interleaving in the COTS LoRa device, we cannot perform OOK modulation with the resolution of a chirp length. If payload bits are embedded to LoRa symbols one by one, then we cannot distinguish modified LoRa data from unchanged LoRa ones from the received bits stream, because the original bits are interleaved on multiple chirps [14]. In fact, the length of a minimum separable data unit lasts for $(4 + CR)$ chirps, meaning that the final symbol duration in a backscatter tag should be: $T_s = (4+CR) \cdot 2^{SF}/BW$ where CR is the "coding rate" parameter [29]. Thus we set the symbol duration as $4ms$ which contains a separable data unit when $CR = 0$. The excitation signal is sent by a LoRa gateway [22] under $10dBm$ transmission power and the receiver is a LoRa gateway with the same model.

In our experiments as shown in Fig. 11, when the Tx-to-tag distance d_1 is set to be $1m$, the tag-to-Rx distance d_2 is over $140m$. Then we set $d_1 = d_2$ in the corridor LOS experiment and $d_1 = 5m$ in the NLOS office environments.

5.1.4 LTE backscatter. LTE backscatter [30] leverages ambient LTE downlink (DL) traffic to realize continuous backscatter communication. Tags synchronize with LTE signals by recognizing the high-envelope primary synchronization signal (PSS) and secondary synchronization signal (SSS). The existing LTE backscatter tag requires a pre-filtering LC circuit to expose SSS and PSS envelope [30], without which it is difficult to detect PSS and SSS. We could generate a standard LTE baseband signal using MATLAB LTE TOOLBOX [31] and up-convert it to 750 MHz center frequency using USRP B210, as shown in Fig. 12. We directly measure the envelope detector output [32], and results are shown in Fig.12(c). We can find that the envelope does change with time yet the PSS signal cannot be well distinguished from other LTE symbols in terms of power. However, performance of the pre-filtering LC circuit is highly dependent on the input signal's symbol duration, thus equipping such circuit will impact versatility of the SD-PHY general hardware.

We present a symbol-level LTE backcatter scheme with SD-PHY, which is free of synchronization on the tag. The basic idea is illustrated in Fig. 13. The top of the figure shows an LTE frame serving as the excitation signal. We could zoom in a fraction of the frame and find corresponding LTE resource elements (REs) including the PSS and SSS. When the ambient LTE excitation signal arrives, the tag modulates the corresponding REs with the local data and then performs frequency shifting. We can see that there are bad data exist in the backscattered LTE signal. Note that the second RE of the backscattered LTE signal covers both "-1" and "1", resulting in a bad data. However, this can be easily resolved with a mean filter at the receiver. In particular, the receiver will compare phases of the resource elements in both the excitation and backscattered LTE signal, where reversed phase means "-1".

In comparison with the sample-level LTE backscatter scheme [30], the symbol-level counterpart is with lower data rate, which is the cost of the SD-PHY's versatility. Our simulation results show that embedding 1-bit tag data into 7 LTE symbols achieves a good trade-off between the tag data rate and the LTE frame detection rate on the receiver side, which makes the SD-PHY tag works at a 1/7 LTE symbol rate ($2kbps$).

To configure the SD-PHY tag, we bypass all correlator coefficients in the wake-up receiver with a mask '111...111'. After setting the wake-up receiver, other parameters are configured as follows. To avoid interference with LTE excitation signals, the shift frequency is set to be 10MHz (There are multiple candidate bands for LTE signals, and we choose $5MHz$ band) with a value $fi(16384, 0, 16, 0)$. The uplink rate is $1/t_{slot} = 2kHz$, with a value $fi(3, 0, 16, 0)$. And then two symbols are π phase apart, i.e. $Symbol_1$ ='0' and $Symbol_2$ ='1'. We send the excitation signal using USRP B210 as shown in Fig. 12(a)

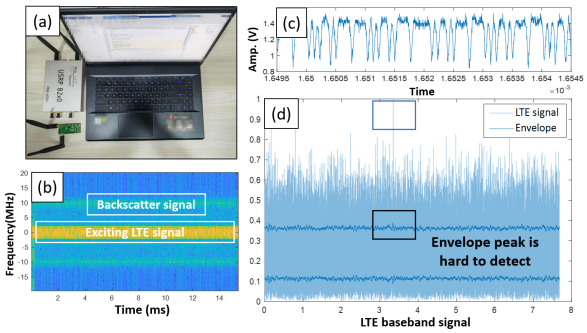

Figure 12: SD-PHY tag realizing LTE backscatter.

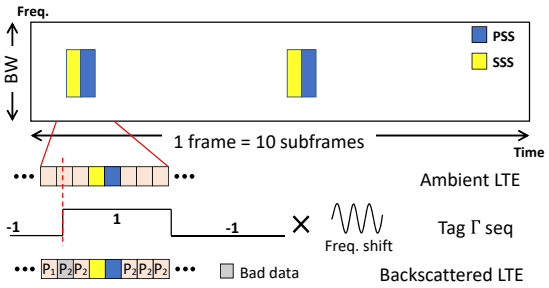

Figure 13: Asynchronous LTE backscatter.

and the received backscatter waveform is illustrated in Fig. 12(b). Since we bypass the wake-up scheme, the backscatter signal is also continuous.

5.2 Customized Protocols

5.2.1 OFDMA backscatter. OFDMA backscatter enables high concurrency in the Wi-Fi backscatter network, which assigns different OFDM subcarriers to different tags [15]. A complete OFDMA backscatter excitation signal consists of a preamble, an OOK sequence, and a continuous wave. The preamble is used to synchronize with the Rx and the OOK sequence is to synchronize with all the tags. On the tag side, we first need to configure the wake-up receiver correlator coefficients according to the OOK sequence,which factually is an 11-bit Barker code '10110111000'. Unfortunately, we find that this sequence, both at 250kHz and 500kHz will experience false triggering in certain environments. This is because ambient signals in the $2.4GHz$ band also contains such a pattern.

In our experiments, we find that the wake-up receiver will be falsely wakened up at a rate of 0.2-121 packets/sec, even if there is no OOK sequence sent by our dedicated excitation signal source. To address the issue, we increase the sequence length by padding five extra bits to the front of the barker code. The correlator coefficients become '1000010110111000' and the mask values are '000...000'. The sequence operating at $500kHz$ can achieve 99.9% wake-up rate on average when the tag is $2m$ away from the RF source.

We now manage to generate appropriate Γ sequence. First, the uplink rate is $20MHz$ ($fi(2^{15}, 0, 16, 0)$) according to the channel bandwidth. An OFDMA tag should work on a subcarrier's frequency. We take subcarrier 16 (5MHz offset) for example, the parameters a_1 and b_1 are configured as $a_1 = b_1 = round(\frac{5MHz}{40MHz} \cdot 2^{16})$. For BPSK modulation, we set $a_0 = round(\frac{\pi}{2\pi} \cdot 2^{16})$ and $b_0 = round(\frac{0}{2\pi} \cdot 2^{16})$

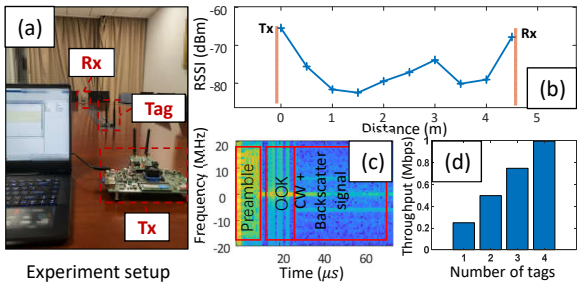

Figure 14: SD-PHY tag realizing OFDMA backscatter.

according to Eq. 9. Finally, a_3 and b_3 are set as $fi(0, 0, 16, 0)$. We send the excitation signal and receive the backscatter signal using WARP v3. We can observe obtained spectrum as shown in Fig. 14(c). To measure performance of the OFDMA backscatter, we fix the positions of both Tx and Rx in an office as illustrated in Fig. 14(a), then move the tag between those two WARPs. We find that it can robustly work at any point with data rate $250kbps$. The measured RSSI results are shown in Fig. 14(b). We use more tags to verify the concurrent transmissions. These tags are assigned with different and adjacent subcarriers (subcarrier index 16-19) by configuring different a_1 and b_1 values. As shown in Fig. 14(d), the aggregated throughput linearly increases when there are 1 to 4 tags.

5.2.2 NetScatter. NetScatter is another high-concurrency backscatter network that uses CSS+OOK modulation in Sub-1GHz band. Each tag is assigned a unique starting frequency regulated in LoRa, which presents the corresponding spectrum pattern after downchirping and FFT at the receiver. Presence and absence of the pattern means that the tag associated with the pattern is transmitting bit 1 or 0, respectively.

To configure SD-PHY parameters for NetScatter, the first step is to consider the wake-up receiver. We still use the configurations in Section 5.2.1, as the detailed synchronization sequence and rate are not explicitly mentioned in the literature. To generate NetScatter Γ sequence, the uplink rate is set to be $500kHz$ with the value $fi(819, 0, 16, 0)$. To avoid self-interference, the shift frequencies are set to be $3MHz$ and $0MHz$ with values $fi(4915, 0, 16, 0)$ and $fi(0, 0, 16, 0)$, respectively. For generating the chirp sequence, we set $a_2 = round(\frac{BW^2}{2^{SF}} \cdot \frac{1}{f_s^2} \cdot 2^{16})$ according to Eq. 11. We let $f_s = BW = 500kHz$, and $SF = 10$, respectively. Then we can obtain $a_2 = fi(2^6, 0, 16, 0)$. We assign the FFT bin 0 to the single tag thus $a_1 = 0$. Parameters b_2, b_1 and b_0 are set to be zeros for the OOK modulation.

In our experiments, the excitation signal transmitter is an analog signal generator [33] with $30dBm$ transmission power. The excitation signal is a single tone at $910MHz$. The receiver is a USRP B210 connected to a laptop for signal processing. To illustrate the backscatter signal spectrum, we let the Rx work at $910MHz$ center frequency with a $10MHz$ sampling rate. The spectrum is shown in Fig. 15(d). We can see that although the backscatter signal is configured to perform upchirping, there is a company downchirp. This is due to the generic hardware of SD-PHY tag cannot perform IQ modulation, and can only generate a real Γ sequence: $cos(2\pi\Delta ft) \cdot cos(2\pi(a_2t^2 + a_1t + a_0))$. We perform decomposition: $(\frac{1}{2}e^{2\pi\Delta ft} + \frac{1}{2}e^{-2\pi\Delta ft}) \cdot (\frac{1}{2}e^{2\pi(a_2t^2+a_1t+a_0)} + \frac{1}{2}e^{-2\pi(a_2t^2+a_1t+a_0)})$.

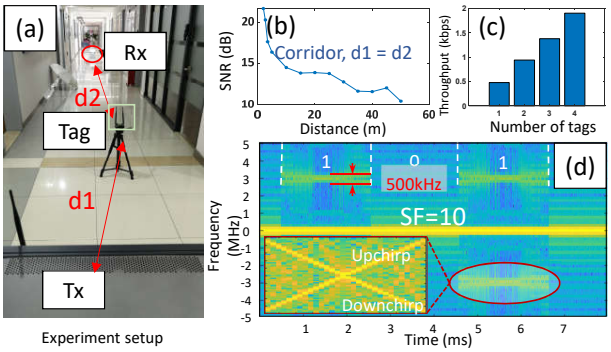

Figure 15: SD-PHY tag realizing NetScatter.

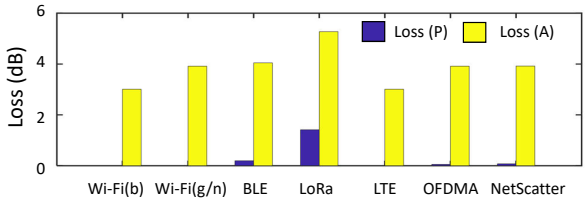

Figure 16: Precision loss of the tag's modulation.

Then two intermediate frequency components show up, with each consisting of a pair of an upchirp and a downchirp. Those company downchirps will not impact the NetScatter performance due to the frequency orthogonality.

In other experiments, the Rx is configured to operate at $913MHz$, which is the center frequency of the backscatter signal with a 4MHz sampling rate to avoid the excitation signal. When the Tx-to-tag distance is $1m$, the backscatter distance is beyond the scale of our test environments, we thus measure the worst case, i.e. when $d_1 = d_2$. The corridor experiment results are shown in Fig. 15(b). The SNR is measured after performing downchirping to the raw received signal. To support concurrent transmissions from multiple tags, we assign different a_1 values to different tags while leaving a_2, a_0 and $\{b_i\}$ unchanged. We verify the concurrent transmission with 4 tags randomly distributed in an $7m \times 8m$ office room. The a_1 values are at the interval of 192, which corresponds to the adjacent tags under $SF = 10, SKIP = 3$ in [16]. The throughput with the number of tags is shown in Fig. 15(c).

6 PERFORMANCE EVALUATION

Precision loss of the tag's modulation. We now evaluate the signal precision loss of the SD-PHY tag's modulation. We note that the precision loss happens in two stages: a) the Taylor series calculation; b) the mapping from inner phase to the available Γ values provided by the generic hardware. To evaluate this, We simulate the tag's inner logic behavior under the Simulink discrete simulation environment and export the fixed-point phase behavior $\phi[n]$. The results are compared with the ideal double-type signal using normalized cross-correlation at the point of zero offset. We define the correlation coefficient value at zero offset as the loss because it reflects how the signal differs from the ideal waveform with a ratio. The results are illustrated in Fig. 16. The dark blue bars (Loss (P)) represent the precision loss caused by the phase calculation in stage a). The yellow bars (Loss (A)) represent all the

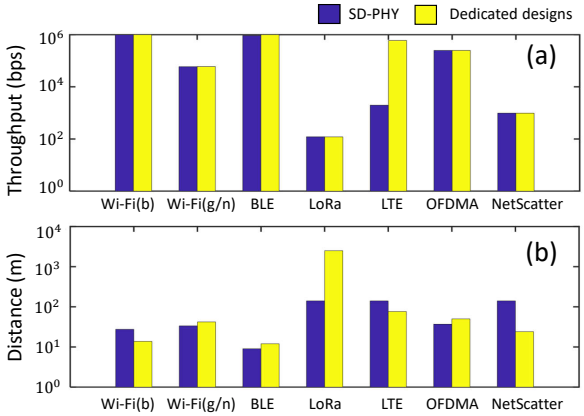

Figure 17: SD-PHY vs dedicated approaches.

precision loss caused in both stages. We can see that the precision loss caused by Taylor series calculation is negligible while the Γ mapping causes the most loss.

Communication throughput/range. We compare performance of the backscatter systems, which are realized by the proposed SD-PHY design and the dedicated design approach as presented in the literature, respectively. The SD-PHY tag is configured as described in the case studies section. We set the Tx-to-tag distance to be $1m$ and measure the maximum achievable PHY data rate at the receiver and the maximum tag-to-Rx distance.

Experimental results are shown in Fig. 17. The tag's modulation order, transmitter's transmission power and bandwidth all influence the overall performance. In Fig. 17(a), when realizing LTE backscatter, systems realized by the SD-PHY approach present lower data rate compared with that by dedicated approach reported in the literature. This is the price of flexibility. In particular, the original LTE backscatter system needs to customize the ED circuit to realize synchronization, which is avoided by the SD-PHY design approach (recall Section 5.1.4). The asynchronized LTE backscatter method sacrifices the capability of detecting PSS and SSS signal for the universal analog front end. Thus the slot-level modulation is adopted for protocol compatibility and causes 10^3 data rate degradation compared to the literature. In Fig. 17(b), the SD-PHY system's communication range is similar to those with dedicated designs except for the lower range in the LoRa backscatter case. This is because the LoRa backscatter adopts a PA to increase the excitation signal power to 30dBm [6].

Some backscatter systems using the original dedicated design can achieve higher data rates than using the SD-PHY design approach (e.g. $11Mbps$ data rate from passive Wi-Fi) for the following reasons: 1) Different choices of PHY parameters. For example, in BLE backscatter, we find it infeasible to realize compatible demodulation based on the COTS receiver as mentioned in Section 5.1.2 and hence we alter the modulation scheme, yielding a different data rate. 2) Limited modulation order due to prototype hardware constraints. Currently, the SD-PHY tag prototype hardware only supports one bit per symbol, thus the tag can only adopt DSSS in realizing 802.11b instead of CCK with a higher data rate. This is mainly due to the limitation of FPGA we adopt in the prototype implementation. The data rate could be improved by applying more

resourceful general hardware design, which physically provides more resources for storing PHY parameters.

Table 2: FPGA resource consumption.

Resource type	Used	Total	Utilization ratio
LUT/ALU	560	4608	12%
Register	560	3549	16%
Block SRAM	0	10	0%

FPGA resource consumption. We evaluate the resource consumption of the SD-PHY baseband. We note that aside from the baseband logic, there is also a data buffer that stores PHY frame bits in our design. We use *GOWIN FPGA Designer* software to synthesize and report the resource consumption of our RTL design in the SD-PHY tag. The total resource consumption is tabulated in Table 2. We can see that the SD-PHY design only consumes a small portion ($< 16\%$) of total resource consumption of the low-power FPGA that we use.

IC power consumption. We now evaluate the IC power consumption of the SD-PHY design. This includes the power consumption of the universal analog hardware and the digital baseband. For the analog hardware, we implement an analog front-end similar to the design in [34]. The envelope detector adopts the pseudo-balun structure and is completely passive. The comparator consumes $0.41\mu W$ under $2MHz$ clock. A ring oscillator based PLL is designed to produce a $40MHz$ system clock, with power consumption $26.2\mu W$. The results mentioned above are based on Cadence Virtuoso [35] with SMIC 40nm process. 2) For the digital baseband, we use Synopsys DC [36] with SMIC 40nm library to synthesize and report power consumption under $1.1V$ voltage supply. To provide a detailed view of all PHY components, we tabulate power consumption results of different blocks in Table 3. Power consumption of dedicated designs are from the literature. The total power consumption considers the sum of three main components (WuRX, baseband, and rate generation) and the RAM that stores tag data. The control plane of the baseband is typically not active and only consumes $0.21\mu W$. When it is active for PHY configurations, the power consumption is $72.70\mu W$ at $1MHz$ after power optimization.

Table 3: IC power consumption of the digital circuit.

Power(μW)	WuRX	BB.	Rate gen.	Total
Passive Wi-Fi	<0.01	36.63	39.88	76.77
FreeRider	2.92	31.27	22.89	57.33
BLE BS.	<0.01	4.21	25.41	29.87
LoRa BS.	0.88	11.88	4.64	17.65
LTE BS.	<0.01	4.21	24.85	29.06
OFDMA BS.	2.92	6.29*	21.56	30.77*
NetScatter	1.46	10.45	9.58	21.48

The results of OFDMA backscatter are marked with asterisks. The value in the table is obtained according to the lowest frequency subcarrier. The actual power consumption is closely related to the frequency of the subcarrier used, which is in the range $6.29\sim62.48\mu W$ (baseband) and $57.58\sim113.78\mu W$ (total). The total power consumption of the SD-PHY tag IC is compared with the dedicated designs in Fig. 18. The OFDMA tag that works with

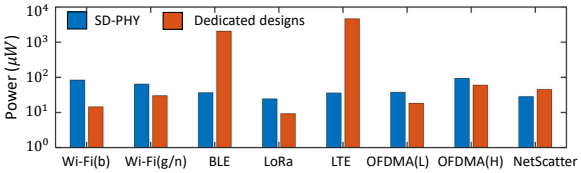

Figure 18: Power consumption.

lowest-frequency subcarrier is termed as "OFDMA(L)" and that with highest-frequency subcarrier is termed as "OFDMA(H)". The y-axis is in *log* scale. We can see that in most cases, the SD-PHY design consumes similar power with dedicated designs (tens of μWs, which can be harvested from ambient energy sources). This means that the unified SD-PHY's flexibility and versatility are at the cost of only tens of microwatts due to the dynamic power caused by the excessive clock frequency and the rate conversion to achieve a universal PHY. Original BLE and LTE backscatter consume significantly more power than the SD-PHY design, because reported power consumption of the two schemes is based on prototype measurement, instead of IC simulation. Recent advances in microelectronics adopt subthreshold designs for low-power mobile applications [37]. It means that the power consumption the SD-PHY tag can be significantly reduced if further voltage optimization is applied in the ASIC design, which is beyond the scope of this paper.

7 DISCUSSIONS

Forward compatibility. For backscatter systems, the excitation signal transmitter such as Wi-Fi AP, BLE nodes and LoRa base station complies with standardized protocols. Such protocols are evolving, which makes forward compatibility a desired design characteristic. In retrospect to the evolution path of those representative standardized protocols, we find that the SD-PHY potentially supports forward compatibility.

In particular, Wi-Fi evolves to have higher modulation order: 11g-64QAM, 11n-256QAM, 11ac-1024QAM, ax-4096QAM, while the fundamental modulation approach adopted is still OFDM with QAM. The modulation order change has no impact on the tag, which just conveys the local information by backscattering the excitation signal with frequency shifting and PSK. However, the symbol duration varies in different Wi-Fi versions: 11g-40μs, 11n-3.6/4.0μs, 11ac-3.6/4.0μs, 11ax-12.8 + 0.8/1.6/3.2μs. This requires the tag to be able to adjust the symbol duration, or the inter-symbol interference will occur. The SD-PHY design supports reconfiguring the symbol duration. The duration value can be updated to the tag with configuring parameters "N_s" and "Uplink rate" as described in Section 3.2.4. Similarly, Bluetooth 4.0 and 5.0 both use GFSK modulation but the bandwidth will increase, LoRa standards have various combinations of *SF* and bandwidth ranges. Such changes in the bandwidth could potentially impact backscatter for self-interference cancellation. This could be easily resolved by SD-PHY by reconfiguring the frequency shifting operation.

Cross frequency band operation. Different PHY protocols are specified under different frequency bands. It requires the SD-PHY tag to support cross frequency operations. Our generic hardware has three components interfacing the RF signal: the ED, the impedance network and the antenna. The existing ED is based on low-barrier Schottky diodes. RF Schottky diodes provided by Infineon and Skyworks naturally support wideband operation up to

24GHz, which covers the bands of most IoT protocols. Impedance network made of capacitors and inductors are frequency-dependent. To make the impedance network independent of the frequency, our prototype hardware adopts the simplest impedance network: 'open circuit' (+∞) and 'closed circuit' (0). We can then calculate the reflection coefficient as $\Gamma(f) = \frac{Z_L(f)-Z^*_{ant}(f)}{Z_L(f)+Z_{ant}(f)}$, where $Z_{ant}(f)$ is the impedance of the antenna and the $Z_L(f)$ is the impedance of the load chosen to be connected to the antenna. If the antenna is perfectly matched at frequency f, then $Z_{ant}(f) = 50\Omega$. We can see that the resulted reflection coefficients are 1 and -1 even when Z_{ant} is not perfectly 50Ω. Antenna is known to be sensitive to the frequency. The frequency band where an antenna works is closely related to the antenna's shape. For example, monopole antennas have a length of around 0.25λ for 50Ω impedance, where λ is the wavelength of the RF signal. Apart from the shape, we note that the impedance matching circuit connected to the antenna can also affect its operating frequency. With dual-band matching process, we can use an impedance matching circuit with passive LC components to realize impedance matching at two different frequency bands like the case in [38]. The SD-PHY tag is mainly designed for the applications in 900MHz and 2.4GHz ISM band, therefore this dual-band matching is sufficient. In our SD-PHY tag prototype, we use a screw mount monopole antenna [39] which is specified at GSM bands. This antenna is also adopted by the PlutoSDR produced by ADI. Experiment results show that the antenna performs well in 900MHz, 2.4GHz and 5.7GHz ISM bands [40].

FPGA vs ASIC realization. Our hardware platform presented in the paper is realized with FPGA; however, it is important to note that the RTL design is fixed in the FPGA. This means that burning a binary file to change the PHY is not necessary. Even if the SD-PHY is realized in ASIC (which is what we do during IC simulation), we are still able to reconfigure the PHY using the scheme as mentioned in Section 3.2.4. For ASIC design, designers usually deploy some registers together with an SPI interface for circuit correction to fight against PVT variations, which is physically equivalent to SD-PHY design. However, these correction registers are for elimating the deviation between actual chip performance and design performance while SD-PHY is to realize a programmable PHY.

Convenience of using the SD-PHY platform. Our SD-PHY hardware platform supports over-the-air (OTA) PHY updating. To realize OTA process, a user needs a laptop connected to an SDR. The SDR should be able to realize 500kHz OOK transmission, which is a simple mission for most SDRs. A configuration frame generator written in Python with GUI can help users calculate PHY parameters and export complex waveforms for SDR transmission. The required fields in the GUI correspond to Table 1. The frame generator software will find valid parameter values that best approximate the user inputs. If the excitation signal is a pure tone signal, SD-PHY tag should comply to certain PHY frame structure to be captured by COTS devices. Therefore, its memory should be written with corresponding PHY bitstream. This bitstream is expected to be provided by the user. Some engineering software can help in this task. We recommend using MATLAB.

8 RELATED WORK

The proposed SD-PHY architecture provides an efficient way to allow the general tag working under various backscatter mechanisms. We note some recent efforts that have been devoted in this vein. Gong *et al.* present an interesting design termed as Multiscatter [41], which makes the same tag able to simultaneously work with various excitation signals including Wi-Fi, Bluetooth and ZigBee. Compared with SD-PHY, the multiscatter tag also demonstrates a way to integrate different ambient backscatter modulation techniques into the single hardware. The core innovation lies in the efficient identification of different ambient signal sources. However, the modulation technique is still fixed and has no reprogrammability, where the versatility is essentially realized in a case-by-case manner. Restricted by the fixed PHY design, a multiscatter tag is unable to function like SD-PHY tags in the following aspects: 1) multiscatter cannot support single-tone transmitters as excitation signal sources. Essentially, it only supports PSK modulation and cannot directly synthesize FSK, CSS and DSSS signals. 2) Multiscatter is not compatible with sub-1GHz protocols like LoRa. It is only designed to work with 2.4GHz ambient RF signal. 3) Multiscatter does not support concurrent transmissions. All multiscatter tags behave identically under a specific excitation signal, which can lead to collisions of backscatter signals.

Backscatter communication itself can be leveraged for cross-technology communication (CTC) [42, 43]. InterScatter shifts frequencies and backscatters the BLE signal so that the signal can be decoded by the Wi-Fi receiver [42]. GateScatter is a backscatter-based gateway bridging ZigBee devices with Wi-Fi hotspot [43]. In comparison, the SD-PHY focuses on agile network reprogrammability, which works under the designated wireless protocol after configuration. SD-PHY provides a higher-level design approach, instead of simply translating among existing backscatter systems.

9 CONCLUSIONS

In this paper, we have presented a novel software-defined PHY (SD-PHY) framework for backscatter networks. This can facilitate innovations in this field by relieving researchers from unnecessary engineering work. With SD-PHY, the tag's PHY-layer behavior can be neatly defined by configuring a set of parameters, which allows the common hardware to generate backscattered signals complying with various wireless protocols. The SD-PHY architecture is based on the key insight that the tag's PHY-layer behavior is essentially determined by reflection coefficient sequence. The SD-PHY is factually to instruct the hardware: How to generate various reflection coefficient sequences that meet different protocols' requirements at the right time; under what clock rate to feed the generated sequence into RF switches on the tag. We have demonstrated that under the unified SD-PHY framework how the same tag can generate different kinds of backscattered signals complying with Wi-Fi, BLE, LoRa, LTE, NetScatter, and OFDMA backscatter. Experimental results show that the system presents the same or close performance compared to dedicated design approach. The SD-PHY architecture could facilitate innovations in the field by relieving researchers from unnecessary engineering work.

10 ACKNOWLEDGEMENTS

The work in this paper is supported by the National Key Research and Development Program of China 2020YFB1708700, and National Natural Science Foundation of China (No. 61922055, 61872233, 61829201, 61532012, 61325012, 61428205). This work is supported by Alibaba Group through Alibaba Innovative Research Program.

REFERENCES

[1] B. Kellogg, A. N. Parks, S. Gollakota, J. R. Smith, and D. Wetherall, "Wi-fi backscatter: internet connectivity for RF-powered devices," in *Proceedings of the 2014 ACM Conference on SIGCOMM (SIGCOMM 14)*, pp. 607–618, 2014.

[2] B. Kellogg, V. Talla, S. Gollakota, and J. R. Smith, "Passive wi-fi: Bringing low power to wi-fi transmissions," in *13th USENIX Symposium on Networked Systems Design and Implementation (NSDI 16)*, pp. 151–164, 2016.

[3] P. Zhang, D. Bharadia, K. Joshi, and S. Katti, "Hitchhike: Practical backscatter using commodity wifi," in *Proceedings of the 14th ACM Conference on Embedded Network Sensor Systems (SenSys 16)*, pp. 259–271, 2016.

[4] P. Zhang, C. Josephson, D. Bharadia, and S. Katti, "Freerider: Backscatter communication using commodity radios," in *Proceedings of the 13th International Conference on emerging Networking EXperiments and Technologies (CoNEXT 17)*, pp. 389–401, 2017.

[5] J. Zhao, W. Gong, and J. Liu, "Spatial stream backscatter using commodity wifi," in *Proceedings of the 16th Annual International Conference on Mobile Systems, Applications, and Services (MobiSys 18)*, pp. 191–203, 2018.

[6] V. Talla, M. Hessar, B. Kellogg, A. Najafi, J. R. Smith, and S. Gollakota, "Lora backscatter: Enabling the vision of ubiquitous connectivity," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 1, no. 3, pp. 1–24, 2017.

[7] Y. Peng, L. Shangguan, Y. Hu, Y. Qian, X. Lin, X. Chen, D. Fang, and K. Jamieson, "Plora: A passive long-range data network from ambient lora transmissions," in *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication (SIGCOMM 18)*, pp. 147–160, 2018.

[8] J. F. Ensworth and M. S. Reynolds, "Every smart phone is a backscatter reader: Modulated backscatter compatibility with bluetooth 4.0 low energy (ble) devices," in *2015 IEEE international conference on RFID (RFID)*, pp. 78–85, IEEE, 2015.

[9] A. Abedi, F. Dehbashi, M. H. Mazaheri, O. Abari, and T. Brecht, "Witag: Seamless wifi backscatter communication," in *Proceedings of the Annual Conference of the ACM Special Interest Group on Data Communication on the Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM 20)*, p. 240–252, 2020.

[10] M. Zhang, S. Chen, J. Zhao, and W. Gong, "Commodity-level BLE backscatter," in *Proceedings of the 19th Annual International Conference on Mobile Systems, Applications, and Services (MobiSys 21)*, pp. 402–414, 2021.

[11] X. Liu, Z. Chi, W. Wang, Y. Yao, P. Hao, and T. Zhu, "Verification and redesign of OFDM backscatter," in *18th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2021, April 12-14, 2021* (J. Mickens and R. Teixeira, eds.), pp. 939–953, USENIX Association, 2021.

[12] X. Guo, L. Shangguan, Y. He, J. Zhang, H. Jiang, A. A. Siddiqi, and Y. Liu, "Aloba: Rethinking on-off keying modulation for ambient lora backscatter," in *Proceedings of the 18th Conference on Embedded Networked Sensor Systems (SenSys 20)*, pp. 192–204, 2020.

[13] J. Jiang, Z. Xu, F. Dang, and J. Wang, "Long-range ambient lora backscatter with parallel decoding," in *Proceedings of the 27th Annual International Conference on Mobile Computing and Networking*, MobiCom '21, (New York, NY, USA), p. 684–696, Association for Computing Machinery, 2021.

[14] H. Li, X. Tong, Q. Li, and X. Tian, "XORLoRa: Lora backscatter communication with commodity devices," in *2020 IEEE 6th International Conference on Computer and Communications (ICCC)*, pp. 706–711, 2020.

[15] R. Zhao, F. Zhu, Y. Feng, S. Peng, X. Tian, H. Yu, and X. Wang, "OFDMA-enabled wi-fi backscatter," in *The 25th Annual International Conference on Mobile Computing and Networking (MobiCom 19)*, pp. 1–15, 2019.

[16] M. Hessar, A. Najafi, and S. Gollakota, "Netscatter: Enabling large-scale backscatter networks," in *Proceedings of the 16th USENIX Conference on Networked Systems Design and Implementation (NSDI 19)*, pp. 271—-283, 2019.

[17] SD-PHY hardware platform, 2022. https://github.com/Swattzz/SD-PHY-Backscatter.

[18] V. Liu, A. N. Parks, V. Talla, S. Gollakota, D. Wetherall, and J. R. Smith, "Ambient backscatter: wireless communication out of thin air," in *Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM (SIGCOMM 13)*, pp. 39–50, 2013.

[19] D. Bharadia, K. R. Joshi, M. Kotaru, and S. Katti, "Backfi: High throughput wifi backscatter," in *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication (SIGCOMM 15)*, pp. 283–296, 2015.

[20] F. Zhu, Y. Feng, Q. Li, X. Tian, and X. Wang, "Digiscatter: Efficiently prototyping large-scale ofdma backscatter networks," in *Proceedings of the 18th International Conference on Mobile Systems, Applications, and Services (MobiSys 20)*, pp. 42––53, 2020.

[21] Z. An, Q. Lin, P. Li, and L. Yang, "General-purpose deep tracking platform across protocols for the internet of things," in *Proceedings of the 18th International Conference on Mobile Systems, Applications, and Services (MobiSys 20)*, pp. 94–106, 2020.

[22] SX1276RF1KAS evaluation module. https://www.semtech.com/products/wireless-rf/lora-transceivers/sx1276rf1kas.

[23] WEBENCH Clock Architect Tool. https://www.ti.com/design-resources/design-tools-simulation.html.

[24] Clocking Wizard. https://www.xilinx.com/products/intellectual-property/clocking_wizard.html.

[25] Fixed-point numeric object. https://www.mathworks.com/help/fixedpoint/ref/embedded.fi.html.

[26] LittleBEE FPGAs. https://www.gowinsemi.com/en/product/detail/2/.

[27] Wi-Fi Site Surveys, Analysis, Troubleshooting. https://www.netspotapp.com/.

[28] Network protocol analyzer. https://www.wireshark.org/.

[29] A. Marquet, N. Montavont, and G. Z. Papadopoulos, "Towards an sdr implementation of lora: Reverse-engineering, demodulation strategies and assessment over rayleigh channel," *Computer Communications*, vol. 153, pp. 595–605, 2020.

[30] Z. Chi, X. Liu, W. Wang, Y. Yao, and T. Zhu, "Leveraging ambient LTE traffic for ubiquitous passive communication," in *Proceedings of the Annual Conference of the ACM Special Interest Group on Data Communication on the Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM 20)*, pp. 172–185, 2020.

[31] MATLAB, "Lte waveform generation and transmission using quick control rf signal generator." https://www.mathworks.com/products/lte.html.

[32] A. Devices, "Lt5534, 50mhz to 3ghz rf power detector." https://www.analog.com/media/en/technical-documentation/data-sheets/5534fc.pdf.

[33] EXG X-Series Signal Generators. https://www.keysight.com/us/en/assets/7018-03381/data-sheets/5991-0039.pdf.

[34] P.-H. P. Wang, C. Zhang, H. Yang, D. Bharadia, and P. P. Mercier, "20.1 a 28μw iot tag that can communicate with commodity wifi transceivers via a single-side-band qpsk backscatter communication technique," in *2020 IEEE International Solid-State Circuits Conference (ISSCC 20)*, pp. 312–314, 2020.

[35] Virtuoso Layout Suite. https://www.cadence.com/ko_KR/home/tools/custom-ic-analog-rf-design/layout-design/virtuoso-layout-suite.html.

[36] DC Ultra. https://www.synopsys.com/implementation-and-signoff/rtl-synthesis-test/dc-ultra.html.

[37] A. Wang, B. H. Calhoun, and A. P. Chandrakasan, *Sub-threshold design for ultra low-power systems*, vol. 95. Springer, 2006.

[38] TI. Monopole PCB Antenna with Single or Dual Band Option. https://www.ti.com/lit/an/swra227e/swra227e.pdf?ts=1650272717597.

[39] Jinchang Electron. JCG 401 GSM Antenna. https://jqrorwxhqioplk5p.ldycdn.com/JCG401-aidqqBpmKjiRliSkjorlklli.pdf.

[40] Analog Devices. ADALM-PLUTO Antennas. https://wiki.analog.com/university/tools/pluto/users/antennas.

[41] W. Gong, L. Yuan, Q. Wang, and J. Zhao, "Multiprotocol backscatter for personal iot sensors," in *Proceedings of the 16th International Conference on Emerging Networking EXperiments and Technologies (CoNEXT 20)*, pp. 261—-273, 2020.

[42] V. Iyer, V. Talla, B. Kellogg, S. Gollakota, and J. R. Smith, "Inter-technology backscatter: Towards internet connectivity for implanted devices," in *Proceedings of the 2016 ACM SIGCOMM Conference (SIGCOMM 16)*, pp. 356–369, 2016.

[43] J. Jung, J. Ryoo, Y. Yi, and S. M. Kim, "Gateway over the air: Towards pervasive internet connectivity for commodity iot," in *Proceedings of the 18th International Conference on Mobile Systems, Applications, and Services (MobiSys 20)*, p. 54–66, 2020.