Tape out Presentation

BLE Team

Jerry Duan, Mingying Xie, Yalun Zheng

EE290C Fall 2018

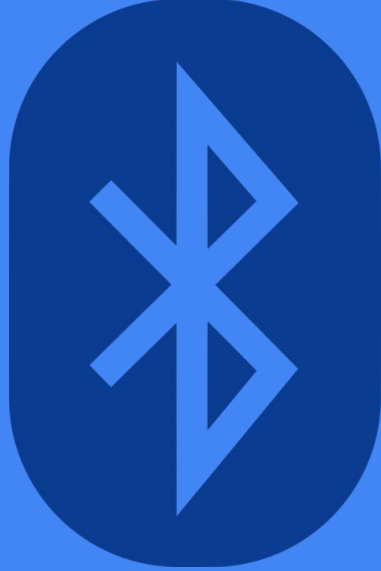


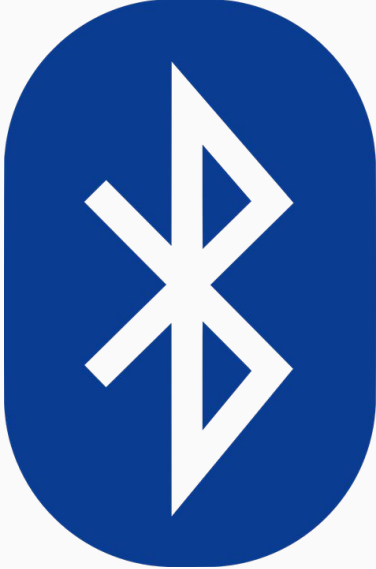
Table of Contents

Background

Block description

Tests and results

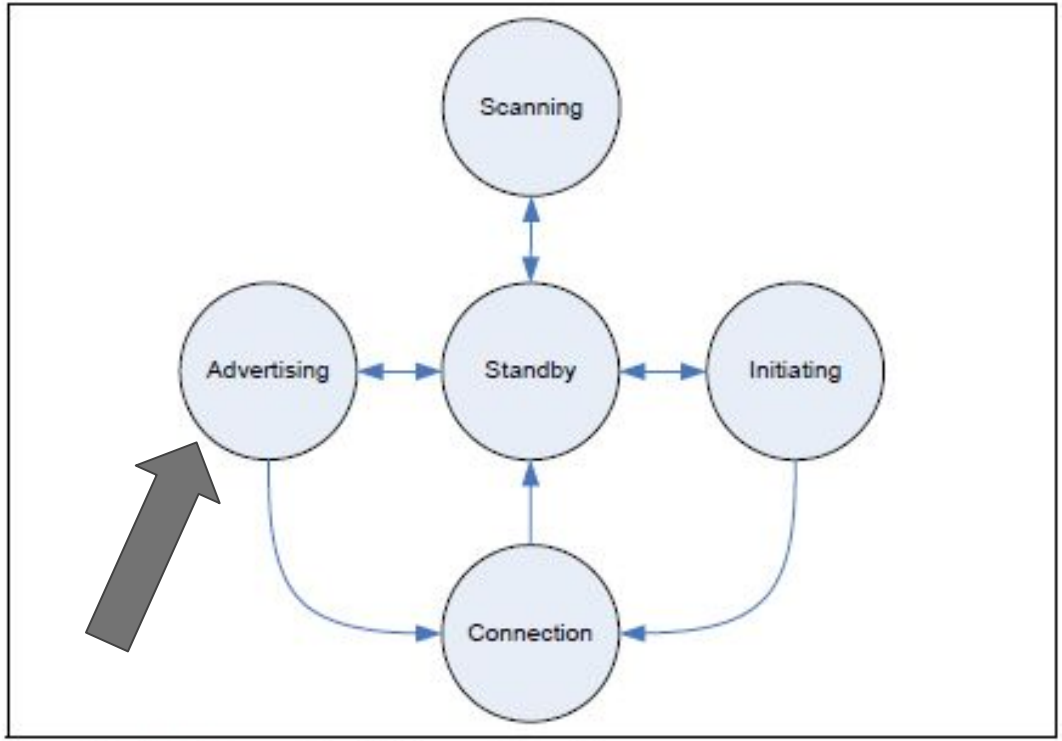
Future work



Motivation

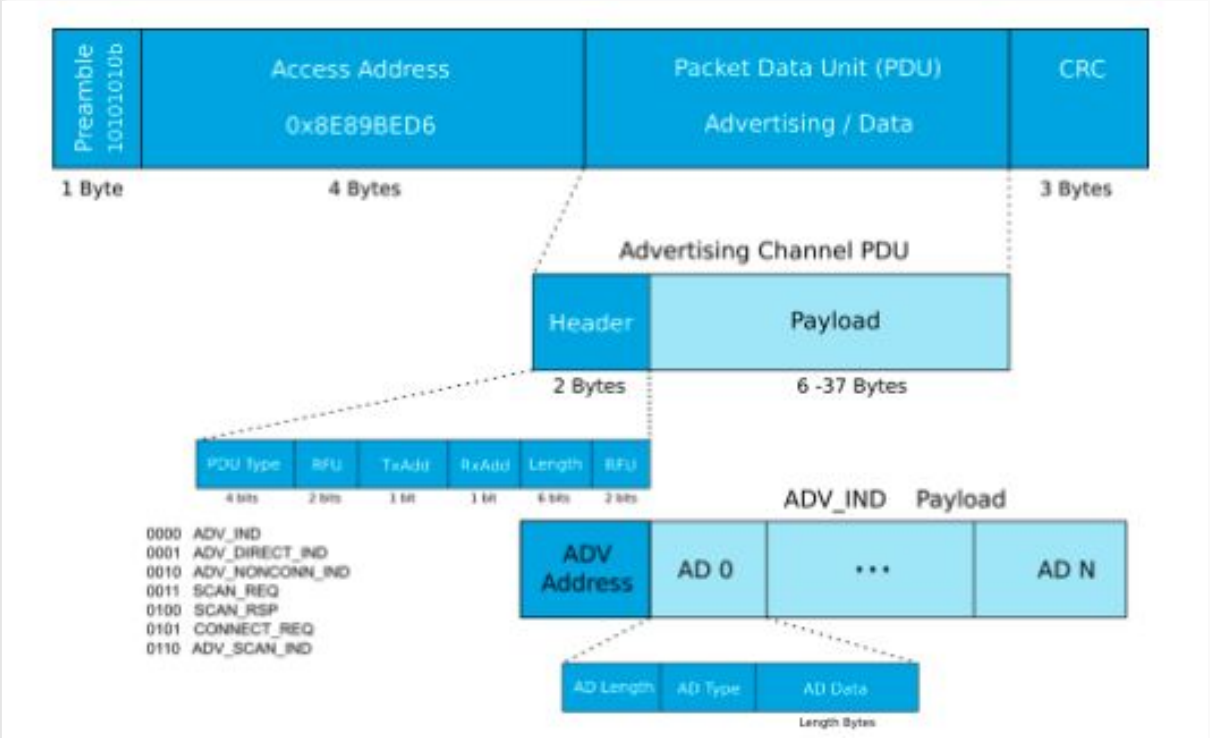
- RISC-V based BLE device
- Crystal-free Standards Compliant Radio
- Low-cost, Low-power

BLE Link Layer



Packet Format

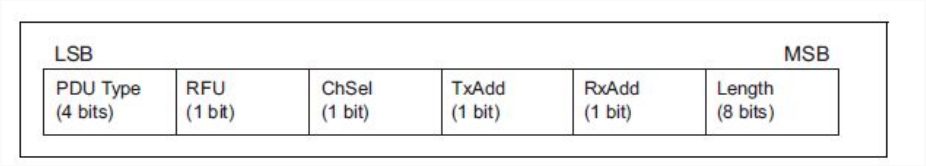
- Preamble
- Access Address
- Protocol Data Unit (PDU)
- Cyclic Redundancy Check (CRC)



http://www.argenox.com/a-ble-advertising-primer/

PDU

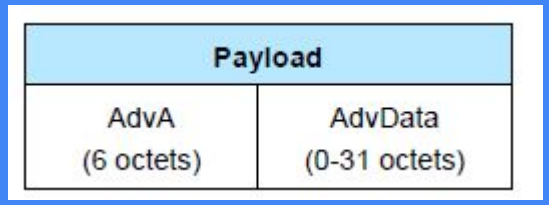
- **PDU Header**
- Payload



PDU Type	PDU Name	Channel	Permitted PHYs		
			LE 1M	LE 2M	LE Coded
0000b	ADV_IND	Primary Advertising	•		
0001b	ADV_DIRECT_IND	Primary Advertising	•		
0010b	ADV_NONCONN_IND	Primary Advertising	•		
0011b	SCAN_REQ	Primary Advertising	•		
	AUX_SCAN_REQ	Secondary Advertising	•	•	•
0100b	SCAN_RSP	Primary Advertising	•		
0101b	CONNECT_IND	Primary Advertising	•		
	AUX_CONNECT_REQ	Secondary Advertising	•	•	•
0110b	ADV_SCAN_IND	Primary Advertising	•		
0111b	ADV_EXT_IND	Primary Advertising	•		•
	AUX_ADV_IND	Secondary Advertising	•	•	•
	AUX_SCAN_RSP	Secondary Advertising	•	•	•
	AUX_SYNC_IND	Secondary Advertising	•	•	•
	AUX_CHAIN_IND	Secondary Advertising	•	•	•
1000b	AUX_CONNECT_RSP	Secondary Advertising	•	•	•
All other values	Reserved for Future Use				

PDU

- PDU Header
- **Payload**



Payload	
AdvA (6 octets)	AdvData (0-31 octets)

Advertising Address:

6-byte, advertiser MAC address

Advertising Data: N sections

- length: 1-byte
- GAP code: 1-byte describing the section

  Section 1: `0x01` - flags

  Section 2: `0x08` - short name
- AdvData: ASCII code

```
section1=[
    0 1 0 0 0 0 0 0 ... % AdvData length 2, LSB first
    1 0 0 0 0 0 0 0 ... % AdvData GAP code 0x01 ("flags"), LSB first
    1 0 1 0 0 0 0 0 ... % AdvData data 5, LSB first.
];
```

```
section2_header=[
    1 1 0 1 0 0 0 0 ... % AdvData length 11, LSB first
    0 0 0 1 0 0 0 0 ... % AdvData GAP code 0x08 ("short name"), LSB first
   % 10 bytes of ASCII data to be appended next
];

value_seq = [ fliplr(dec2bin(int8('U'),8)) fliplr(dec2bin(int8('C'),8)) fliplr(dec2
```

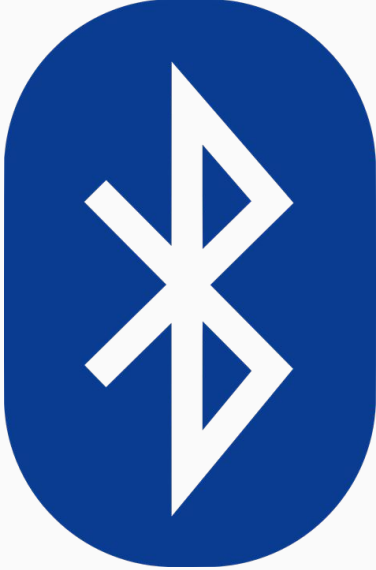
https://www.bluetooth.com/specifications/assigned-numbers/generic-access-profile

Background

Block description

Tests and results

Future work



Packet Assembler

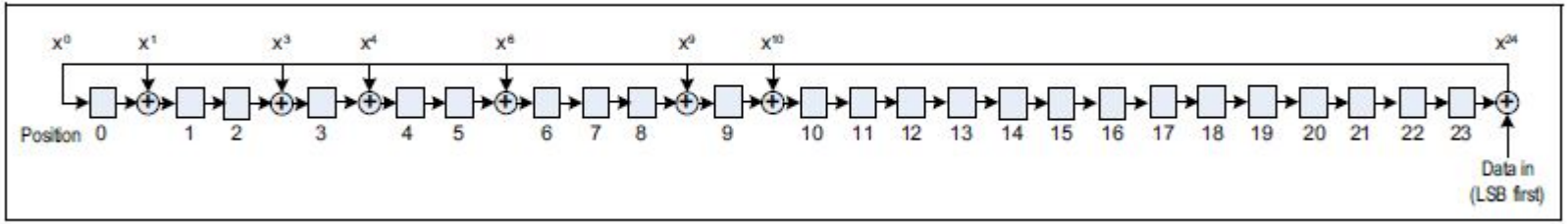
- Collect data packets and assemble them into a complete BLE packet
- Use finite state machine to keep track of packet stage
- Utilize CRC and whitening modules according to BLE spec

Packet Disassembler

- Collect BLE packet bit by bit and send out sections of data
- Use finite state machine similar to packet assembler
- Utilize CRC and de-whitening modules
- Perform error checking
    - Packets with incorrect AA/CRC will be rejected

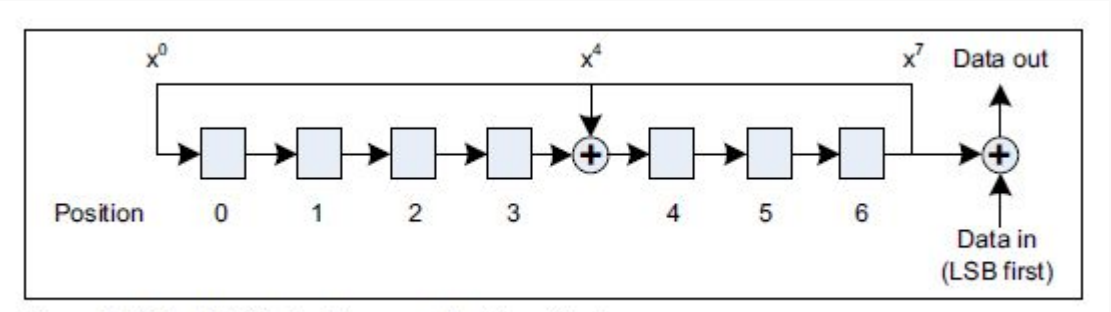
CRC

- 24 bits
- Detect accidental change to data
- Generated through Linear Feedback Shift Register (LFSR)
- Calculated based on the PDU field



Whitening

- Used to avoid long sequences of zeros and ones
- Implemented similar to CRC, using 7-bit LFSR
- Calculated based on PDU and CRC fields

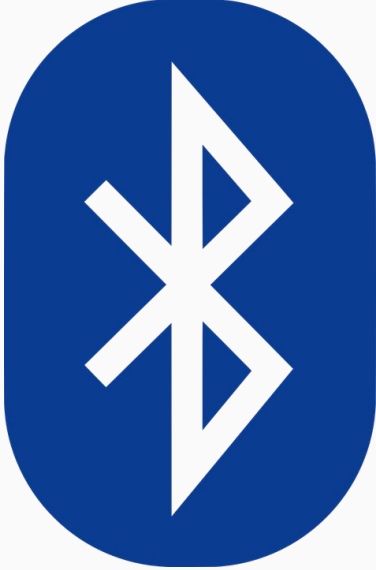


Background

Block description

Tests and results

Future work

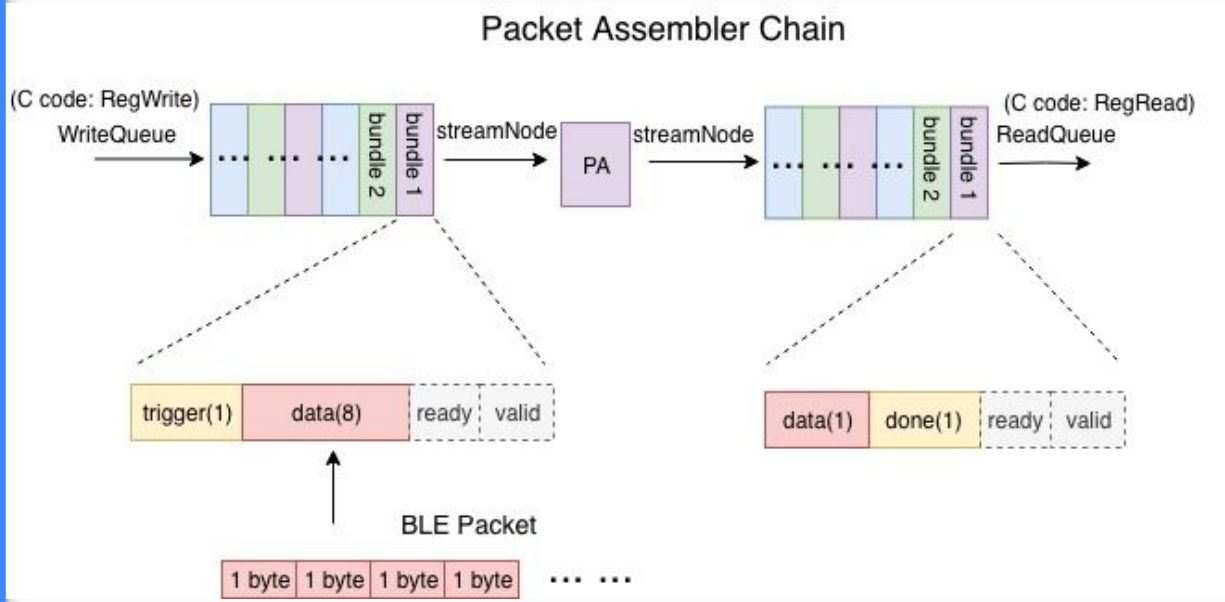


Scala Unit test

- Provided unit tests for packet assembler, CRC and whitening blocks
- Can be tested by typing “sbt test” in the root directory

Packet Assembler Chain

- Connect Packet Assembler to Rocketchip
- BLE Packet written into a FIFO in form of bundles
- Check the result against software golden model



Packet Assembler Chain - Results

```
This emulator compiled with JTAG Remote Bitbang client. To enable, use +jtag_rbb_enable=1.
Listening on port 53225
pack data: 00000001d6
pack data: 00000000d6
pack data: 00000000be
pack data: 0000000089
pack data: 000000008e
pack data: 0000000002
pack data: 000000000f
pack data: 00000000c6
pack data: 0000000080
pack data: 0000000032
pack data: 0000000072
pack data: 0000000002
pack data: 0000000000
pack data: 0000000002
pack data: 0000000001
pack data: 0000000005
pack data: 0000000006
pack data: 0000000008
pack data: 0000000032
pack data: 0000000039
pack data: 0000000030
pack data: 0000000043
0101010101101011011111011001000101110001111100011011101110001001100001001111000010101011001001100000110111101110000011000010100001110010011110011010001010000011110010111010000011000100111011001110101
```

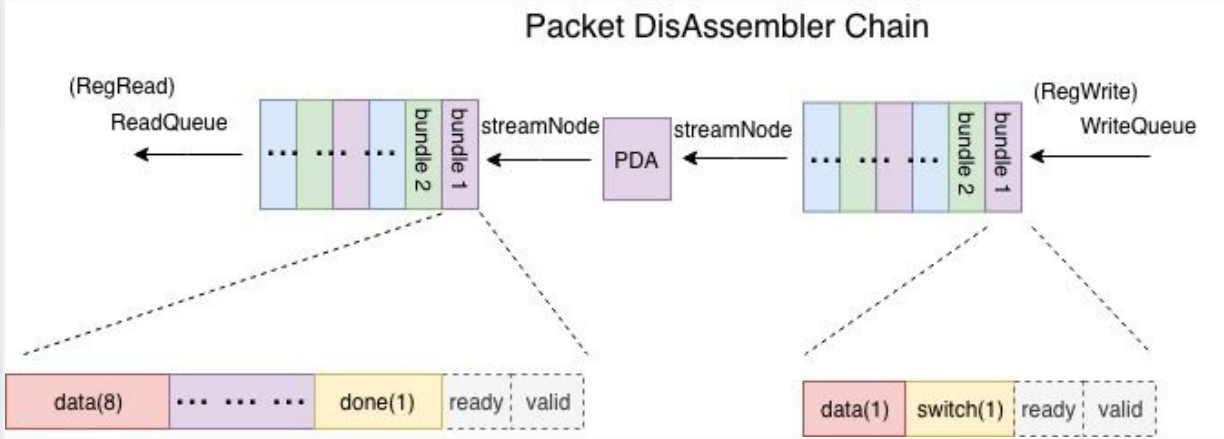
pack data: 00000001d6

Access Address

0x8E89BED6

Packet Disassembler Chain

- Similar to PA Chain
- C code will throw a flag when AA/CRC is wrong

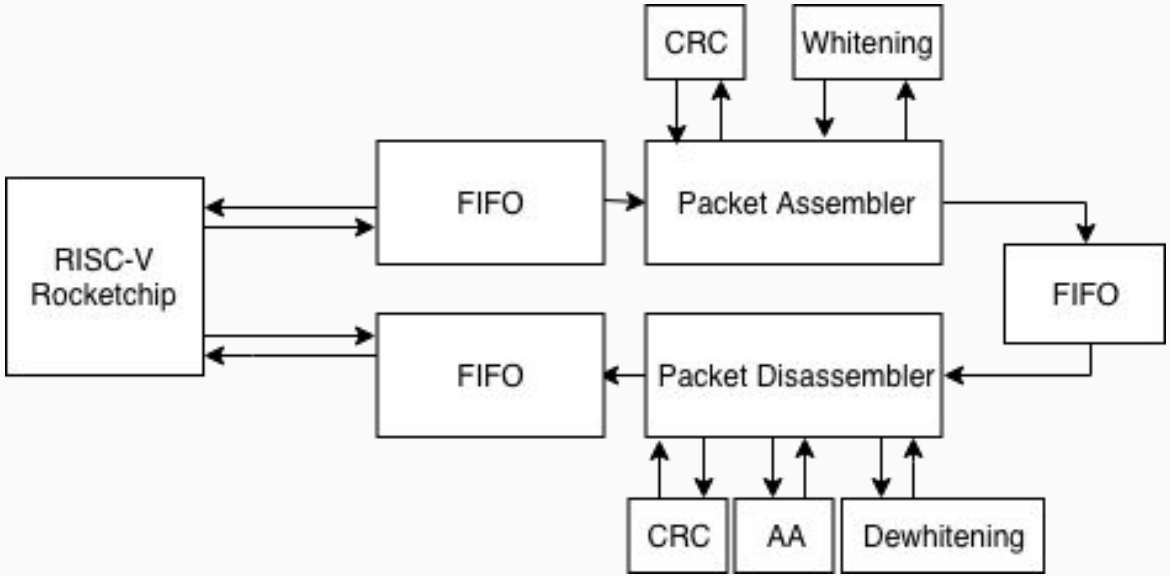


Packet DisAssembler Chain - Results

```
unpack data: d6
unpack data: be
unpack data: 89
unpack data: 8e
unpack data: 02
unpack data: 0f
unpack data: c6
unpack data: 80
unpack data: 32
unpack data: 72
unpack data: 02
unpack data: 00
unpack data: 02
unpack data: 01
unpack data: 05
unpack data: 05
unpack data: 08
unpack data: 32
unpack data: 39
unpack data: 30
unpack data: 43
unpack data: c7
unpack data: fa
unpack data: 65
Finished disassembling
```

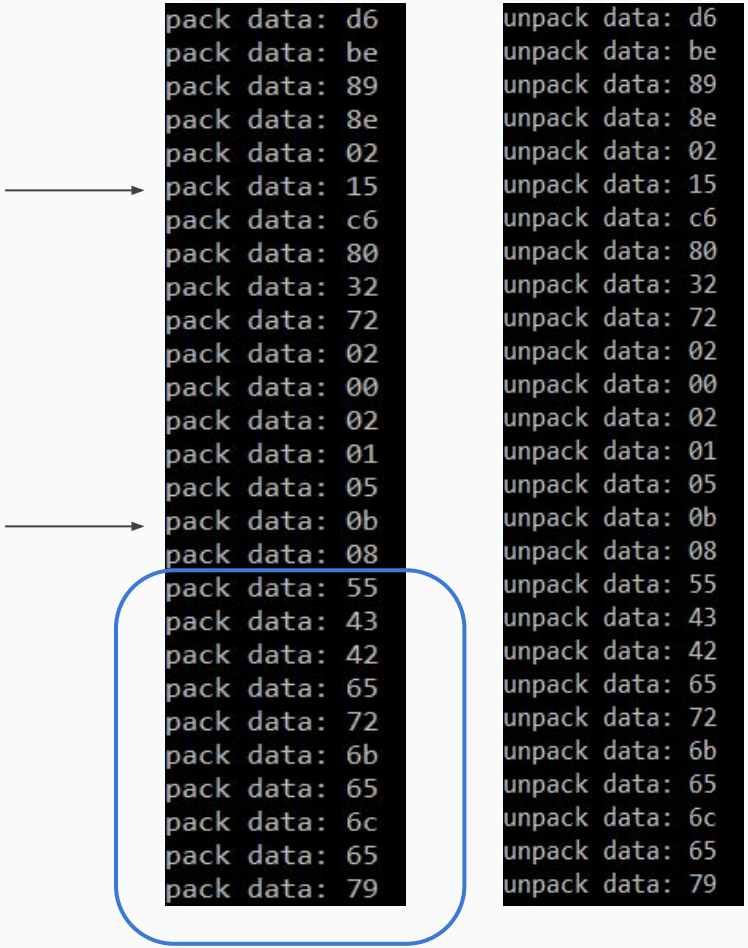
```
unpack data: d6
unpack data: be
unpack data: 89
unpack data: 8e
unpack data: 02
unpack data: 0f
unpack data: c6
unpack data: 80
unpack data: 32
unpack data: 72
unpack data: 02
unpack data: 00
unpack data: 02
unpack data: 01
unpack data: 05
unpack data: 05
unpack data: 08
unpack data: 32
unpack data: 39
unpack data: 30
unpack data: 43
unpack data: 47
CRC Invalid
unpack data: fa
unpack data: 65
Finished disassembling
```

Loop Test


CRC
Whitening
RISC-V
Rocketchip
FIFO
Packet Assembler
FIFO
FIFO
Packet Disassembler
CRC
AA
Dewhitening

Loop Test Result

- The payload is “UCBerkeley”
- Input and output data matches

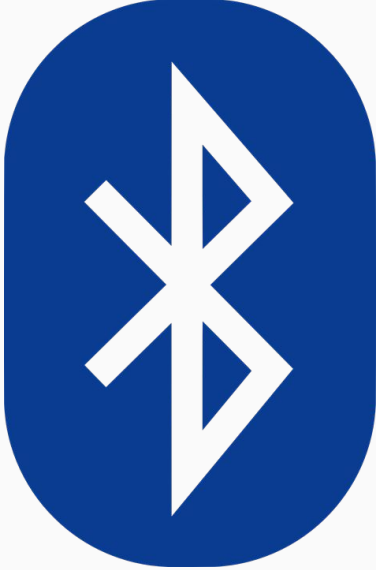


Background

Block description

Tests and results

Future work



Future work

- Besides advertising PDU type, implement scan type (eg.SCAN_REQ), connect type(CONNECT_REQ) and so on
- Interrupts enable Sleep/Busy mode of CPU
- Implement Bluetooth 5 add-on features like FEC (Forward Error Correction)
- Take operation frequency into consideration when the digital BLE baseband has to talk with analog circuits

Acknowledgement

Here is our appreciation to Prof. Borivoje Nikolic, Prof. Kristofer Pister and GSI Paul Rigge for guiding us in this project. Their valuable suggestions and feedback help us move forward. Also the work from last semester's group inspired us greatly and here is their tape-out (https://github.com/tapeout/ble-baseband). Lastly, we would like to thank David Burnett and Rachel Zoll for helping us get on board and explain the BLE packet structure and tests.

Reference

[1] M. Hughes, ”What is Bluetooth 5? Learn about the Bit Paths Behind the New BLE Standard”, *All About Cir- cuit*, July. 2017.

[2] Bluetooth, ”Bluetooth Core Specification V5.0”, vol.6, Dec. 2016.

[3] Z. Gao, ”CRC and Whitening”, *UCB EE290C Spring 2018*, May. 2018.

[4] R. Renn, ”Packet Assembler and Disassembler Final Report”, *UCB EE290C Spring 2018*, May. 2018.

[5] A. Jois, ”DMA-TileLink Interface”, *UCB EE290C Spring 2018*, May. 2018.

[6] C. Fu, ”DMA RF-side”, *UCB EE290C Spring 2018*, May. 2018.

Questions?