Dual-Mode PSK Transceiver on SDR With FPGA

Wuqiong Zhao [iD], *Student Member, IEEE*

(This is a draft version and has not been completed yet.)

Abstract—**In this experiment, we implement a dual-mode PSK transceiver on SDR with FPGA, supporting both BPSK and QPSK. Moreover, the transceiver is designed to be able to switch between the two modes by introducing packet-based communication, where modulation information can be extracted from the packet header.**

Index Terms—Phase-shift keying (PSK), software-defined radio (SDR), transceiver design, modulation, demodulation, field programmable gate array (FPGA).

I. INTRODUCTION

SOFTWARE-DEFINED radio (SDR) is interesting, like application in millimeter wave [1]. FPGA is also interesting!

Instead of employing high-level synthesis (HLS) [2], we directly implement the transceiver on FPGA using hardware description language (HDL) Verilog, for a better control of the underlying hardware.

The design source (Vivado project) and this paper (in LaTeX) are open source [3].

II. SYSTEM OVERVIEW

A. Software-Defined Radio

We employ eNodeX [4], an SDR platform equipped with a pair of configurable Global System for Mobile Communications (GSM) transmitter (Tx) and receiver (Rx) antennas.

B. Transceiver Design

The current transmitter and receiver are implemented on the same FPGA, but can be readily extended to different FPGAs with small frequency offsets. The system overview is shown in Fig. 1.

Clock Generator. Required clocks are generated from the PL clock. All reset signals are generated using Processor System Reset Modules [5], which can provide synchronized power-up reset signals.

RF Data Converter. This block is contains analog-to-digital converter (ADC) and digital-to-analog converter (DAC), enabled by a vendored AD9361 module.

Tx Signal Generator. The transmitted signal is generated in this block. Currently, it repeats a certain pattern and does not accept external input. But it can be readily extended to a more complicated design, based on the current interface template.
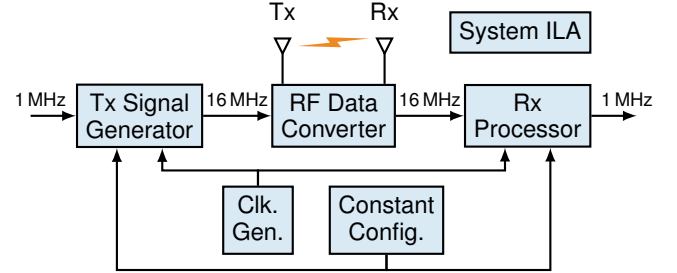
Fig. 1. Transceiver system overview.

Rx Processor. This block is responsible to process the received signal, including demodulation and data extraction from a packet. It is the most complicated block in the system.

System ILA. The system integrated logic analyzer (system ILA) [6] is used to observe the internal signals.

Constant Configurations. Several parameters can be configured in this block. Most importantly, the mode control constants (`MODE_CTRL`) are shown in Table I.

TABLE I
MODE CONTROL CONSTANTS

Mode	Localparam	Value	`is_bpsk`	Packet
BPSK	MODE_BPSK	4'b0001 (1)	1'b1	No
QPSK	MODE_QPSK	4'b0010 (2)	1'b0	No
Mixed	MODE_MIX	4'b0100 (4)	variable	Yes

As shown in Table I, the system can be configured to work in BPSK, QPSK, or the mixed mode. The remaining part will focus on the mixed mode, which has `MODE_CTRL` equal to `MODE_MIX` (`4'b0100`).

C. BPSK/QPSK Modulation

The BPSK and QPSK modulation constellation graphs used in our system are shown in Fig. 2. Different from the traditional setting, out adopted BPSK constellation in Fig. 2(a) is a combination of I and Q components. This is to make sure the phases used in BPSK are among those in QPSK, to ensure a smooth transition between the two modes, especially because the header field is always modulated in BPSK.

III. TRANSMITTER

A. Carrier NCO

The carrier frequency is generated by a numerically controlled oscillator (NCO). In Vivado, we use the Direct Digital Synthesis (DDS) Compiler IP core to generate the NCO. Both the cosine and sine components are used.
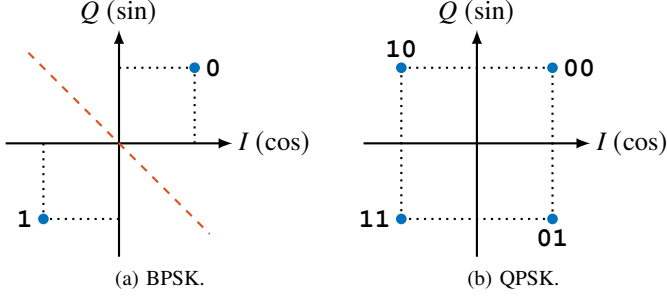
(a) BPSK.   (b) QPSK.

Fig. 2. BPSK/QPSK modulation constellation used in our system.

B. PSK Modulation

1) Pseudo-random Noise (PN) Generator: In this experiment, the transmitted signal are pseudo-random noise (PN) sequences. Typically, we implement the PN generator with $N = 4$ and $N = 5$ with a shift register [7]. The Verilog code for the module PN_Gen is shown below.

```verilog
module PN_Gen # (parameter N = 5) (
  input      clk,
  output reg pn
);
  reg [N-1:0] PN_buf = 1; wire rst;
  generate
  if (N == 5)
    always @ (posedge clk)
      if (rst) begin PN_buf <= 5'd1; pn <= 0; end
      else begin
        PN_buf <= { PN_buf[3:0], PN_buf[4] ^ PN_buf[2] };
        pn <= PN_buf[4];
      end
  else if (N == 4)
    always @ (posedge clk)
      if (rst) begin PN_buf <= 4'd1; pn <= 0; end
      else begin
        PN_buf <= { PN_buf[2:0], PN_buf[3] + PN_buf[2] };
        pn <= PN_buf[3];
      end
  else ; // NOT implemented yet!
  endgenerate
  assign rst = !(|PN_buf);  // reset when PN_buf is all 0
endmodule
```

2) Modulation With I and Q Streams: The modulation is performed by multiplying selecting the appropriate carrier phase according to the input bits and the constellation. Note the Q component always has a phase 90° ahead of the I component.

IV. RECEIVER

A. Overview

The receiver performs carrier synchronization, symbol synchronization and PSK detection first, without considering the packet structure. The depacketizer and the packet extraction is based on the synchronized bit stream after PSK detection.

B. Carrier Synchronization Using Costas Loop

A Costas loop [8] is used for carrier synchronization. The basic idea of a Costas loop is to provide an error (carrier phase offset) feedback. The negative feedback is used to lock the carrier phase to the PSK signal. The generation of the error feedback is specific to different modulation schemes, as well as their constellation graphs.

The proposed dual-mode Costas loop is shown in Fig. 3, which can be switched between BPSK and QPSK via a control signal is_bpsk. The two multipliers directly following the PSK stream input are phase detectors, whose low-frequency components represent the baseband signal. Therefore, the baseband I and Q signal can be extracted after passing through a low-pass filter (LPF). Thus, it is ready to map the obtained I/Q signal to the constellation graph, and generate the error signal when comparing with the designed one.
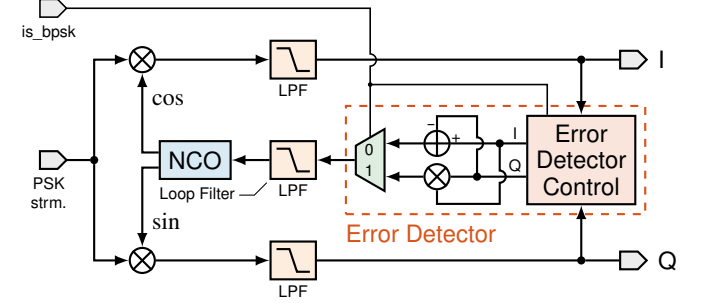

Fig. 3. Costas loop for carrier synchronization with BPSK/QPSK support.

The error feedback for BPSK is defined as

$$e_{\text{BPSK}} = (I + Q) \times (I - Q), \tag{1}$$

and the error feedback for QPSK is defined as

$$e_{\text{QPSK}} = I \cdot \text{sgn}(Q) - Q \cdot \text{sgn}(I). \tag{2}$$

For stability, the loop filter as an LPF is added before the error is fed back to the NCO. The NCO implemented with DDS has the phase defined as

$$\begin{aligned} \phi[n] &= \phi[n-1] + \Delta\phi[n] \\ &= \phi[n-1] + (f_0 + k \cdot f_{\text{feedback}}), \end{aligned} \tag{3}$$

where $\phi[n]$ is the phase increment at time n, f_0 is the free running clock (4.096 MHz in our design), and k is the feedback coefficient. The core Verilog implementation of the Error Detector Control is given below, which shows that for QPSK, the feedback value is arithmetically right shifted by 6 bits to match the scale of that in BPSK.

```verilog
if (is_bpsk) begin // BPSK
  out_I_tdata <= in_I_tvalid ? in_I_tdata + in_Q_tdata : 0;
  out_Q_tdata <= in_Q_tvalid ? in_I_tdata - in_Q_tdata : 0;
end
else begin // QPSK
  out_I_tdata <= in_I_tvalid ? (in_Q_tdata[WIDTH-1] ?
    -in_I_tdata : in_I_tdata) >>> 6 : 0;
  out_Q_tdata <= in_Q_tvalid ? (in_I_tdata[WIDTH-1] ?
    -in_Q_tdata : in_Q_tdata) >>> 6 : 0;
end
```

In the Costas loop, the feedback coefficient k is an important coefficient to finetune according to the system. This parameter is reflected as FEEDBACK_SHIFT, and $k \triangleq 2^{-\text{FEEDBACK\_SHIFT}}$. The convergence of the Costas loop under different carrier frequency offset (CFO) is discussed in Section VI-B.

C. Symbol Synchronization Using Gardner Loop

A Gardner loop [9] is used to achieve symbol (timing) synchronization. The structure of a Gardner loop is shown in Fig. 4. The loop itself is not directly shown in the figure,
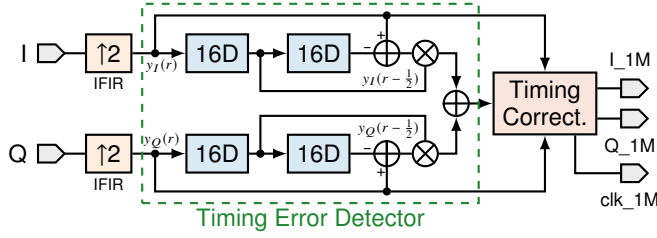
Fig. 4. Structure of a Gardner loop for symbol timing synchronization.

To reduce implementation complexity, we use the sign of strobe values as mentioned in [9]. The total symbol timing error considering I and Q components is

$$u_t(r) = y_I(r - \tfrac{1}{2}) \left[\text{sgn}(y_I(r)) - \text{sgn}(y_I(r-1)) \right]$$
$$+ y_Q(r - \tfrac{1}{2}) \left[\text{sgn}(y_Q(r)) - \text{sgn}(y_Q(r-1)) \right], \quad (4)$$

where r has a symbol frequency of $1.024\,\text{MHz}$. For better timing performance, we linearly interpolate the $16.385\,\text{MHz}$ input I/Q data to $32.768\,\text{MHz}$. Therefore, $y_I(r-1)$ and $y_Q(r-1)$ are delayed by 32 clocks. In FPGA implementation, for each I/Q stream, two shift registers of depth 16 are used. Notably, since we adopt the BPSK constellation in Fig. 2(a), the symbol timing error depends on both I and Q components, the same as QPSK.

V. PACKET-BASED COMMUNICATION

A. Frame Structure

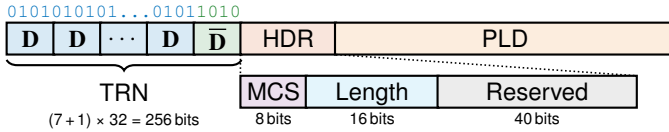The frame structure is shown in Fig. 5, consisting of the TRN, HDR and the PLD fields.

Fig. 5. Frame structure of the packet-based communication.

Training (TRN). The training field is used to provide packet timing information (coarse synchronization), as well as synchronize the carrier and symbol timing. It consists of 7 repetitions of **D** and one **D̄**, where **D** and **D̄** are of length 32. **D** and **D̄** are repetitive sequences of '01' and '10', respectively. The training field from bit 0 to $(7+1) \times 32 - 1 = 255$ is defined as

$$\text{TRN}[i] = \begin{cases} \text{mod}(i, 2), & i = 0, 1, \ldots, 223, \\ \text{mod}(i+1, 2), & i = 224, 225, \ldots, 255. \end{cases} \quad (5)$$

Notably, the phase transition from bit 223 to 224 is used to indicate the boundary of the packet.

Header (HDR). The header field is used to provide packet information, including the modulation and coding scheme (MCS) and the packet length (Length) in bits. The remaining bits are reserved for future use. The MCS field currently only determines the use of BPSK or QPSK. The MCSs for BPSK and QPSK are defined as '01010101' and '10101010' respectively.

Payload (PLD). The payload field is used to carry the actual data. Its length in bits (1 bit for each BPSK symbol, 2 bits

for each QPSK symbol) should match the Length field in the header.

B. Packetizer Design

The packetizer FSM has 5 states: IDLE, HDR, PLD, LAST and WAIT. The IDLE state is used to wait for the packet start. It transits to the HDR state when the valid and ready signals are both high, i.e., the packet starts transmitting. The HDR state is used to transmit the header (including TRN and HDR)[1], as discussed in Section V-A. It then transmits to the PLD state[2] to transmit the payload stored in the FIFO. Notably, the FIFO depth should not be smaller than the header length (i.e. 320). The LAST state is used to transmit the last symbol of the payload, when the AXIS `tlast` signal will be high. After transmitting all payload data, the WAIT status is employed to consume all remaining data in the FIFO. This is only used in our design to better demonstrate the relationship between the transmitted bits and the received bits, and this status can be removed for other applications. Finally, after the FIFO is cleared, the state will tradition to the IDLE state waiting for new data income.

C. SPB Detection

SPD detection consists of strength detection (SD), packet detection (PD) and boundary detection (BD). They jointly provide information for the coarse packet timing.

1) Strength Detection (SD): The strength detection checks the baseband I/Q amplitude from the Costas loop. It is useful because the PSK detector will always output a value, and it can confuse the packet parsing when the noise coincides with the packet sequence. For stability, the `SD_flag` is asserted when either the I or Q signal amplitude is larger than `RX_SD_THRESHOLD` at one clock within the window of `RX_SD_WINDOW` clocks.

2) Packet Detection (PD): The presence of a packet can be identified by detecting repetitive '010101...' sequence (modulated using BPSK). This can be performed by checking the continuous 1s for the differential results. The `PD_flag` is asserted when the differential results remain 1 for `RX_PD_WINDOW` clocks. Interestingly, the packet detection itself does not require a synchronized carrier, as a carrier phase shift does not significantly affect the '01' sequence detection.

3) Boundary Detection (BD): Exploiting the phase transition from bit 223 to 224 in the training field (see Section V-A and Fig. 5), the boundary detection can be performed by checking the differential value being 0 after the `PD_flag` being asserted. To ensure stability, the `BD_flag` is asserted after `RX_BD_WINDOW` clocks of continuous 1s in differential values. At boundary detection, the sign of the bit 223 and 224 are also recorded in `BD_sgn`, which can be used to counter the phase ambiguity in BPSK demodulation.

---

[1]For the packetizer, we do not distinguish TRN and HDR for simplicity, and they are called the header in contrast to the payload.

[2]If the payload symbol length is 1, it will directly transit to the LAST state.

D. Depacketizer Design

The depacketizer works the opposite way of the packetizer. It has 6 states: IDLE, TRN, HDR, PLD, LAST and WAIT. The TRN and the HDR state must be distinguished because the need to extract useful information in the TRN field. The IDLE state transits to the TRN state when `BD_flag` is high. Since the `BD_flag` contains the packet timing information, the demagnetizer can count the number of clocks to determine the start of the HDR field. In the HDR field, both the MCS and packet length are extracted. Then, the depacketizer transits to the PLD state to extract the payload. The `tuser` signal is used to indicate the modulation scheme (1 for BPSK and 0 for QPSK). The `tlast` signal is used to indicate the end of the payload, at the LAST state. To correct disassert the PD and BD signal, the WAIT state is employed to wait for one more clock before it comes to the IDLE state.

VI. Simulation Results

In all Verilog simulation, random is added to the Rx ADC, to better simulate the real-world scenario.

A. Transmitter

The transmitter DAC simulaiton results are shown in Fig. 6.

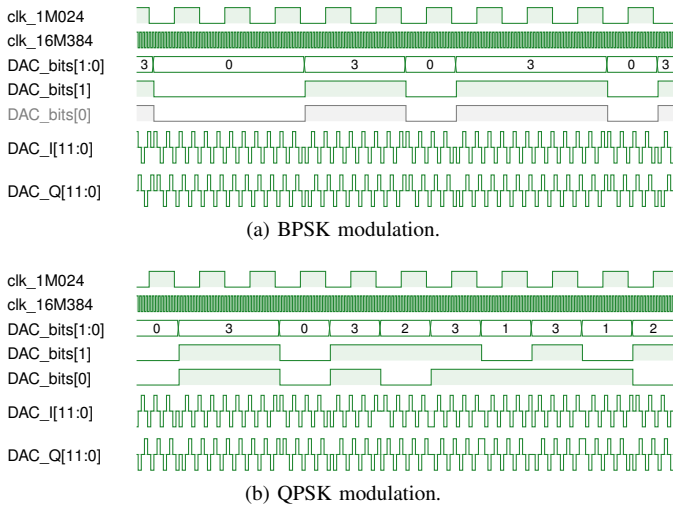(a) BPSK modulation.

(b) QPSK modulation.

Fig. 6. DAC simulation results of the transmitter.

B. Carrier Synchronization Convergence

The simulation results of the Costas loop are shown in Fig. 7. In the BPSK mode, the Costas loop can successfully handle the CFO smaller than 7.81 kHz. It can be verified by the fact that the feedback saturates around a constant (corresponding to the CFO value), which . In the QPSK mode, the value is 1.96 kHz. The convergence is relatively fast.

Notably, the convergence performance of QPSK is poorer than BPSK in the current parameter set, which is by design. For a dual-mode system which transmits payload in both BPSK and QPSK with a BPSK header, so the carrier synchronization is achieved at the BPSK header, therefore, QPSK does not
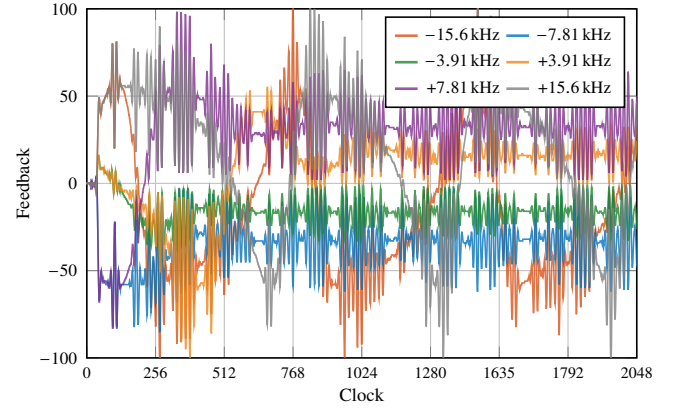

Fig. 7. Carrier synchronization for BPSK. The feedback value after the loop filter v.s. time, with different CFO.

need a strong feedback. Furthermore, the smaller feedback makes the BPSK and QPSK transmission smoother, avoiding a sudden phase jump of $90°$.

C. Mixed-Mode Receiver

In this simulation, we show the successful transmission of packets in both BPSK and QPSK modulations. The simulation results are shown in Fig. 8.

VII. FPGA Implementation

The design is implemented in Vivado 2022.2.

The hardware resources consumption on Zynq-7020 with the default synthesis and implementation strategy is shown in Table II. Therefore, the design itself is area-efficient, and can be readily incorporated into a larger system.

TABLE II
HARDWARE RESOURCES CONSUMPTION ON ZYNQ-7020

Resource	Utilization	Available	Util. Rate
LUT	7,248	53,200	8.04%
LUTRAM	1,121	17,400	6.44%
FF	8,144	106,400	7.65%
BRAM	27	140	19.29%
DSP	51	220	23.18%
IO	33	200	16.50%
BUFG	10	32	31.25%
MMCM	2	4	50.00%

VIII. Experiment Results on SDR

The experiment results are observed via a system ILA in Vivado, and 4 general-purpose input/output (GPIO) pins are used to output some 1-bit signals, including the 1-bit Tx and Rx data stream and their corresponding clock.

In our design, 4 GPIO pins are connected, as listed in Table III. Fig. 9 shows the oscilloscope results of the GPIO outputs.

Fig. 9(a) and Fig. 9(b) connects the `GPIO_TH1` and `GPIO_TH2` pins, showing the transmitted 1-bit sequence (Tx) and the respective received 1-bit sequence (Rx). Clearly, the
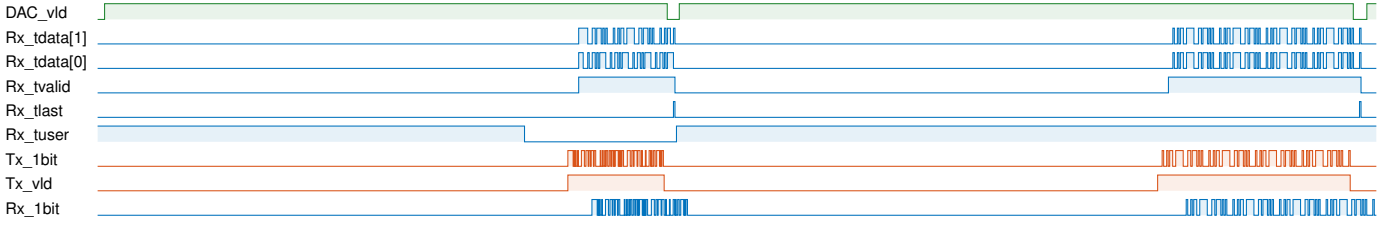
Fig. 8. Mixed-mode simulation results.

TABLE III
GPIO PIN CONNECTIONS

Pin	Signal
GPIO_TH1	Tx 1-bit sequence (clock: 1.024 MHz or 2.048 MHz)
GPIO_TH2	Rx 1-bit sequence (sync. w/ Rx timing clock when BPSK)
GPIO_TH3	2.048 MHz global clock from the clock divider
GPIO_TH4	Rx timing clock (~1.024 MHz)

sequence is successfully recovered in both cases, with a certain time delay. The structure of pn_5 sequence in BPSK is clearly seen in Fig. 9(a), which has a bit frequency of 1.024 MHz. By contrast, the bit frequency of QPSK packets in Fig. 9(b) is 2.048 MHz.

IX. DISCUSSIONS

A. Possible Enhancement

Frame structure design. CRC and/or checksums can be added to the frame structure.

Changing parameters on the fly. AXI peripheral [10] can be used to change the parameters in the Constant_Config on the fly, if the board allows.

B. Possible Extensions Beyond the Experiment

The training (TRN) field can be better utilized for additional experiments. For example, signal-to-noise (SNR) can be estimated at the TRN field.

Channel estimation algorithm [11], [12].

Auto generation [13].

X. CONCLUSION

In this paper,

APPENDIX A
BLOCK DIAGRAMS

A. Block Diagrams Design

B. Debugging With Block Diagrams

AXI connections. 1. connection one out of the bus. 2. FREQ_HZ property.

Testbenches for block diagrams.

APPENDIX B
FIGURES IN THIS PAPER

All figures except for block diagrams in this paper are created using T*i*kZ, part of LATEX. The way I create them is quite interesting, and you can find the source code in the GitHub repository [3].

REFERENCES

[1] R. Zhao, T. Woodford, T. Wei, K. Qian, and X. Zhang, "M-cube: A millimeter-wave massive MIMO software radio," in *Proc. ACM 26th Annu. Int. Conf. Mobile Comput. Network. (MobiCom)*, Sep. 2020, pp. 1–14.

[2] W. Zhao, C. Li, Z. Ji, Z. Guo, X. Chen, Y. You, Y. Huang, X. You, and C. Zhang, "Flexible high-level synthesis library for linear transformations," *IEEE Trans. Circuits Syst. II*, 2023, to be published.

[3] W. Zhao, "sdr-psk-fpga," GitHub repository, 2024. [Online]. Available: https://github.com/Teddy-van-Jerry/sdr-psk-fpga

[4] Wuhan Labtech, "eNodeX 30B SDR innovation & implementation platform," 2024. [Online]. Available: http://www.labtech.cn/Product-software-eNodeX-30B.html

[5] AMD, "Processor system reset module v5.0 product guide (PG164)," Nov. 2015. [Online]. Available: https://docs.xilinx.com/v/u/en-US/pg164-proc-sys-reset

[6] ——, "System integrated logic analyzer v1.1 product guide (PG261)," Feb. 2021. [Online]. Available: https://docs.xilinx.com/v/u/en-US/pg261-system-ila

[7] F. James, "A review of pseudorandom number generators," *Comput. Phys. Commun.*, vol. 60, no. 3, pp. 329–344, Oct. 1990.

[8] M. Simon and W. Lindsey, "Optimum performance of suppressed carrier receivers with Costas loop tracking," *IEEE Trans. Commun.*, vol. 25, no. 2, pp. 215–227, Feb. 1977.

[9] F. Gardner, "A BPSK/QPSK timing-error detector for sampled receivers," *IEEE Trans. Commun.*, vol. 34, no. 5, pp. 423–429, May 1986.

[10] AMD, "AXI external peripheral controller (EPC) v2.0 product guide (PG127)," Oct. 2016. [Online]. Available: https://docs.xilinx.com/v/u/en-US/pg127-axi-epc

[11] W. Zhao, Y. You, L. Zhang, X. You, and C. Zhang, "OMPL-SBL algorithm for intelligent reflecting surface-aided mmWave channel estimation," *IEEE Trans. Veh. Technol.*, vol. 72, no. 11, pp. 15 121–15 126, Nov. 2023.

[12] Y. You, W. Zhao, L. Zhang, X. You, and C. Zhang, "Beam pattern and reflection pattern design for channel estimation in RIS-assisted mmwave MIMO systems," *IEEE Trans. Veh. Technol.*, 2023, to be published, doi: 10.1109/TVT.2023.3309950.

[13] W. Zhao, C. Li, Z. Ji, Y. You, X. You, and C. Zhang, "Automatic timing-driven top-level hardware design for digital signal processing," in *Proc. IEEE 15th Int. Conf. ASIC (ASICON)*, Oct. 2023.
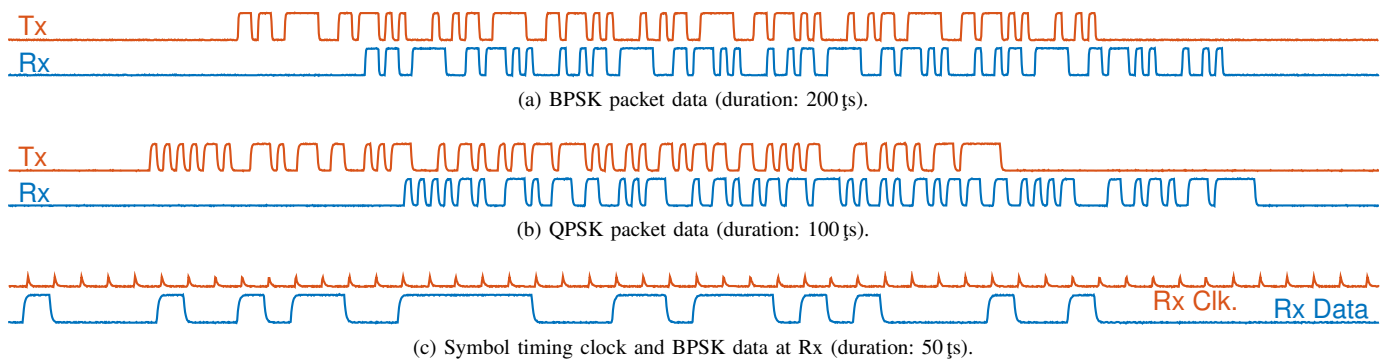
(a) BPSK packet data (duration: 200 μs).

(b) QPSK packet data (duration: 100 μs).

(c) Symbol timing clock and BPSK data at Rx (duration: 50 μs).

Fig. 9. Two-channel oscilloscope results of GPIO outputs.

(a) A BPSK packet.

(b) A QPSK packet.

Fig. 10. System ILA results.

Fig. 11. Top block diagram.