OpenMili: A 60 GHz Software Radio Platform With a Reconfigurable Phased-Array Antenna

Jialiang Zhang, Xinyu Zhang, Pushkar Kulkarni and Parameswaran Ramanathan
Department of Electrical and Computer Engineering
University of Wisconsin-Madison
{jialiang.zhang, xyzhang}@ece.wisc.edu, {pkulkarni, parmesh.ramanathan}@wisc.edu

ABSTRACT

The 60 GHz wireless technology holds great potential for multi-Gbps communications and high-precision radio sensing. But the lack of an accessible experimental platform has been impeding its progress. In this paper, we overcome the barrier with OpenMili, a reconfigurable 60 GHz radio architecture. OpenMili builds from off-the-shelf FPGA processor, data converters and 60 GHz RF front-end. It employs customized clocking, channelization and interfacing modules, to achieve Gsps sampling bandwidth, Gbps wireless bit-rate, and Gsps sample streaming from/to a PC host. It also incorporates the first programmable, electronically steerable 60 GHz phased-array antenna. OpenMili adopts programming models that ease development, through automatic parallelization inside signal processing blocks, and modular, rate-insensitive interfaces across blocks. It provides common reference designs to bootstrap the development of new network protocols and sensing applications. We verify the effectiveness of OpenMili through benchmark communication/sensing experiments, and showcase its usage by prototyping a pairwise phased-array localization scheme, and a learning-assisted real-time beam adaptation protocol.

CCS Concepts

•**Networks** → *Network experimentation;* Programming interfaces; •**Hardware** → *Digital signal processing;* •**Computer systems organization** → Reconfigurable computing;

Keywords

60 GHz, Millimeter-wave, Software radio, Testbed, Experimental platform

1. INTRODUCTION

The unlicensed millimeter-wave (mmWave) spectrum around the 60 GHz frequency promises a blueprint of wireless networking at wire-speed. The vast amount of spectrum resource, spanning 57 GHz to 64 GHz in many countries, enables multi-Gpbs data rate. The small wavelength ($\sim 5mm$) enables miniature antenna-arrays that can form highly directional "pencil beams" to boost link quality and spatial reuse. mmWave is thus considered as an enabling technology for 5G wireless broadband [1]. Recent commercialization of 60 GHz devices (*e.g.*, by Qualcomm [2] and Intel [3]) also triggers low-cost *mmWave sensing* applications, which used to be available only in dedicated environment for medical/security inspection. Together, short wavelength and high directionality translates into high sensitivity, enabling subtle object localization/tracking [4], vital-sign detection and mobile mmWave imaging [5].

Realizing the vision of mmWave networking and sensing necessitates a reconfigurable experimental platform that allows prototyping before the protocols/applications are deployed. Ideally, one would need a mmWave software-radio platform that allows reconfiguration from the PHY layer up to applications, can transmit customized waveforms and acquire RSS/phase information for sensing applications [4, 6–8]. On the 2.4 GHz and 5 GHz microwave spectrum, counterpart devices such as USRP [9], WARP [10] and Sora [11] have reshaped the landscape of wireless experimentation in the past decade, speeding up the ratification of next-generation wireless standards (*e.g.*, 802.11ax [12]) and sensing appliances [13]. However, to our knowledge, there exists no reconfigurable platform that can capture the unique features of 60 GHz wireless systems, particularly the Gsps sampling bandwidth and electronically steerable phased-array antennas.

In this paper, we propose *OpenMili*, an open-access 60 GHz software-radio, which fills the gap and opens up new directions for mmWave sensing and protocol development. OpenMili has a software-defined mmWave network stack spanning PHY layer signal processing to applications. It can also act as a programmable 60 GHz radio sensor with a custom-built phased-array. From the hardware architecture perspective, OpenMili integrates off-the-shelf baseband processing unit (BPU), ADC/DAC and 60 GHz RF front-end (frequency up/down-converters), and develops a signal chain to enable Gsps of sampling bandwidth. It allows flexible channelization and reclocks the 60 GHz front-end to overcome its inherent phase-noise problem. OpenMili's baseband processing unit (BPU) centers around a Kintex UltraScale FPGA, and uses a customized PCIe module to realize 1 Gsps real-time sample streaming from/to a PC host. The PC host can reconfigure/monitor the RF front-end and signal processing modules in real-time. OpenMili's most out-

MobiCom'16, October 03-07, 2016, New York City, NY, USA
© 2016 ACM. ISBN 978-1-4503-4226-1/16/10. . . $15.00
DOI: http://dx.doi.org/10.1145/2973750.2973760

standing feature is a reconfigurable phased-array antenna that can switch between 16 beam patterns at μs granularity, under the control of the BPU. We design the phased-array specifically to fit the WR-15 waveguide (a standard antenna interface on 60 GHz radios), ensuring it can retrofit both OpenMili's RF front-end and other commercial mmWave radios that are typically equipped with WR-15 horn antennas [14].

From the software architecture perspective, OpenMili takes advantage of the massive parallelization on the FPGA to enable Gsps sample processing. It eases prototyping by using C++ to define signal processing blocks, and using AXI [15] as inter-block gluing mechanism, which facilitates modularity and eliminates the need for inter-block rate matching—a well-known headache in FPGA programming. We choose this programming model also because the complexity of intra-block parallelization can be hidden from application developers, although coarse-grained inter-block parallelization still need to be explicitly expressed.

OpenMili provides three reference designs which we believe to capture the unique aspects of mmWave and can be instrumental for a wide range of 60 GHz network protocols and wireless sensing applications [4, 6–8]. *(i)* Gbps baseband communication module: allowing a wide range of network protocol development on the FPGA or PC host; *(ii)* Real-time RSS/phase sensing: using an 802.11ad-like preamble to sense the channel state information (CSI), with around 300K CSI readings per second across 1 GHz bandwidth, enabling many real-time sensing applications [8]. *(iii)* Real-time phased-array controller: allowing 16 beamforming patterns based on a discrete codebook. Codebook entry selection is made in real-time on a Microblaze processor, programmable in C. This module can also access the real-time CSI statistics, allowing beam adaptation based on channel conditions.

To showcase the use of OpenMili in prototyping 60 GHz systems, we propose and implement two new 60 GHz location sensing and beamforming adaptation mechanisms, which are of independent interest. *(i)* Pairwise relative localization of phased-arrays. Many recent systems have used 2.4 GHz phased-arrays [16,17] for angle-of-arrival (AoA) estimation, but they need multiple phased-arrays to triangulate a target radio, and they assume the phase-shift values are continuously adjustable, which is not applicable to practical 60 GHz phased-arrays that use hard-wired phase-shifters. We propose a simple algorithm that leverages the discrete phase-shifting to estimate the AoA as well as distance between a pair of phased-arrays, enabling pairwise localization instead of triangulation. *(ii)* Learning-assisted real-time beam adaptation. The short-wavelength at 60 GHz enables compact phased-array design, with many antenna elements but correspondingly a large number of beamforming codebook entries to choose from, which entails huge adaptation overhead at run time. We propose a new principle to make this tradeoff: we allow a pair of phased-array nodes to learn the correlation between beam patterns offline, and then prune the adaptation space at run-time, which substantially saves the beam searching overhead.

Contributions. To prepare for an open-source release, we have intentionally used off-the-shelf hardware modules to build OpenMili (except for the programmable phased-array). Our main contribution lies in tasking these modules into a reconfigurable architecture, designing the software framework and common interfaces to enable the development of new mmWave communication and sensing applications. This contribution breaks down into the following aspects:

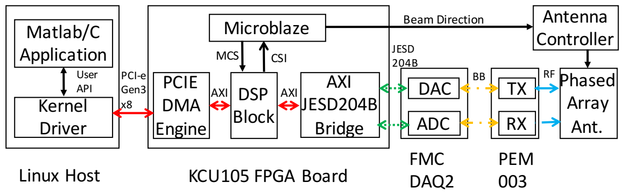


Figure 1: OpenMili's hardware architecture.

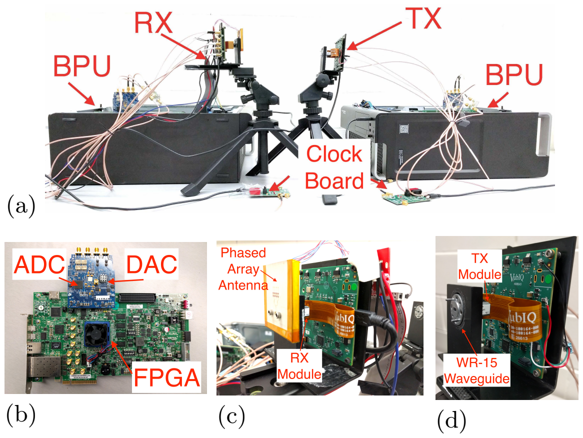


Figure 2: Portrait of OpenMili: (a) Overall architecture; (b) Baseband processing uint (BPU); (c) Receiver with phased array antenna; (d) Transmitter without mounting antenna.

(i) Hardware architecture (Sec. 2 and 4): We develop the first 60 GHz reconfigurable radio architecture that achieves Gsps sampling bandwidth, Gbps wireless bit-rate, and Gsps real-time sample streaming from/to a PC host. The hardware architecture builds on customized clocking, channelization and interfacing modules. Most remarkably, it incorporates the first programmable 60 GHz phased-array design.

(ii) Software framework (Sec. 3): We explore FPGA programming models that ease development, through automatic parallelization inside signal processing blocks, and modular, rate-insensitive interfaces across blocks. We also provide reference designs to bootstrap the development of protocols and sensing applications, featuring the Gbps communications and beam steering capabilities of 60 GHz radios.

(iii) New network protocol and sensing modality (Sec. 6): We design a new 60 GHz beam adaptation protocol and a simple pairwise location application. We further showcase the usefulness of the reconfigurable platform in prototyping and experimenting with such mechanisms.

The software code and hardware schematics of OpenMili will be available on our project website [18].

2. HARDWARE ARCHITECTURE

The diagram in Figure 1 and portrait in Figure 2 illustrate OpenMili's hardware architecture and Tx/Rx signal chains. OpenMili comprises 5 major modules: PC host, baseband processing unit (BPU), data converters (ADC/DAC), RF front-end, and phased-array antenna. The PC host configures the high-level parameters of other modules, and can also generate digital waveforms or log the received digital signal samples for offline processing. It interfaces with the BPU via PCIe gen3. The BPU is an FPGA module that executes real-time physical layer signal processing and MAC layer protocols. Along the Tx signal chain, basedband signals generated by the BPU or PC host are eventually converted to analog waveforms by the DAC, upconverted to 60 GHz by the RF front-end, and emitted through a custom-built phased-array antenna; and vice versa for the Rx signal chain.

Besides reconfigurability, the hardware architecture aims for a few ambitious performance goals: Gsps sampling rate, real-time Gsps signal processing at BPU and sample transportation to PC host, and low phase noise. Below we describe the hardware modules, along with the interface design and optimization to meet the goals.

2.1 Hardware Modules

RF front-end. OpenMili uses the PEM-003 60 GHz transceivers [19] as RF front-end. PEM-003 is essentially an 802.11ad compatible frequency up-converter/down-converter. Its carrier frequency can switch between 4 channels from 57.24 GHz to 64.80 GHz, each channel spanning 1.8 GHz analog bandwidth. Alternatively, it can degrade to a custom-mode, and switch between 15 channels, each spanning 540 MHz. It also has a programmable RF gain controller, allowing continuous adjustment of output power up to 12 dBm. PEM-003 can be connected to any 60 GHz antennas with a WR-15 waveguide interface.

Baseband processing unit (BPU). OpenMili's BPU employs Xilinx's KCU105 development board, centered around a Kintex UltraScale FPGA *XCKU040.* We choose XCKU040 FPGA as it supports the PCIe gen3 interface which can provide enough bandwidth to support multi-Gsps data streaming to a PC host. Also, it has 1920 on-chip DSP slices that serve as arithmetic logic units (ALU). Each DSP slice has two 32-bit adder, one 24-bit multiplier and other logics (*e.g.*, multiplexer, shifter). The DSP slices together can provide a computation throughput of up to 960G multiply and add operations per second, making Gsps signal processing possible.

The BPU acts as the central processing module within the OpenMili architecture. It not only executes real-time signal processing, but also houses the FMC (FPGA magazine board) sampling board that interfaces with the data converters, and the PCIe interface to the PC host, as well as 1.8V GPIO (AXI-lite to control, 40 MHz max I/O speed) to control the phased array antenna.

ADC and DAC. OpenMili's data converter module employs the FMCDAQ2 development board [20] from Analog Devices, which integrates a dual-channel AD9680 ADC [21], AD9144 DAC [22], and other peripheral components on an FMC daughter board attached to the BPU.

Each ADC channel (I or Q) samples at 1 Gsps with a 12-bit resolution, supporting up to 1 GHz bandwidth within the baseband. It has an internal data path with $2\times, 4\times$ and $8\times$ decimation filters, and a numerically controlled oscillator. This makes it possible to instantaneously switch between different bandwidth configurations under the control of the BPU.

The DAC has 14-bit high resolution, and default to 1 Gsps sampling rate to match the ADC rate. Its outputs waveforms are staircase-like, whose frequency response is a *sinc* function and causes a 6 dB loss at the maximum sampling frequency [23]. This will degrade the magnitude of high-frequency baseband signals. To reduce such frequency selective artifacts, we enable the $2\times$ upsampler inside the DAC, which increases the output sampling rate from 1 Gsps to 2 Gsps. Such oversampling effectively limits the bandwidth of output baseband signals to 1 GHz, within which the DAC's gain remains relatively flat.

2.2 Designing Interfaces Between Hardware Modules

Interfacing data converters with RF front-end. The PEM-003 RF front-end can only output or receive I/Q samples via a differential interface, *i.e.*, it has no reference to ground and voltage samples are taken as the difference between two wires. Yet, the data converters use a single-ended interface, with voltage measured between a single wire and ground. To bridge the PEM-003 and data converters, we insert a BALUN circuit [24] which performs the single-ended to differential conversion for each I and Q channel. Each BALUN has 500 MHz bandwidth, so it also serves as an analog low-pass filter for the I/Q channel, preventing the notorious aliasing effect caused by digital sampling [23].

An additional interfacing issue lies in voltage mismatch: the DAC has an output power level of 0 dBm [25], whereas the maximum input level of the PEM-003 is -20 dBm. To overcome this issue, we insert a 30 dB attenuator in between, which ensures a -30 dBm input power into PEM-003, well below the limit. Albeit conservative, this does not sacrifice performance, because PEM-003 has an RF gain up to 38 dBm, which can rescale the input signals to $38-30 = 8$ dBm and still ensure the transmitter is not saturated (PEM-003's transmitter front-end saturates at around 12 dBm [26]). On the other hand, PEM-003's receiver front-end has a variable gain amplifier, which can be adjusted to match the ADC's input power level without any external attenuator.

Interfacing BPU and data converters using JESD-204B. The ADC/DAC use JESD204B, a high-speed serial interfacing protocol, to transport data samples from/to the BPU's FPGA. JESD204B transports data in the units of 16-bit (to cover the maximum resolution of data converters), and the sampling rate of both I and Q channels are 1 Gsps. So the total data rates on the JESD204B interface is $16_{bps/converter} \cdot 2_{converters} \cdot 1Gsps = 32Gbps.$ Together with 25% overhead induced by JESD204B's error correction code, the total data rate is 40 Gbps. Both the ADC and DAC use up to 4 serial channels, each being a copper line with 10.76 Gbps maximum rate. So we reconfigure the JESD204B and limit each channel's rate to 10 Gsps to meet the 40 Gbps requirement.

Interfaces to the PC host. To interface the BPU with the PC host, we custom-built a PCIe gen3 driver, which allows the BPU to route all digital samples in real time between the PC and the data converters (detailed in Sec. 3.2). Between the PEM-003 RF front-end and the PC host, there exists a built-in USB interface, with a proprietary graphical interface for configuring the RF parameters. But this interface precludes integration with other components in OpenMili. We thus reverse-engineer the USB interface using USBPcap [27], and decypher the configuration commands and USB protocols between the host and PEM-003. Then, we develop our own USB driver in C, which replays the protocol to reconfigure the PEM-003 via real-time command lines.

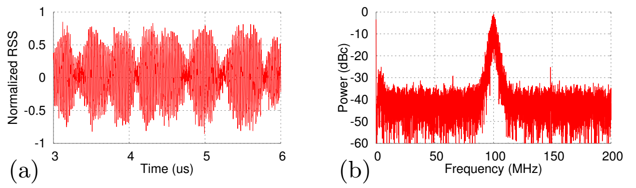


Figure 3: Waveform/spectrum of a 100 MHz sine tone with the original carrier clock in PEM-003.

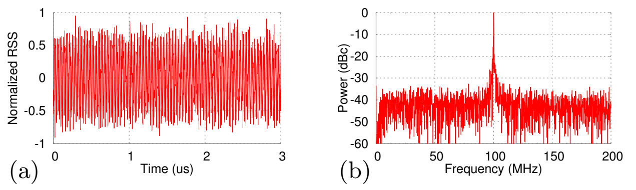


Figure 4: Waveform/spectrum after reclocking.

2.3 Reclocking and Noise Reduction

Reducing carrier clock noise. Phase noise, *i.e.*, short-term fluctuation in the phase of carrier waveform, represents a key metric in rating an RF front-end. For a millimeter-wave radio, phase-noise is especially critical, because the carrier clock (tens of GHz) is generated by multiplying a low-frequency (MHz) reference clock, and even a low phase-noise at the reference will be magnified by orders of magnitude [28]. Specific to the PEM-003 front-end, its reference clock works at 308.571 MHz, and the phase-noise will be amplified by $60GHz/308.571MHz = 194 = 23dB$.

For an empirical understanding the phase noise of PEM-003, we send a 100 MHz baseband sine tone with a constant magnitude, using a highly-directional 3.4° horn antenna to preclude multipath channel distortion. Figure 3(a) shows the received waveform, whose magnitude manifests many notches over time. This is mainly attributed to sudden phase fluctuation, which prevents the sine wave from reaching its magnitude. Figure 3(b) plots the frequency-domain spectrum of the received signal, which is supposed to be a single peak, but actually smeared significantly, indicating the noise is multiplicative rather than additive. This also hints to the characteristics of phase noise [28], which cannot be eliminated using conventional linear filters.

Since we cannot modify the on-chip clock of PEM-003, we use an external clock [29], driven by a high performance PLL, to substitute the original reference clock. The new clock is connected to PEM-003 via its MCX reference clock port. Figure 4 shows the 100 MHz sine tone after the reclocking. We observe a stable envelop, corresponding to a low-noise single-carrier frequency spectrum, which verifies the effectiveness of reclocking.

Reducing DC leakage noise. DC noise comes from the leakage of RF circuit's carrier signals, which translates into the zero-frequency in baseband, and may compromise the SNR of low-frequency baseband signals. In OpenMili, DC leakage is attributed to the RF hardware PEM-003, but can be mitigated via software-based signal processing in two ways.

(i) Band shifting, *i.e.*, migrating the entire baseband signals away from zero-frequency, which we realize by modulating the signals with a numerical controlled oscillator (NCO) built in the data converters. Figure 5 provides experimental evidence for this mechanism. In this experiment, we send a narrowband QPSK signal with 500 kHz bandwidth and sweep the center frequency from 1MHz to 500 MHz. We measure the error vector magnitude (EVM) of received signals as the normalized Euclidean distance between the transmitted and received signal vectors in the complex plane. The results show that the EVM near DC can be as high as 22%, due to strong DC leakage. But the EVM rolls off over frequency and becomes negligible above 100 MHz. Thus, in case when bandwidth can be sacrificed, we always migrate the baseband signals beyond 100 MHz to ensure high SNR.

(ii) Oversampling, equivalent to maintaining the Gsps sampling rate in OpenMili but reducing baseband signal's bandwidth. We realize this by implementing a downsampler (inside the data converter board), combined with a low-pass filter, both reconfigurable through OpenMili's BPU. To understand the benefits empirically, we send out a 200 MHz single-tone signal at 200MHz, and then measure the received SNR. The result in Figure 6 shows that every 2× of oversampling provides around 3 dB SNR gain, albeit cutting the signal bandwidth by half. It is up to the application developer to decide on the proper tradeoff.

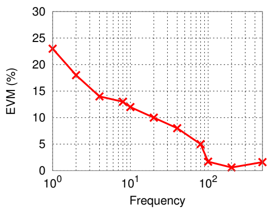


Figure 5: Impact of DC leakage noise.

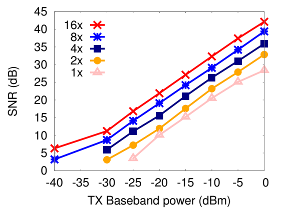


Figure 6: Oversampling improves SNR.

	Real-time	Non-real-time
Control	Microblaze	PC
SP (heavy)	FPGA	PC
SP (light)	Microblaze	PC

Table 1: Choosing the processing unit for different tasks and requirements. SP: signal processing.

3. SOFTWARE FRAMEWORK AND PROGRAMMING MODELS

In OpenMili, the general purpose processor (GPP) on the PC host, the FPGA-based BPU, and the microblaze processor together form a heterogeneous computing architecture. Depending on application needs, one or multiple processing units need to be chosen appropriately to balance the tradeoffs between computational throughput and flexibility, as summarized in Table 1. OpenMili's software framework and programming models are designed accordingly to ease such choices. We build high-rate real-time signal processing blocks in FPGA using C++, and leverage high-level synthesis (HLS) to automate the code-level parallelization and achieve Gsps processing speed. We have also developed firmware to enable Gsps sample transportation between the BPU and the PC host, and allow fast prototyping of signal processing algorithms running on the GPP. Besides, we employ the BPU's microblaze processor when it becomes necessary to execute real-time control and signaling operations (*e.g.*, medium access protocols) between signal processing blocks. Table 2 summarizes the percentage of FPGA resource utilization in OpenMili's software framework.

	Base	w/ microblaze	Reference design
LUT	5.09	9	23
Reg.	6.07	14	32
BRAM	20	43	54
DSP	0	3.32	34.14

Table 2: FPGA utilization percentage with OpenMili's heterogeneous computing architecture and software framework.

3.1 Programming Models for FPGA Based Real-Time Processing

In general, a software-radio should support three categories of operations: TWEAK (defining and modifying a signal processing block); TAP (monitoring the output/status of a block); and INSERT/DELETE (*i.e.*, adding/removing blocks) [30]. OpenMili's programming model supports these operations with two design goals: *(i)* tweaking a digital signal processing (DSP) block to achieve Gsps processing speed with deterministic timing, leveraging automated massive parallelization instead of low-level hardware language. *(ii)* enabling flexible and *modular* TAP and INSERT, such that developers need not worry about rate compatibility between consecutive DSP blocks, which is a common headache for low-level FPGA development.

3.1.1 Automating Parallelization and Pipelining for Gsps Processing

Why using HLS in SDR? HLS is an automated design process that interprets a high-level language (*e.g.*, C/C++) into hardware description language (HDL) that can be executed by the FPGA. HLS has gained popularity and maturity in the past a few years, and major FPGA providers (*e.g.*, Xilinx and Altera) have all released their own HLS toolchain [31, 32]. Building SDR application in HLS has several advantage over the traditional HLS.

The HLS programming model affords similar flexibility as the GPP-based GNURadio [33], which uses high level language to define and glue the DSP blocks. A plethora of common DSP blocks are already available from FPGA providers and third-party developers. HLS hides majority of the hardware details from DSP algorithm developers. More importantly, HLS allows us to flexibly adjust the level of algorithm parallelization, which is critical for real-time Gsps signal processing.

Pipelining and parallelization in HLS. The clock speed of modern FPGAs is limited to 200∼300 MHz. To enable Gsps signal processing, we must take advantage of the massive parallelization. In traditional HDL, transforming the signal processing algorithms to fit FPGA's parallel architecture entails significant programming efforts. A minor adjustment to parallelization level may result in a chain effect, requiring an overhaul of the entire HDL code base. In contrast, given a target parallelization level, an HLS toolchain can tell how many cycles are needed for a DSP block to generate all the outputs, with how many resource elements. This helps developers to make the best tradeoff between resource consumption and performance.

We showcase the automated parallelization using an FIR filter block, which comprises a sequence of multiply-and-add operations, most commonly seen in DSP blocks. Figure 7 shows the C++ code for an N-tap FIR filter. The *for* loop executes one multiply-and-add in each iteration, and it needs N cycles to generate the outputs for each new signal sample. In contrast, Figure 8 shows a manually "unrolled" version of the code, with unrolling factor of 2, *i.e.*, it can calculate 2 FIR stages within one cycle. A higher unrolling factor achieves higher parallelization, but at the cost of higher hardware resource consumption. HLS enables us to automate the unrolling procedure – developer only needs to set the unrolling factor in a configuration file, and then HLS can automatically accomplish the loop unrolling based on the single version of C++ code, while providing a latency/throughput report. Developer only needs to iterate over the configuration instead of rewriting the code.

```
for (i=N-1;i>=0;i--) {
        shift_reg[i]=shift_reg[i-1];
        acc+= shift_reg[i]*c[i];
 }
```

Figure 7: Original FIR filter in C++.

```
for (i=N-1;i>=0;i-=2) {
       shift_reg[i]=shift_reg[i-2];
       shift_reg[i-1]=shift_reg[i-3];
       acc+=shift_reg[i]*c[i];
       acc+=shift_reg[i-1]*c[i-1];
       }
```

Figure 8: Partially unrolled FIR filter (factor=2).

Table 3 lists the latency and resource usage report for a 16-tap FIR filter implementation on OpenMili's BPU, with different loop unrolling factors. Suppose the FPGA is clocked at 250 MHz, then HLS suggests an unrolling factor of 64 to achieve 4 samples/cycle, *i.e.*, 1 Gsps processing rate.

3.1.2 Using AXI to Facilitate Modular Design

OpenMili adopts the Advanced eXtensible Interface (AXI) [15] — an on-chip interconnect standard for the connection and management of functional blocks in system-on-a-chip designs — as a generic mechanism to glue the DSP blocks. This design choice affords two critical properties to simplify the FPGA programming: *(i) Modularity:* allowing TAP, INSERT/DELETE of one DSP block without affecting others. *(ii) Rate insensitivity:* allowing faster DSP blocks to be connected directly, while using AXI to automate the rate-matching mechanism. In contrast, in conventional HDL based programming, the developer needs to have a global view of the entire DSP chain, manually assuring the operations of adjacent DSP blocks are synchronized at cycle level.

OpenMili employs two kinds of AXI buses: *AXI-Stream* and *AXI-Lite*. AXI-Stream is used to transport high speed samples sequentially between DSP blocks, JESD204 transceivers, PCIe DMA controllers, *etc.* AXI-Stream provides a maximum bus width of 2048 bits, and when running on the 250 MHz Xilinx Kintex UltraScale FPGA, it has 64 GB/s rate, sufficient for interconnecting Gsps DSP blocks. AXI-Stream provides two critical advantages in programming OpenMili: *(i) Standard inter-block interfaces.* In OpenMili, the HLS can directly augment a customized DSP block with the AXI-Stream interface. In addition, since all the DSP blocks follow the interfacing standard, one block can be substituted by others (DELETE/INSERT) by rewiring a line in the GUI toolkit. AXI-Stream also eases the incorporation of existing DSP IP libraries into a custom design. *(ii) Rate insensitive design.* All the AXI-Stream interfaces share the same bus clock (250 MHz in OpenMili). The HLS design tool will handle the hand-shake between DSP blocks and insert FIFO for buffering incoming/outgoing signal samples.

UnRolling Factor	Samples per cycle	DSP48 Usage
1(Unrolled)	1/16	1
2	1/8	2
4	1/4	4
8	1/2	8
16	1	16
32	2	32
64	4	64

Table 3: Latency and resource usage for 16 tap FIR filter.

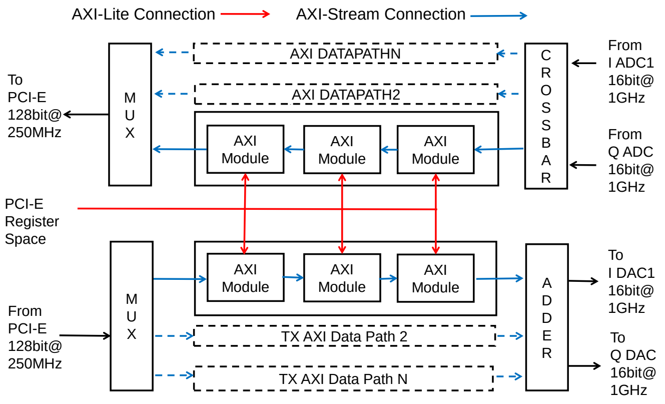


Figure 9: OpenMili's AXI-based baseband processing system.

Therefore, the programmer needs not worry about the inter-block synchronization or rate matching which is a common challenge in multi-rate DSP systems.

On the other hand, AXI-Lite is a light-weight, low-speed AXI protocol for memory address based register access. In OpenMili, we use AXI-Lite to control/reconfigure the low-level components, *e.g.*, sampling speed, center frequency, downsampling and upsampling ratio of ADC/DAC, and phase shift values of the phased-array antenna *etc.* All such configurations are stored in registers connected to the AXI-lite bus, and both the PC host and Microblaze processor can access the register through memory operations.

Figure 9 summarizes the usage of AXI inside OpenMili's software architecture. The inter-module connection for the PCIe interface (PC to BPU) and the JESD204B interface (BPU to ADC/DAC) are all implemented using AXI-Stream bus which can natively support INSERT/DELETE. This allows arbitrary DSP blocks to be connected to the host PC or data converters, in a compatible way as the inter-DSP-block connection. To support TAP, we add an AXI-Stream multiplexer between the PCIe DMA controller and DSP blocks. User can access the intermediate result of any DSP block by connecting the signal to the MUX, which reroute the signal to the PC host for observation. We also built a round-robin scheduler for the multiplexer, which works together with the PCIe driver (Sec. 3.2). It can transport the data from different blocks in a time-division manner, demultiplex the data after going through PCIe, and deliver them to user space C/MATLAB applications. Moreover, we use AXI-Lite to augment control functionalities on signal processing blocks, *e.g.*, MAC-level rate adaptation in communication applications and waveform selection in sensing applications.

Here we use a frequency-modulated continuous-wave (FMCW) Radar as an example to showcase the usage of AXI-based modular programming in OpenMili. Figure 10 illustrates a diagram of the FMCW Radar. The output frequency of a

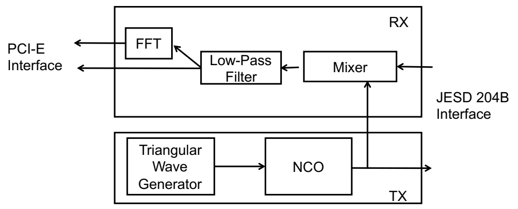
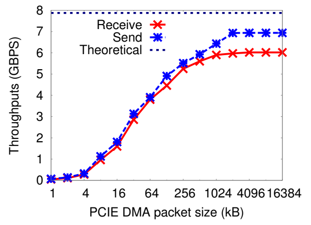


Figure 10: Prototyping an FMCW Radar using OpenMili's programming model.



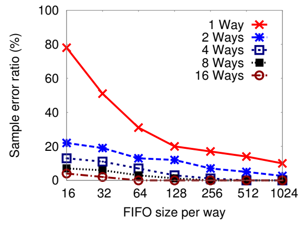


Figure 11: Throughput of PCIe driver between BPU and PC host.

Figure 12: Sample loss under different BPU buffer levels.

Numerically Controlled Oscillator (NCO) is controlled by a continuous-wave generator which could be configured to generate triangular wave, saw wave, and sine wave with different sweeping rate. With AXI-Stream, we can flexibly route the NCO output to both DAC (to emit continuous waves) and mixer (to obtain differential frequency). The received and mixed signals go through an FFT module for differential frequency analysis, and the results are passed to the PC host, to realize radar applications such as ranging, tracking, *etc.* All connections (arrows) here are realized using the modular AXI-Stream interface.

3.2 Fast Prototyping Using GPP

To enable Gsps sample transfer between the PC host and BPU, we develop the PCIe driver based on the PCIe 2.0 data streaming reference design from Xilinx [34]. The driver comprises a control plane and data plane. The control plane provides low speed channel to control the low-level components like DMA controller on FPGA, ADC and DAC, *etc.* The challenge in building the PCIe data plane lies in the mismatch between stream data and packet data — The data converter and DSP logic produce/consume stream data which have no start and ending, but the PCIe can only handle packetized data via bursty DMA transfer.

We overhauled the PCIe 2.0 reference design to enable PCIe 3.0 transportation, by upgrading the corresponding bus width from 128 bit to 256 bit, boosting the nominal bandwidth from 4GBps to 8GBps. We empirically found that larger DMA packet size leads to higher throughput (Figure 11), because it amortizes the per-packet coordination overhead between the host PC's DMA engine and the FPGA's DMA controller. To achieve the PCIe throughput requirement of 40 Gbps (Sec. 2.2), a minimum DMA packet size of 256kB is needed.

However, the FPGA is designed for computation rather than storage, so its FIFO has a very shallow buffer depth of 128 kB. We thus design a ring buffer whose control logic is implemented in the BPU. Figure 12 quantifies the sample error ratio (sum of sample loss and duplication ratio) when transferring random digital samples based on the ring buffer. We empirically choose the buffer configuration (256kB*8ways or 512kB*4ways) with minimal total buffer size.

As the CPU will replace the cache in the size of a cache line (typically 64kB), the typical discontinuous memory allocation will reduce the speed of memory access at the host side. Thus, we use cmem, a contiguous physical memory allocator [35] as a kernel space buffer between user applications and DMA engine, in order to maximize memory-DMA efficiency. As the cmem only has 128 MB of space, the kernel driver will automatically copy the data between cmem buffer and user memory. Consequently, the maximum sample length is determined by the system memory size. In our host PC, we allocate 24 GB memory for sample trace collection. At 1 Gsps sampling rate, this amounts to a trace length of 6 seconds in 16-bit I/Q channel mode and 24 seconds in 8-bit real-channel mode.

To support fast prototyping on the host PC, we implement a C API and a MATLAB API atop. The APIs include both data-stream interface (for reading and monitoring the DSP block), which reads/writes to the addresses of sample buffers, and AXI-Lite based register access interface (for hardware configuration like ADC/DAC and phased-array controller), which accesses the registers on the BPU. As MATLAB has its own data structure, a sample's real and image parts are not stored in continuous addresses. To reduce the impact of data copying, our MATLAB API accepts matrix whose column is the time and row is the content for different buffer. In this manner, the data order from/to MATLAB is consistent with the data order used by the DAC/ADC.

4. PROGRAMMABLE PHASED ARRAY ANTENNA

OpenMili's antenna array design aims for three features: *(i) Programmability:* supporting real-time switching between codebook entries corresponding to different beam patterns. *(ii) Scalability:* the number of antenna elements can easily scale up to support larger phased-array size, and hence more beam patterns with higher directionality. *(iii) Compatibility* with the WR-15 waveguide interface on the PEM-003 mmWave front-end.

Existing mmWave phased arrays are typically integrated on-chip antennas that interface with monolithic integrated circuits (MMICs) [36,37]. In contrast, as a standalone module, OpenMili's phased array faces unique challenges that entail customized design choices. *First*, the antenna elements need to be mounted on a planar substrate which needs proper transition to the WR-15 waveguide. At 60 GHz frequencies, significant insertion loss can occur when using a straightforward touch-and-mount transition. *Second*, to avoid grating lobes in the antenna array, the spacing between the adjacent antenna elements should be as small as wavelength-scale ($\sim$5 *mm*), otherwise significant amount of energy is radiated in undesired directions. Also, mmWave signals must be routed optimally under this tight dimension constraint. *Third*, to have a functional planar design at 60 GHz, the substrates with higher dielectric constants are very thin (a few microns). Therefore, mechanical stability support is needed when interfaced with the waveguide, as small misalignment can lead to severe signal loss.

Below we present OpenMili's phased array design meeting above challenges, and its beam pattern controller design to offer the programmability feature.

4.1 Phased-Array Antenna Design

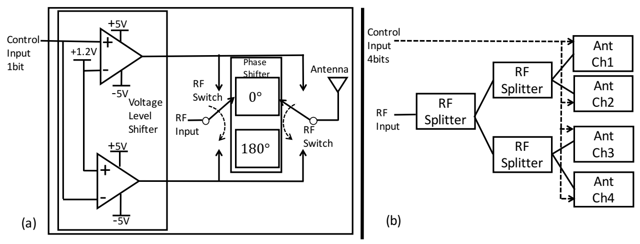


Figure 13: Architecture of OpenMili's phased-array antenna: (a) one antenna channel; (b) Four antenna channels.

4.1.1 Architecture

We design a linear array, with microstrip antenna elements driven by a phase-shifting network, as illustrated in Figure 13. The mmWave front end generates waveguide ended signal which goes through a *waveguide to microstrip transition* to make the signal *compatible* with the antenna's planar structure. Then, the signal propagates through a series of tree-structured power divider networks, with a 3 dB power division at each junction. The tree structure allows the phased array to be *scalable* to the 2^{th} power number of antenna elements.

To realize phase shifting, we use a discrete network with varying transmission line lengths in each signal branch, causing two different delays corresponding to two phase shift values of $0°$ and $180°$, respectively. A single pole double throw (SPDT) switch [38] is used to select the appropriate phase shift in each signal path. This signal further goes through a second stage of switches before feeding the antenna array element. Overall, the phase shifting network allows us to realize any codebook where individual antenna element's weight can be either e^0 or $e^{j\pi}$.

We emphasize that this architecture can be easily expanded to support a richer codebook and larger number of antenna elements. For example, one can use 60 GHz SP3T switches (*e.g.*, [39]) to add an additional phase shift branch. Alternative switches (*e.g.* [40]) exist that can also be used as a variable voltage attenuator. This will futher provide amplitude control for each antenna element path. We can cascade these switches with the existing ones in OpenMili such that more phase shifts and amplitude attenuation options can together create larger codebooks and hence more beams. On the other hand, to enlarge the phased array size, we can simply expand the binary-tree structured phase-shifting network, adding more branches and antenna elements.

4.1.2 Design and Fabrication

Waveguide to microstrip transition: At mmWave frequencies, a mature approach to waveguide-microstrip transition is to place a single substrate in between. However, this requires via holes connecting the grounds of bottom and top surfaces of the substrate. Such a requirement is easily satisfied for integrated antennas but prone to manufacturing error for OpenMili's standalone phased array. Therefore, we adopt a via-less design for its simple construction, low cost and ease of fabrication. As shown in Figure 14, we use a rectangular patch element on the bottom surface of the substrate which couples energy to a microstrip line on the top surface of the substrate. In order to have a short circuit at the edge of the waveguide, an open circuit is created at a

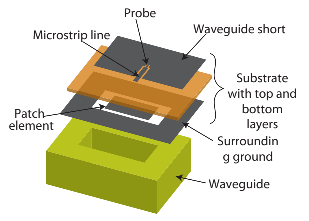
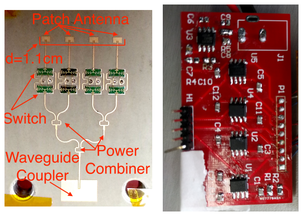


Figure 14: Schematic of waveguide to microstrip transition [41].



(a) (b)

Figure 15: (a) Phased array antenna; (b) antenna controller board.

$\lambda_g/4$ distance in the parallel plate waveguide [41].

Power divider network: Power divider network is a classical routing structure inside mmWave antenna arrays. A variety of power divider networks have been presented in the literature, *e.g.*, Lange [42], simple T-junction [43] or a Wilkinson [44]. We adopt the Wilkinson power divider without the isolation resistor, considering its low loss, as its distribution uses simple mitered curve with only one width across the power divider network, in contrast to other schemes that divide power by splitting a wide signal trace into narrower ones which causes loss [45].

Phase shifting network and antenna elements: Microstrip patch is used for each antenna element due to its low cost, low profile and ease of fabrication for mmWave front end module. A 4-element linear array of microstrip patch antennas is designed. An inset fed technique is used for impedance matching. Spacing between adjacent antenna elements is 0.6λ, which is selected based on the chosen phase shifting and switching network. Two electrical delay lines are designed for each arm of the antenna feed network, such that they present a phase shift of 180°. Single pole double throw (SPDT) switches [38] are used to select between these two phase shifts for each antenna element. The selection of a particular phase shift for each antenna element is carried out using control logic implemented in the BPU (more details in Sec. 4.2).

Substrate, fixture, and fabrication: Choosing the right dielectric medium at mmWave frequencies is vital to optimal performance. The electromagnetic waves can be guided not only with conductor configurations but also along dielectric layers, giving rise to surface waves resulting in performance degradation. In addition, low impedance line results in wider strips, which also tends to lower the cut-off frequencies of undesirable higher-order modes within or near the operating frequency range. Therefore, to avoid higher order modes and the propagation of surface waves, we use thin substrates. Empirical methods [46] have been proposed that relate the substrate's thickness and dielectric properties to its operating frequency. We used Rogers 6010.2LM substrate [47] with a thickness of 5 mils (0.127 mm), which has a maximum operating frequency of about 74 GHz. To offer mechanical stability, we also designed a fixture made of aluminum, and attached in between the phased-array and the WR-15 waveguide.

Before fabrication, the phased array design is simulated using a 3D EM solver software, CST Microwave Studio, which takes as input the geometry and dielectric properties of the phased-array's constituting materials. Figure 15(a) illustrates the fabricated phased-array. We will compare the array's simulated gain pattern with measurement in Sec. 5.3.

4.2 Programmable Phased Array Controller

In OpenMili, antenna beam-switching command can be initiated by the PC host or Microblaze processor (programmable in C), which in turn controls the FPGA's GPIO and drives the voltage change of each switch in the phase-shifting network. The phased array controller needs a 4-bit input into the phase-shifting network to generate the 16 beam patterns. Our implementation extends the Xilinx AXI-to-GPIO IP core to enable the Microblaze/PC to control the antenna switches. The Microblaze/PC uses the AXI-Lite interface (Sec. 3.1.2) to write a 32-bit word (supporting up to 32 switches) to a register with a predefined 32-bit bus address. The register switches the voltage of the GPIO ports, whose output will be updated at the next rising edge of the AXI bus clock.

Note that each switch is a PIN diode that has 22 mA forward current and needs to flip between positive and negative voltages, so we cannot directly interface it with the FPGA's GPIO which is 1.8V unipolar and bears 8 mA drive current. We thus build a customized phased-array controller board (Figure 15(b)), which can be plugged into the FPGA's PMOD interface – a standard interface for connecting with peripheral modules. Figure 13 illustrates a diagram of the phased-array controller, with three main functions: *(i)* Convert the positive power supply into negative power supply; *(ii)* Convert the GPIO's 0V-1.8V unipolar signal into -5V to +5V bipolar signal and provide the 22 mA forward current; *(iii)* Provide enough output current. More specifically, we use the MAX889 negative power IC [48] to generate the negative power supply, and AD8001 [49] as a bipolar comparator to convert the 1.8V IO, in order to meet the voltage and current requirement of the antenna switch.

5. REFERENCE BLOCKS AND EXPERIMENTAL VALIDATION

In this section, we present the design and experimentation of a few reference blocks, which verify the performance of OpenMili's hardware architecture, as well as the effectiveness of its programming models in prototyping communication and sensing applications.

5.1 Gbps Baseband Communication System

Prototyping the 802.11ad baseband communication system. We have built a real-time Gsps baseband communication system on OpenMili, which implements the majority of the 802.11ad PHY-layer [50]. Figure 16 shows the system architecture. Along the transmitter path, we can generate data bits from either the PC host or BPU. The data are patched with CRC checksum and then sent to a convolution encoder (supporting rate 1/2, 2/3, 3/4 and 7/8). Coded binary bits are modulated into analog waveforms through a mapper and IFFT module, which supports both single-carrier-FDMA and OFDM as specified in 802.11ad. Before routing the waveform to the DAC, we prepend known preambles which form a complete packet. Along the receiver path, we have implemented preamble-based packet detection, synchronization, and channel estimation modules. The detected packet is demodulated following a reverse path as the transmitter's.

All the DSP blocks run on the BPU, and are prototyped following the programming model in Sec. 3. We implement (TWEAK) each DSP block using HLS, and connect them us-

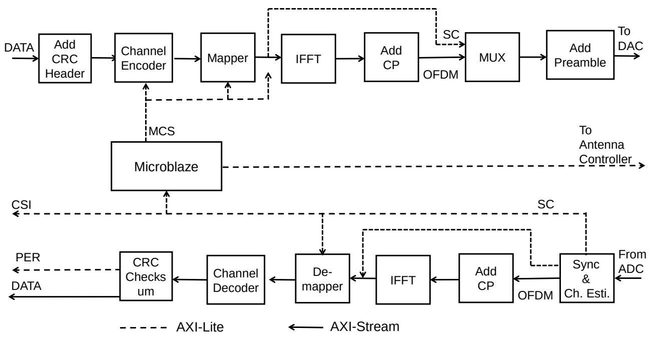


Figure 16: Baseband communication system reference design.

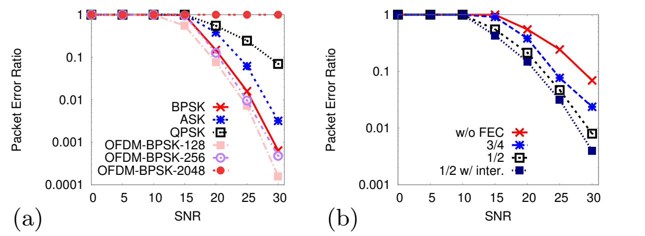


Figure 17: Packet error ratio (a) under different modulation schemes and (b) coding schemes.

ing AXI-Stream for automatic rate matching. Moreover, we prototype simple monitoring (TAP) functions, which compute performance statistics, *e.g.*, packet error rate (PER), and store them in an AXI-Lite register. The register can be accessed by both the host PC and Microblaze following the standard AXI-Lite read operations. Further, the decoded data bits can be forwarded to the host PC in real-time through the PCIe driver that we developed.

We have also implemented basic rate adaptation and beam adaptation mechanisms within this reference communication system. Basically, the receiver can compute a received signal strength (RSS) value based on the packet preamble. The RSS is stored in a register inside the BPU with pre-defined memory address. The Microblaze can read the register and then determine the modulation/coding scheme (MCS) using any customized rate adaptation protocols. Further, it can switch across different beam directions by commanding the phased-array controller (Sec. 4.2). It then measures the RSS of each and choose the best direction. A more sophisticated MAC-level signaling mechanism as in 802.11ad [50] can be implemented but is beyond the scope our reference design.

Experimental validation. We verify the reference design by measuring the performance of different MCS under different SNR levels (created by configuring OpenMili's RF gain values). By default, we send packets of size 1KB, and fix the antenna beam to a single direction. Figure 17(a) plots the measured SNR-PER curve, averaged over 10^4 packets in each run. We observe that a relatively high SNR threshold of 15 dB is needed to achieve a reasonable PER performance (< 0.1), since channel coding is not executed here. In general, OFDM achieves higher performance than the single carrier modulation, because it can absorb the multipath reflections using the cyclic prefix (CP) mechanism [11]. In addition, given a fixed bandwidth, a larger number of subcarriers result in much worse performance. For example, at 25 dB SNR, OFDM with 256 subcarriers (OFDM-BPSK-256) results in PER=0.01, whereas OFDM with 2048 subcarriers has PER≈1. This is mainly due to the inherent phase-noise in 60 GHz radios, which cannot be completely eliminated since even small phase noise at the reference clock will be amplified by orders of magnitude at 60 GHz (Sec. 2.3). Phase noise "smears" the frequency spectrum of a subcarrier, resulting in inter-carrier interference, which worsens the bit errors.

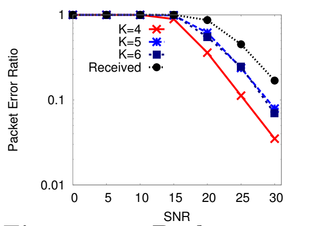


Figure 18: Packet error ratio comparison with ideal Rician channel model.

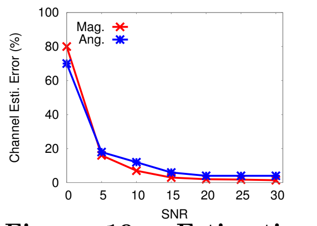


Figure 19: Estimation error of magnitude and phase in the sensing reference design.

Figure 17(b) shows the impact of coding and interleaving, focusing on QPSK without loss of generality. A higher channel coding rate improves PER performance, consistent with theoretical predictions [23]. Moreover, with a bit-interleaver (randomizing bit positions in the packet), bursty channel errors are reduced, and hence lower PER. In Figure 18, we further compare the PER performance with an ideal simulated Rician channel with different K parameters, where K denotes the ratio between the LOS component and NLOS component in a Rician model. The measured PER is between Rician curves with $K = 4$ and $K = 6$, and nearly overlaps with $K = 5$. On one hand, this implies that OpenMili's hardware/software modules are not introducing any artifacts that compromise the channel measurement and communication performance. On the other hand, it indicates that even if we use a single-beam directional antenna, non-negligible multipath effect still exist, likely caused by ambient reflections as well as unavoidable side lobes in the antenna gain pattern.

Table 4 lists the MCS levels supported in the reference design, by default using OFDM with 256 subcarriers. The corresponding SNR thresholds are obtained when measured PER reaches around 1%. As in legacy OFDM communication systems, we place null subcarriers near DC and the band edge, which confines the effective bandwidth to 800 MHz. Although the MCS is not comparable with 802.11ad, its spans from 12.5 Mbps to 1.3 Gbps, and already allows exploration of Gbps rate adaptation protocols.

5.2 Real-Time GHz RSS/Phase Sensing

OpenMili supports two modes of wireless sensing applications. In the *radar mode*, OpenMili serves as a programmable radar, where the transmitter emits 60 GHz signals while the receiver captures and processes the reflected signals. To isolate direct leakage between the Tx and Rx, we can either mount highly-directional horn antennas on the RF front-end, or place 60 GHz absorbers in between. An example radar prototype has been discussed in Sec. 3.1.2.

In the *communication system as a sensor* (CSAS) mode, the transmitter emits 802.11ad-like packet preambles, and the receiver estimates the channel information by processing the received preambles. This is equivalent to stripping off the modulation/coding blocks in the communication reference design. More specifically, following 802.11ad, we im-

MCS	Coding Rate	Mod.	Bitrate	SNR th.
0	1/2+32x spread	BPSK	12.5Mbps	3
1	1/2	BPSK	400Mbps	13
2	3/4	BPSK	600Mbps	15
3	13/16	BPSK	650Mbps	19
4	1/2	QPSK	800Mbps	21
5	3/4	QPSK	1200Mbps	24
6	13/16	QPSK	1300Mbps	25

Table 4: Capabilities of baseband reference design.

Codebook Entry	Mainlobe Direction	Mainlobe Gain (dBi)	Sidelobe Direction	Sidelobe Level (dB)
1,1,1,1	0°	14.3	+/-39.1°	2.46
1,1,1,-1	+/-15°	11.2	+/- 48°	5.47
1,1,-1,1	+/-36°	11.3	4°	8.31
1,1,-1,-1	+/- 19°	12.2	+/- 73°	-14.55
1,-1,1,-1	+/-36°	11.3	4°	8.31
1,-1,1,-1	+/- 47°	12.5	+/- 11.5°	4.5
1,-1,-1,1	+/-31°	12.4	+/- 66°	-4.67
1,-1,-1,-1	+/-15°	11.2	+/- 48°	5.47

Table 5: Direction and Gain of main lobe and largest sidelobe for each codebook entry

plement a preamble structure that uses 2176 samples for detection/synchronization and 1152 samples for per-subcarrier channel state information (CSI) estimation. With 1 Gsps sampling rate, the preamble duration becomes $3.4\mu s$. Upon receiving the preamble, the receiver can compute the CSI and store it in a register in one clock cycle ($4ns$), and the Microblaze can read the CSI in 12 clock cycles ($48ns$). Therefore, the overall CSI sensing latency predominantly depends on the preamble transmission time, *i.e.*, OpenMili can obtain the CSI of the entire 1 GHz channel every $3.4\mu s$.

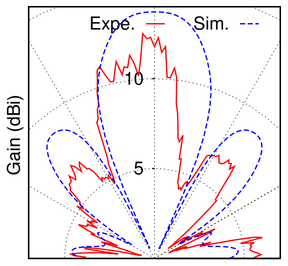
We evaluate the channel sensing accuracy by measuring the EVM across all subcarriers. Figure 19 plots the EVM across a wide range of SNR levels. At low SNR, the channel sensing errors are mainly caused by channel noise. Beyond 15 dB SNR, the phase error converges to around 4% and magnitude error to 1.5%. Such residual errors originate from the hardware noise in OpenMili, which induces a small amount of magnitude/phase variations.

5.3 Real-Time Phased-Array Controller

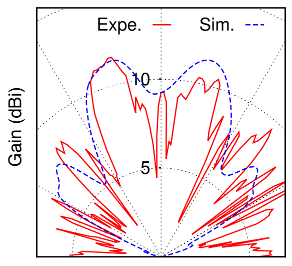
Table 5 lists OpenMili's phased-array beam patterns obtained based on the simulation method in 4.1. Due to codebook symmetry, we have 8 beam patterns in total and 5 unique ones. For codebook entry (1,1,1,1), there is only one mainlobe giving rise to a gain of 14.3 dBi. Whereas for other entries there are two mainlobes which split the gain.

To measure the actual beam pattern, we mount the phased array on a Tx and a 3.4° horn antenna on the Rx. We then use a step motor to rotate the phased array with 1 degree granularity, while using 60GHz RF absorbers to block reflection paths around the Tx/Rx. Figure 20 show the measured angular gains of codebook entries (1,1,1,1) and (1,1,-1,-1), which demonstrate consistent patterns with the simulation results. We have observed similar consistencies for other codebooks but omit the results due to space constraint.

To evaluate the phased-array's switching efficiency, we use a oscilloscope to monitor the input/output of our controller board and measure the rise and fall time. We observe that both have a similar rise/fall time of below 2.5 ns. Hence, the



(a)



(b)

Figure 20: Simulated and measured radiation pattern of: (a) codebook (1,1,1,1); (b) codebook (1,1,-1,-1).

stabilization time of our phased array antenna is below 2.5 ns, which is negligible compared with the channel estimation which is the latency bottleneck (Section 5.2).

6. CASE STUDIES

6.1 Pairwise Localization of 60 GHz Phased-Arrays

We propose a simple scheme that leverages the ultra-wide band (high time resolution) and discrete beam switching capability of 60 GHz phased-arrays to realize pairwise localization instead of multi-node triangulation [16, 17, 51]. This scheme targets the scenario where the Tx and Rx have a line-of-sight (LOS), which represents the most common cases for 60 GHz links. There exists mature ways to determine if the LOS condition is satisfied [16].

The localization scheme has two primitives: ranging and AoA estimation. We use a pulse delay method to estimate the LOS range between Tx and Rx, whose fundamental principle is similar to the FMCW radar. The Tx sends a known pseudo-random sequence and Rx runs a cross correlation and translates the peak position into time of flight estimation. Using OpenMili's Gsps sampling capability, we can achieve 1 ns timing resolution which gives a 30 cm ranging resolution.

To estimate the AoA, we face two challenges unseen in prior work [16, 17]: the limited number of discrete beam patterns and the irregular beam pattern with imperfect directionality. However, we can harness the measured gain pattern of the few beam patterns as follows. For simplicity, suppose the Tx uses a fixed beam pointing to the Rx, which is necessary for establishing a communication path. The radiation pattern of a beam can be obtained at factory calibration time. Specific to OpenMili's phased-array, we have measured the radiation patterns with 1° granularity (Sec 5.3). So we have 180 gain vectors, each consisting of the gains of all 5 effective beam patterns at a specific angle. We can derive the signal power at the antenna from pathloss model [52] using the ranging result. We then add it to the gain vectors to get the RSS estimation vector at each direction. When conducting the localization, we measure the RSS vector by switching across all beam directions. Finally, we run a vector matching between the measured RSS vectors and the calibrated RSS vector in the foregoing step. The direction of the best match is used as the AoA estimation. Given the range and AoA, the Tx and Rx's relative position is readily available.

We have quickly prototyped the pairwise localization scheme

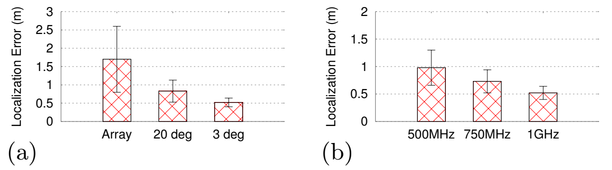


Figure 21: Localization error with (a) phased-array antenna and horn antennas with different beamwidth; (b) different bandwidth.

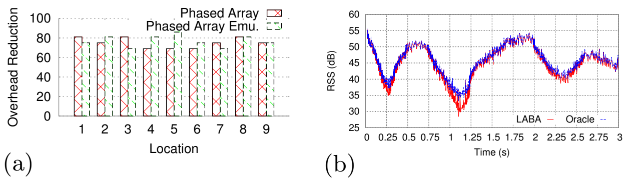


Figure 22: (a) Reduction of beam search overhead and (b) RSS under blockage events (Oracle switches across all beam directions in real-time and finds the best).

using OpenMili's GPP programming model. We test its effectiveness by placing the Tx and Rx at random locations in a $11m \times 8m$ indoor environment. Besides the phased-array, we also use two different sets of horn antennas (beamwidth $A_1 = 20°$ and $A_2 = 3.4°$) as benchmark, which are steered using a mechanical rotator (with granularity of π/A_1 and π/A_2). These horn antennas can be considered as ideal phased-arrays with slow steering but without any side lobes. For a given Tx, we measure the location error between estimated and ground-truth Rx spot.

Our experimental results in Figure 21 (a) show the average error is 0.51m, 0.72m, 1.63m, for the 3.4°, 20° horn antenna, and phased-array antenna, respectively. The phased-array has larger error due to its limited number of beam directions, and also the multipath effects created by imperfect beam patterns (esp. sidelobes). Figure 21 (b) further shows the impact of sampling bandwith. A wider bandwidth translates into higher time resolution, and hence the Gsps bandwidth achieves 0.52m mean error, in contrast to 0.99m at 500 MHz.

6.2 Learning Assisted Beam Adaptation (LABA)

A dominating indoor use case of 60 GHz networks (*e.g.*, 802.11ad) lies in cordless computing, *i.e.*, replacing Gbps Ethernet and monitor cables with a 60 GHz wireless links [53]. However, due to heavy human activities, such links are prone to blockage. Although electronic beam-steering is designed into 802.11ad to detour blockage, searching for the best beam direction involves non-trivial signaling overhead, especially considering the up to hundred-scale beam patterns that can be generated by a 60 GHz phased-array [54].

We propose LABA, a simple learning assisted protocol to reduce the beam search space and hence the overhead. LABA is applicable to quasi-stationary links, which may be blocked frequently, but moved only occasionally. Although a node may generate many beam patterns, the gain patterns of different beams may be partially overlapping and correlated across spatial angles [55]. Therefore, LABA learns the correlation offline, and prunes the correlated beams during run-time beam searching. Specifically, during the learning phase, a person stands at a random position p between the Tx and Rx to block the link, while we use OpenMili to collect each beam pattern j's RSS, denoted as S_p^j. Suppose we create P random blockage positions in total, then for each beam pattern j, we have a vector of measured RSS $[S_1^j, S_2^j, \cdots, S_P^j]$ across all blockage positions. We then compute the pairwise cross correlation between the RSS vectors. At run time, whenever severe rate drop occurs, LABA initiates beam steering just like a normal 802.11ad protocol, but only searches across the beam patterns that have < 0.5 correlation with the current one.

We have implemented LABA's learning module on OpenMili's GPP module, and run-time adaptation mechanism on its real-time phased-array controlling module (on the Microblaze). To show its effectiveness, we test it across 9 different locations in 3 environments with different reflection characteristics: an office, corridor and student lab. We use OpenMili's phased-array hardware, as well as a phased-array emulation method [55], which first uses a 3.4° horn antenna to collect the channel AoA pattern and then convolve it with the angular gain pattern of an ideal 4 element linear array. As shown in Figure 22(a), LABA can consistently prune the search space and reduce the beam searching overhead by around 70%, compared with an exhaustive search. The reductions is slightly lower than the emulated phased-array, which assumes isotropic radiation patterns for each antenna element and thus causing more overlapping/correlation across beam patterns. Due to limited number of antenna elements, we are unable to generate quasi-omni beam patterns to run the 802.11ad-compatible hierarchical beam-searching, but LABA's pruning method can be readily used to prune any existing beam searching method.

In Figure 22(b), we show the RSS traces across 3 human blockage events. We see that LABA demonstrate a similar level of RSS as the exhaustive search, implying it either chooses the best beam direction or choose one with similar RSS. This effectively verifies that LABA reduces beam searching overhead without sacrificing link performance. It also verifies that OpenMili can switch across all beam patterns without any noticeable latency.

7. RELATED WORK

Software Radio Platforms. In the past a few years, a variety of software-radio platforms have been proposed to facilitate the development of new wireless protocols. These platforms share a similar high-level architecture: they connect a PC host with an external baseband processing unit (typically an FPGA), which attaches ADC/DAC and subsequently an RF front-end. Different platforms may partition the processing tasks between the BPU and PC host differently, to make a tradeoff between flexibility and performance. On one end of the spectrum lies the *purely software-defined frameworks*. The well-known USRP/GNURadio [9, 33] platform, for example, runs PHY-layer modulation/demodulation algorithms entirely on the PC host's general processor (GPP). Despite its flexibility, USRP/GNURadio suffers from long processing latency and PC-to-radio interface latency, which precludes real-time MAC operations. A more powerful system, Sora [11] applies sophisticated algorithm-specific parallelization (*e.g.*, core dedication) to achieve similar throughput as commercial 802.11b radios. On the other hand, most recent software-radio platforms adopt a *hybrid framework*. For example, WARP [10] allocates intensive signal processing functions to the FPGA, and light-weight control functions to the Microblaze or PC processor.

In terms of software framework, OpenMili is heavily inspired by WARP. The main difference is that WARP uses Matlab Simulink to define signal processing blocks, whereas OpenMili uses C++ based HLS. More importantly, WARP hard-wires the blocks, and hence, the developer must have precise estimation of the processing latency of each block. This hinders *modular refinement*, because any modification must take into perspective the timing of the entire signal processing flow. In contrast, OpenMili separates the signaling between blocks using AXI, which can essentially buffer adjacent blocks' input/output samples, thus decoupling their rate-dependency. This principled design borrows from a rich history in DSP systems implemented as a composition of blocks. For example, Airblue [56] and Atomix [30] designed customized FIFO buffers to enable modularized programming of DSP blocks. We note that, when developing PHY-layer signal processing systems, all these software-radio systems, including OpenMili, may use HLS to automate the intra-block parallelization. But they still require the developer to have knowledge about the relation between blocks, and to manually partition the blocks to maximize parallelism. This is particularly important for OpenMili with Gsps of processing load on FPGAs that are clocked at sub-GHz.

As for hardware architecture, OpenMili distinguishes itself in its Gsps sampling/processing rate and programmable phased-array, running at the 60 GHz carrier frequency. The challenges in enabling these features have been covered in previous sections. Some recent studies customized 60 GHz testbeds [55, 57, 58], using WARP or signal generator (network analyzer) as BPU. These testbeds do not capture the two unique features of 60 GHz radios, although they have similar or higher cost than OpenMili[1]. Arnold [59] and Zetterberg *et al.* [60] developed customized RF front-end which, together with USRP, form a programmable 60 GHz platform. But the platform has a low sampling rate limited by USRP's narrow band, and it uses fixed horn or on-chip antennas. National Instruments recently released a mmWave experimental platform based on their PXI baseband processing unit [61]. The platform runs on the 71GHz–76GHz spectrum and costs \$134K — almost 5× compared with OpenMili. Other commercial platforms (*e.g.*, Keysight's [62]) have similar limitations. All in all, OpenMili represents the first reconfigurable platform to support 60 GHz wireless protocols where real-time phased-array beamforming acts as a predominant feature to overcome blockage/mobility [55,63]. It is also the first to enable novel 60 GHz sensing applications that capitalize on the electronically steerable phased-array and GHz sampling bandwidth.

We note that the RFIC industry have mature ways to manufacture on-chip 60 GHz phased-arrays [64–66]. Yet, it entails non-trivial challenges to design a programmable phased-array using discrete phase-switches and interfacing it with the WR-15 waveguide form-factor. We believe OpenMili's phased-array design can be a standalone contribution, especially considering the fact that it can be detached from OpenMili and applied to commercial/customized 60 GHz radios with the WR-15 antenna interface [14, 55, 57, 58].

[1]The material/fabrication cost of OpenMili's phased array is ∼\$1.5K, similar to the horn antenna in [55,57,58]. OpenMili uses the same RF front-end as [55,57,58]. Its BPU/converters have similar cost as WARP (∼\$5K), much cheaper than a network analyzer.

Phased-array based location sensing. Phased-array has shown potential in radio localization, especially due as it can identify AoA via multi-antenna signal processing algorithms [67, 68]. Recent systems [17, 51] renovated such algorithms to isolate LOS paths, and can localize a client via multi-AP triangulation. However, these systems commonly use radios with a digital phased-array which can continuously adjust phased-shift values. In contrast, practical millimeter-wave radios [50,69,70] only allow a set of discrete phase shifters, which entails new AoA processing. We have explored a basic localization mechanism on such a practical phased-array, which not only senses AoA, but also distance between a pair of 60 GHz phased-arrays. This can be a primitive component for a wide-range of 60 GHz sensing applications, *e.g.*, location-aware beam-steering [71].

Beam adaptation protocols in directional-antenna networks. Low-frequency directional-antenna based MAC or routing protocols have been extensively explored in adhoc networks [72]. At the 60 GHz frequency band, many new challenges emerge, especially due to larger scale phased array with up-to hundred-scale antenna elements, and hence higher directionality. Also, shorter wavelength causes more vulnerability to obstacle blockage. Many PHY/MAC protocols [73–76] have been proposed to improve the efficiency of beamsteering and overcome blockage, yet they only build on simulation. They also use arbitrarily designed codebooks, whereas practical phased arrays need to respect fabrication constraints. In addition, large phased-array means large number of beam patterns to adapt which, combined with available bit-rate levels, substantially inflates the decision space when link quality degrades. Our case study of LABA exploits the practical tradeoffs herein, and propose a learning-based framework to facilitate the decision making. This is of independent interest and can be a building block for a wide range of MAC/application level protocols. Our recent project, BeamSpy [77] addressed a similar problem of overcoming blockage for 60 GHz links. BeamSpy adopts a sophisticated channel model that characterizes the beam correlation by explicitly modeling the magnitude, phase and angle of propagation paths. In contrast, LABA learns the correlation between different beam patterns from historical measurement. It treats the correlation function as a black box and learns it by training. In this sense, LABA can simplify the beam-quality prediction.

8. CONCLUSION

We have designed and implemented OpenMili, the first reconfigurable architecture that captures the unique aspects of 60 GHz radios, especially the Gsps sampling/processing rate, and electronically steerable phased-array antenna. OpenMili will be a ready-to-use, open-source platform to speed up the development of next-generation Gbps wireless protocols and sensing applications. Moreover, we believe our experience in the hardware/software framework will be transferable to future high-performance mmWave software-radios.

Acknowledgements

We appreciate the insightful comments and feedback from the anonymous reviewers. The work reported in this paper is supported in part by the NSF under Grant CNS-1506657, CNS-1518728, CNS-1343363, and CNS-1350039.

9. REFERENCES

[1] T. Rappaport, S. Sun, R. Mayzus, H. Zhao, Y. Azar, K. Wang, G. Wong, J. Schulz, M. Samimi, and F. Gutierrez, "Millimeter Wave Mobile Communications for 5G Cellular: It Will Work!" *IEEE Access*, vol. 1, 2013.

[2] Qualcomm, "Qualcomm VIVE 802.11ad ," https://www.qualcomm.com/products/vive/11ad, 2015.

[3] Intel, "Intel Tri-Band Wireless-AC 18260," 2015. [Online]. Available: http://www.intel.com/content/dam/www/public/us/en/documents/product-briefs/tri-band-wireless-ac17265-brief.pdf

[4] T. Wei and X. Zhang, "mTrack: High-Precision Passive Tracking Using Millimeter Wave Radios," in *Proc. of ACM MobiCom*, 2015.

[5] Y. Zhu, Y. Zhu, B. Y. Zhao, and H. Zheng, "Reusing 60GHz Radios for Mobile Radar Imaging," in *Proc. of ACM MobiCom*, 2015.

[6] Q. Pu, S. Gupta, S. Gollakota, and S. Patel, "Whole-home Gesture Recognition Using Wireless Signals," in *Proc. of ACM MobiCom*, 2013.

[7] K. Joshi, D. Bharadia, M. Kotaru, and S. Katti, "WiDeo: Fine-grained Device-free Motion Tracing Using RF Backscatter," in *Proc. of USENIX NSDI*, 2015.

[8] Google, "Project Soli," 2015.

[9] Ettus Research LLC, "Universal Software Radio Peripheral (USRP)," http://www.ettus.com/.

[10] A. Khattab, J. Camp, C. Hunter, P. Murphy, A. Sabharwal, and E. W. Knightly, "WARP: a Flexible Platform for Clean-Slate Wireless Medium Access Protocol Design," *SIGMOBILE Mob. Comput. Commun. Rev.*, vol. 12, 2008.

[11] K. Tan, J. Zhang, J. Fang, H. Liu, Y. Ye, S. Wang, Y. Zhang, H. Wu, W. Wang, and G. M. Voelker, "Sora: High Performance Software Radio Using General Purpose Multi-core Processors," in *Proc. of USENIX NSDI*, 2009.

[12] IEEE, "802.11x ," http://www.ieee802.org/11/, 2015.

[13] D. Halperin, W. Hu, A. Sheth, and D. Wetherall, "Tool Release: Gathering 802.11N Traces with Channel State Information," *SIGCOMM Comput. Commun. Rev.*, vol. 41, no. 1, 2011.

[14] HXI LLC, "HXI GigaLink E-Band 60 GHz Radio Systems," 2013. [Online]. Available: http://www.hxi.com/D_Radios.asp

[15] Xilinx, "AXI Reference Guide," http://www.xilinx.com, 2015.

[16] J. Xiong and K. Jamieson, "ArrayTrack: A Fine-grained Indoor Location System," in *Proc. of USENIX NSDI*, 2013.

[17] K. Joshi, S. Hong, and S. Katti, "PinPoint: Localizing Interfering Radios," in *Proc. of USENIX NSDI*, 2013.

[18] X. Zhang and P. Ramanathan, "WiMi: Wisconsin Millimeter-wave Software Radio," 2016. [Online]. Available: http://xyzhang.ece.wisc.edu/wimi/

[19] "60 GHz Transmit/Receive Development Systems," http://www.pasternack.com, 2014.

[20] Analog Devices, "AD-FMCDAQ2-EBZ Evaluation Board," 2015. [Online]. Available: http://www.analog.com/en/design-center/evaluation-hardware-and-software/evaluation-boards-kits/eval-ad-fmcdaq2-ebz.html

[21] ——, "AD9680 14-Bit, 1.25 GSPS/1 GSPS/820 MSPS/500 MSPS JESD204B, Dual Analog-to-Digital Converter," 2015. [Online]. Available: http://www.analog.com/en/products/analog-to-digital-converters/high-speed-ad-10msps/ad9680.html

[22] ——, "AD9144 Quad, 16-Bit, 2.8 GSPS, Digital-to-Analog Converter," 2015. [Online]. Available: http://www.analog.com/en/products/digital-to-analog-converters/da-converters/AD9144.html

[23] S. W. Smith, *The Scientist & Engineer's Guide to Digital Signal Processing.* California Technical Publishing, 1997.

[24] Minicircuits, "RF Transformer ADT2-1T," http://www.minicircuits.com/pdfs/ADT2-1T.pdf, 2016.

[25] Analog DEvices, "AD9144 Quad, 16-Bit, 2.8 GSPS, TxDAC Digital-to-Analog Converter," http://www.analog.com.

[26] Pasternack Enterprises, Inc., "60 GHz Transmitter Waveguide Module," 2014. [Online]. Available: http://www.pasternack.com/60-ghz-transmitter-module-pem001-p.aspx

[27] USBPcap Developers, "Open Source USB Packet Capture," http://desowin.org/usbpcap/, 2016.

[28] B. Schiek, H.-J. Siweris, and I. Rolfes, *Noise in High-Frequency Circuits and Oscillators.* Wiley, 2006.

[29] Analog Devices, "ADF4351 Evaluation Board," http://www.analog.com.

[30] M. Bansal, A. Schulman, and S. Katti, "Atomix: A Framework for Deploying Signal Processing Applications on Wireless Infrastructure," in *Proc. of USENIX NSDI*, 2015.

[31] Xilinx, "UltraFast High-Level Productivity Design Methodology Guide," http://www.xilinx.com.

[32] Altera, "Spectra-Q Engine," https://www.altera.com, 2015.

[33] GNURadio, "The GNU Software Radio," http://gnuradio.org/trac/wiki.

[34] Xilinx, "KCU105 PCI Express Streaming Data Plane TRD," http://www.xilinx.com.

[35] Texus Instrument., "Cmem overview." [Online]. Available: http://processors.wiki.ti.com/index.php/CMEM_Overview

[36] S. Lin, K. B. Ng, H. Wong, K. M. Luk, S. S. Wong, and A. S. Y. Poon, "A 60GHz Digitally Controlled RF Beamforming Array in 65nm CMOS With Off-Chip Antennas," in *IEEE Radio Frequency Integrated Circuits Symposium (RFIC)*, 2011.

[37] A. Valdes-Garcia, S. Reynolds, A. Natarajan, D. Kam, D. Liu, J. W. Lai, Y. L. O. Huang, P. Y. Chen, M. D. Tsai, J. H. C. Zhan, S. Nicolson, and B. Floyd, "Single-Element and Phased-Array Transceiver Chipsets for 60-GHz Gb/s Communications," *IEEE Communications Magazine*, vol. 49, no. 4, 2011.

[38] Analog Devices, "SPDT, Reflective Switch Chip, 55 - 86 GHz," http://www.analog.com/media/en/technical-documentation/data-sheets/hmc-sdd112.pdf, 2015.

[39] TriQuint Semiconductor, "60 - 90 GHz SP3T Switch Flip Chip," 2016. [Online]. Available: http://www.triquint.com/products/p/TGS4305-FC

[40] Gotmic, "Multifunction switch chips," 2016. [Online]. Available: http://www.gotmic.se/switches.html

[41] Kazuyuki SEO, "Study of Planar Microstrip-to-Waveguide Transitions in Millimemter-Wave Band," 2011. [Online]. Available: http://repo.lib.nitech.ac.jp/bitstream/123456789/2285/1/ko2010_Seo.pdf

[42] J. Lange, "Interdigitated Strip-Line Quadrature Hybrid," in *G-MTT International Microwave Symposium*, 1969.

[43] D. M. Pozar, *Microwave Engineering.* John Wiley & Sons, 2009.

[44] E. J. Wilkinson, "An N-Way Hybrid Power Divider," *IRE Transactions on Microwave Theory and Techniques*, vol. 8, no. 1, 1960.

[45] F. D. L. Peters, D. Hammou, S. O. Tatu, and T. A. Denidni, "Modified Millimeter-Wave Wilkinson Power Divider For Antenna Feeding Networks," *Progress In Electromagnetics Research Letters*, vol. 17, 2010.

[46] Orfanidis, Sophocles J., "Electromagnetic Waves and Antennas," 2014. [Online]. Available: http://www.ece.rutgers.edu/~orfanidi/ewa/orfanidis-ewa-book.pdf

[47] Rogers Corporation, "Rogers 6010 Datasheet," https://www.rogerscorp.com/documents/612/acs/RT-duroid-6006-6010LM-laminate-data-sheet.pdf, 2014.

[48] Maxim, Inc., "MAX889 High-Frequency, Regulated, 200mA, Inverting Charge Pump," https://www.maximintegrated.com/en/products/power/charge-pumps/MAX889.html, 2014.

[49] "AD8001 800 MHz 50 mW urrent Feedback Amplifier,"

2014. [Online]. Available: http://www.analog.com/en/products/amplifiers/operational-amplifiers/current-feedback-amplifiers/ad8001.html

[50] IEEE Standards Association, "IEEE Standards 802.11ad-2012: Enhancements for Very High Throughput in the 60 GHz Band," 2012.

[51] S. S. Hong and S. R. Katti, "DOF: A Local Wireless Information Plane," in *Proc. of ACM SIGCOMM*, 2011.

[52] A. Maltsev, V. Erceg, E. Perahia, C. Hansen, R. Maslennikov, A. Lomayev, A. Sevastyanov, and A. Khoryaev, "Channel Models for 60 GHz WLAN Systems," *IEEE 802.11-09/0334r8*, 2010.

[53] IEEE 802.11ad Working Group, "Consolidation of Usage Models in 802.11ad," http://mentor.ieee.org/802.11/dcn/07/11-07-2988-04-0000-liaison-from-wi-fi-alliance-to-802-11-regarding-wfa-vht-study-group-consolidation-of-usage-models.ppt, 2015.

[54] S. Zihir, O. D. Gurbuz, A. Karroy, S. Raman, and G. M. Rebeiz, "A 60 GHz Single-Chip 256-Element Wafer-Scale Phased Array With EIRP of 45 dBm Using Sub-Reticle Stitching," in *IEEE Radio Frequency Integrated Circuits Symposium (RFIC)*, 2015.

[55] S. Sur, V. Venkateswaran, X. Zhang, and P. Ramanathan, "60 GHz Indoor Networking through Flexible Beams: A Link-Level Profiling," in *Proc. of ACM SIGMETRICS*, 2015.

[56] M. C. Ng, K. E. Fleming, M. Vutukuru, S. Gross, Arvind, and H. Balakrishnan, "Airblue: A System for Cross-layer Wireless Protocol Development," in *Proc. of ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS)*, 2010.

[57] T. Nitsche, A. B. Flores, E. W. Knightly, and J. Widmer, "Steering with Eyes Closed: mm-Wave Beam Steering without In-Band Measurement," in *Proc. of IEEE INFOCOM*, 2015.

[58] T. Nitsche, G. Bielsa, , I. Tejado, A. Loch, and J. Widmer, "Boon and Bane of 60 GHz Networks: Practical Insights into Beamforming, Interference, and Frame Level Operation," in *Proc. of ACM CoNEXT*, 2015.

[59] J. Arnold, L. Simic, M. Petrova, and P. Mähönen, "Demo: Spectrum-Agile mm-Wave Packet Radio Implementation on USRPs," in *Proc. of Workshop on Software Radio Implementation Forum (SRIF)*, 2015.

[60] P. Zetterberg and R. Fardi, "Open Source SDR Frontend and Measurements for 60-GHz Wireless Experimentation," *IEEE Access*, vol. 3, 2015.

[61] National Instruments, "mmWave Transceiver System," 2016. [Online]. Available: http://www.ni.com/sdr/mmwave/

[62] Keysight Technologies, "5G Waveform Generation and Analysis Testbed, Reference Solution," 2016. [Online]. Available: http://www.keysight.com/en/pd-2567072/5g-waveform-generation-and-analysis-testbed-reference-solution

[63] S. Singh, F. Ziliotto, U. Madhow, E. M. Belding, and M. Rodwell, "Blockage and Directivity in 60 GHz Wireless Personal Area Networks," *IEEE JSAC*, vol. 27, no. 8, 2009.

[64] S. Emami, R. Wiser, E. Ali, M. Forbes, M. Gordon, X. Guan, S. Lo, P. McElwee, J. Parker, J. Tani, J. Gilbert, and C. Doan, "A 60GHz CMOS Phased-Array Transceiver Pair for Multi-Gb/s Wireless Communications," in *IEEE International Solid-State Circuits Conference (ISSCC)*, 2011.

[65] S. Lin, K. Ng, H. Wong, K. Luk, S. Wong, and A. Poon, "A 60GHz Digitally Controlled RF Beamforming Array in 65nm CMOS With Off-Chip Antennas," in *IEEE Radio Frequency Integrated Circuits Symposium (RFIC)*, 2011.

[66] A. Valdes-Garcia, S. Nicolson, J.-W. Lai, A. Natarajan, P.-Y. Chen, S. Reynolds, J.-H. Zhan, D. Kam, D. Liu, and B. Floyd, "A Fully Integrated 16-Element Phased-Array Transmitter in SiGe BiCMOS for 60-GHz Communications," *IEEE Journal of Solid-State Circuits*, vol. 45, no. 12, 2010.

[67] R. Schmidt, "Multiple Emitter Location and Signal Parameter Estimation," *IEEE Transactions on Antennas and Propagation*, vol. 34, no. 3, 1986.

[68] R. Roy and T. Kailath, "ESPRIT-Estimation of Signal Parameters Via Rotational Invariance Techniques," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 37, no. 7, 1989.

[69] IEEE Standards Association, "IEEE Standards 802.15.3c-2009: Millimeter-wave-based Alternate Physical Layer Extension," 2009.

[70] ECMA International, "Standard ECMA-387: High Rate 60 GHz PHY, MAC and PALs," 2010.

[71] V. Navda, A. P. Subramanian, K. Dhanasekaran, A. Timm-Giel, and S. Das, "MobiSteer: Using Steerable Beam Directional Antenna for Vehicular Network Access," in *Proc. of ACM MobiSys*, 2007.

[72] O. Bazan and M. Jaseemuddin, "A Survey On MAC Protocols for Wireless Adhoc Networks with Beamforming Antennas," *IEEE Communications Surveys and Tutorials*, vol. 14, no. 2, 2012.

[73] B. Li, Z. Zhou, W. Zou, X. Sun, and G. Du, "On the Efficient Beam-Forming Training for 60GHz Wireless Personal Area Networks," *IEEE Transactions on Wireless Communications*, vol. 12, no. 2, 2013.

[74] J. Wang, Z. Lan, C. woo Pyo, T. Baykas, C.-S. Sum, M. Rahman, J. Gao, R. Funada, F. Kojima, H. Harada, and S. Kato, "Beam Codebook Based Beamforming Protocol for Multi-Gbps Millimeter-Wave WPAN Systems," *IEEE Journal on Selected Areas in Communications*, vol. 27, no. 8, 2009.

[75] K. Ramachandran, N. Prasad, K. Hosoya, K. Maruhashi, and S. Rangarajan, "Adaptive Beamforming for 60 GHz Radios: Challenges and Preliminary Solutions," in *ACM mmCom*, 2010.

[76] X. An, C.-S. Sum, R. Prasad, J. Wang, Z. Lan, J. Wang, R. Hekmat, H. Harada, and I. Niemegeers, "Beam Switching Support to Resolve Link-Blockage Problem in 60 GHz WPANs," in *IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, 2009.

[77] S. Sur, X. Zhang, P. Ramanathan, and R. Chandra, "BeamSpy: Enabling Robust 60 GHz Links Under Blockage," in *USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2016.