EE/CprE/SE 491 WEEKLY REPORT 6

02/13 – 02/20

Group number: SD26-10

Project title: Open-sourced Radio Microcontroller for Fabrication

Client &/Advisor: Henry Duwe duwe@iastate.edu

Client/Company/Organization: ISU; ECPE

Team Members:

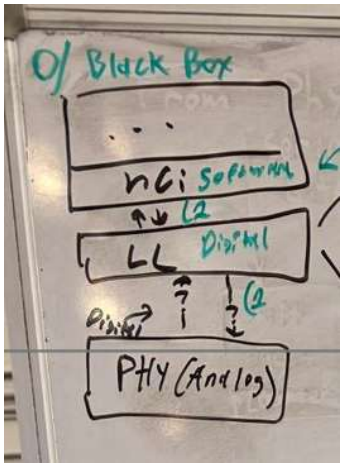
- ***Zane Salti***
- ***Jackson Doyle***
- ***Daniel Pytel***
- ***Tyler Bibus***
- ***Aiden Han-Lindemyer***
- ***Parker Pederson***
- ***Akash Gojuru***
- ***Seth Klaassen***

Weekly Summary:

Link Layer:

Created a top level plan for how to proceed with defining the system and then designing it. The plan is as follows:

1. Create a top level black box diagram:



2. Define I/O:

   This step is defining how each layer will interface with each other and exchange information. This began this week but will be ongoing into next week.

   a. HOST-HCI:

      Defining the interface between the Host and the HCI consists of creating

a list of the functions that the HCI will need to support, datapaths, and other signals.

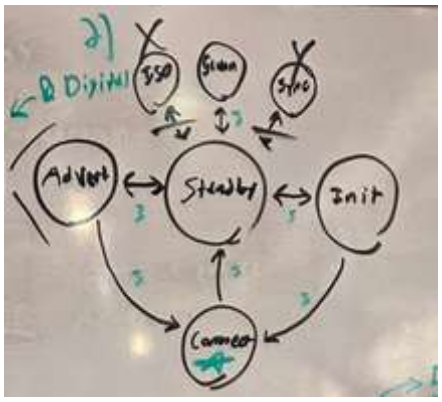
b. HCI-LL:

Defining the interface between the Host and the Link Layer consists of determining what control signals the HCI commands generate as well as determining how the datapath will continue from the HCI into the Link Layer.

c. LL-PHY:

Defining the interface between the Link Layer and the Physical Layer means determining what control signals the physical layer needs in order to function correctly as well as defining how the bitstreams work. All of these connections will be digital, this is to keep all of the analogue work grouped together in the physical layer.

3. Top Level State Machine:



The top level state diagram is already given by the bluetooth standard. The task for our group is to define when these states transition. The inputs that affect these states come from both the HCI, the PHY, and the internal state of the Link Layer.

4. Define Intra-state behaviour

This step is to fill in the details of how each state should behave. The top level state machine defines several modes of operation but lacks the detail required to define how the machine should behave inside of each internal state.

5. Define Hardware Elements

This final step is turning the functional descriptions of each mode into discrete hardware elements that can be described in verilog and then instantiated together to form a final system.

Host Controller Interface:

The main achievement of this week was to define which bluetooth standards need to be supported by our controller to achieve the end of creating a compliant bluetooth LE device. The standard is broken up into the four categories, M are mandatory HCI commands that must be supported by all bluetooth LE devices, O are optional commands that can be optionally supported by any bluetooth LE device, E are emitted or excluded commands that aren't to be supported, and C.X are conditional commands that must be added to support a certain extension of the standard. Supporting an HCI command implies that its underlying features are supported. A list was compiled of all standards that we are planning on adding support for. The features highlighted in yellow are a project extension. The features highlighted in orange are not required for our implementation.

c.1	Mandatory if the LE Controller supports transmitting packets, otherwise
	excluded.
c.2	Mandatory if the LE Controller supports receiving packets, otherwise
	excluded.
c.3	Mandatory if the LE Controller supports Connection State, otherwise
	excluded.
c.4	Mandatory if LE Feature (LE Encryption) is supported, otherwise excluded.
c.6	Mandatory if LE Feature (Connection Parameters Request procedure) is
	supported, otherwise excluded.
c.8	Mandatory if LE Feature (LE Data Packet Length Extension) is supported,
	otherwise optional.
c.15	Mandatory if LE Controller supports transmitting scannable advertisements,
	otherwise excluded.
c.36	Mandatory if the LE Controller supports Central role or supports both
	Peripheral role and LE Feature (Channel Classification), otherwise optional
	if LE Feature (Extended Advertising) is supported and the LE Controller
	supports Advertising State or if LE Feature (Isochronous Broadcaster) is
	supported, otherwise excluded.
c.59	Mandatory if the LE Controller supports Central role, otherwise excluded.
c.60	Mandatory if the LE Controller supports Central role and LE Feature (LE Encryption), otherwise excluded.
c.61	Mandatory if the LE Controller supports Peripheral role and LE Feature (LE Encryption), otherwise excluded.
c.62	Mandatory if the LE Controller supports Central role or supports both Peripheral role and LE Feature (Connection Parameters Request Procedure), otherwise excluded.
c.94	Mandatory if the LE Create Connection or LE Extended Create Connection command is supported, otherwise excluded.
c.96	Optional if the LE Controller supports Connection State, otherwise excluded.
c.97	Mandatory if Advertising State is supported, otherwise excluded.
c.98	Mandatory if Scanning State is supported, otherwise excluded.
c.99	Mandatory if LE Generate DHKey command [v2] is supported, otherwise optional.
c.107	Mandatory if the Set Controller To Host Flow Control command is supported, otherwise excluded.
c.109	Mandatory if the Set MWS Signaling command is supported, otherwise excluded.
c.145	Mandatory if any event in event mask page 2 is supported, otherwise optional.
c.155	Mandatory if the Write Authenticated Payload Timeout command is supported, otherwise excluded.
c.156	Mandatory if the Read Local Supported Codecs command [v2] is supported, otherwise excluded.

Each of these features has an associated list of commands that must be supported by the HCI which we have compiled into a spreadsheet with hyperlinks and notes. Below is an excerpt from the spreadsheet.

Version	Name	Standard	Opcodes' OGF	Opcodes' OCF
Mandatory for LE				
1.1	Command Complete event	M		0x0E
1.1	Command Status event	M		0x0F
1.1	Read BD_ADDR command	M		0x0009
1.1	Read Local Supported Features command	M		0x0003

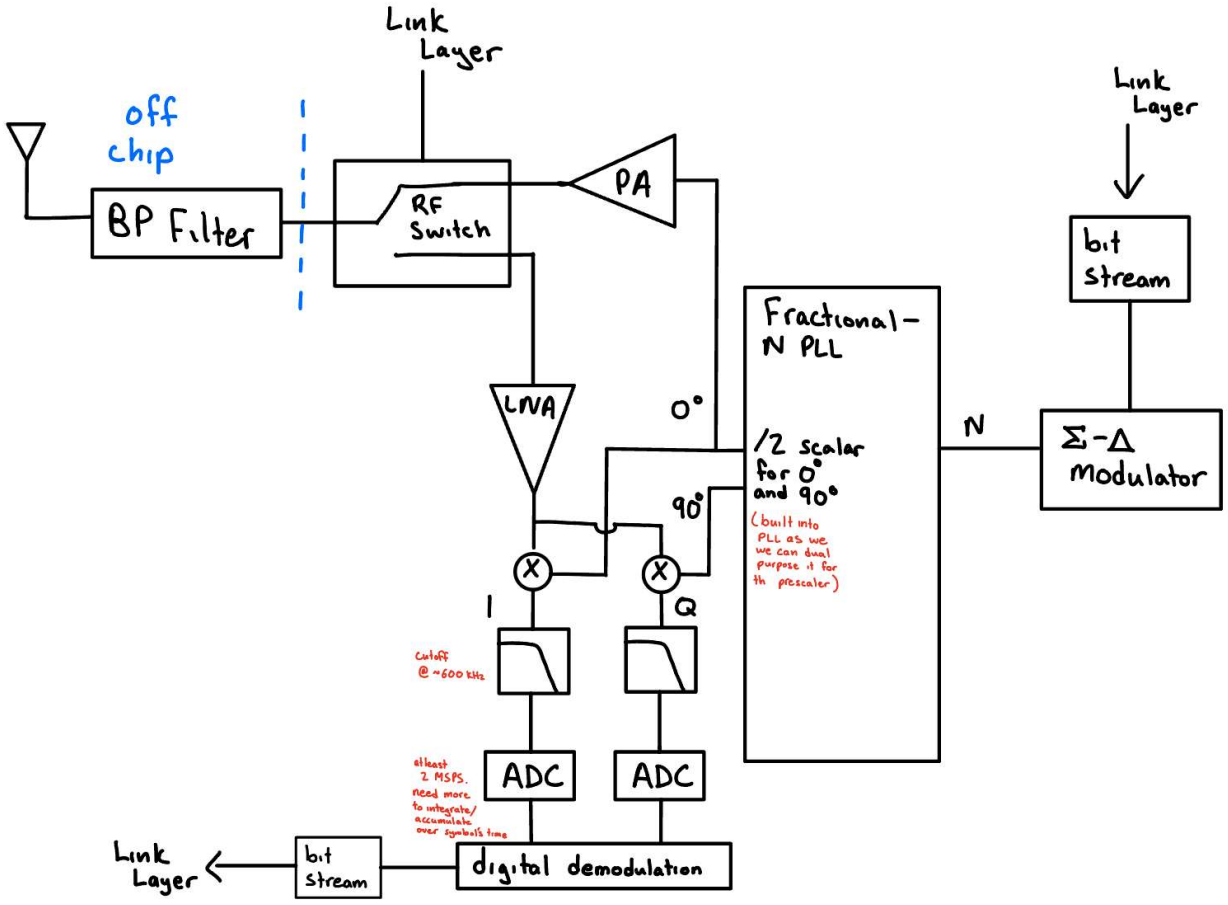
Phy layer:

Updated transceiver architecture and looked at possible RF switch architectures.

Past week accomplishments:

- Zane Salti:

Talked with Dr. Tayeb for feedback on the transceiver architecture. He recommended IQ downconversion to be used on the receive end as opposed to just using a single mixer. A 90 degree phase shifted signal can come from using a frequency divider (via flip flops) (this goes into the different ways a 90 degree phase shift can be accomplished). This would require the PLL to be at twice the frequency. The frequency divider can be implemented inside the PLL as a pre-scaler divider. Low-pass filters after the IQ mixers can be used to attenuate out of channel communications. A cutoff of ~600 kHz is sort of arbitrarily chosen, for now, to give space for the in-channel frequency shifts. . Considering that we're going for the 1M modulation scheme, our ADC needs to sample well over 2 MSamples/second. The ADCs can be low resolution (one or two bits) considering that the point is to track if the frequency is positive or negative (in reference to the LO being set to the channel center). Post ADCs, digital demodulation can be used to accumulate/integrate over a symbol's lifetime to determine the symbol. The switching from a positive to a negative frequency, or vice-versa, can be used to synchronize the beginning of a symbol. The preamble of alternating one's and zero's can be used to recognize the start of a packet.



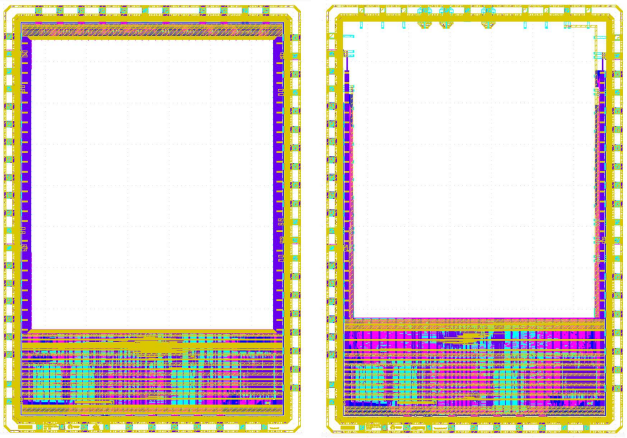
Discussed with Ibram, a former senior design student who did major design and work on an IC PLL, about the status of the PLL, architecture and design choices of it, and issues and possible solutions.

For testing, found that BabbleSim and Bumble could be useful.

Looked at Zephyr and found that they also have a software implementation of the Link Layer. They have a hardware abstraction layer (HAL) for interacting with the PHY radio.

Explored possible RF switch architectures on https://www.microwaves101.com/encyclopedias/switch-design. The series/shunt SPDT switch seemed like a good balance between simplicity and performance. Some of the other switch architectures in the article used transmission lines which aren't really applicable to IC design (well at least not applicable at our frequency).

Looked at the Caravel and Caravan IC templates. The default is the Caravel template, though the Caravan template offers high frequency bare pads, which is desired:



Caravel Caravan

Contributed to the HCI supported commands spreadsheet, continued reading bluetooth documentation, and watched “xschem” and “magic” tooling videos online.

- Jackson Doyle: Studied RF and continued towards proficiency in the toolflow. Developed a better understanding of the principles behind the PLL design from the last group.
- Daniel Pytel: Updated the SD website. Also continued Toolflow tutorials and did some miscellaneous RF research.
- Tyler Bibus: Developed in tandem with LL members on a general top level plan for LL development. Defined top level black boxes and state machines. Defined tools to be used for Digital logic development, specifying Verilog as the HDL language of choice. Started working on I/O design list for LL and HCI.
- Parker Pederson: Used our requirements in order to select specific Bluetooth standards. Created a spreadsheet that compiled a list of all required opcodes for the HCI.
- Akash Gojuru: Completed the ChipForge tutorials and reviewed the LL section.
- Seth Klaassen: Read through the link layer documentation and started designing the rf switch, currently seeing if a wideband or diode pin switch would be more sensible
- Aiden Han-Lindemyer: Created BT Advertizing PHY channel PDU and signals list spreadsheet.

Pending issues:

- Zane Salti: Link Layer architecture hasn’t yet been settled on.
- Jackson Doyle: Start development of the RF switch in software if needed. Otherwise continue to polish ability to use the toolflow
- Daniel Pytel: Scoping the hardware design and general toolflow struggles.
- Tyler Bibus: Still wrapping head around HCI and LL interfaces given limited documentation.
- Aiden Han-Lindemyer: More reading of LL architecture to write documents and gain understanding.
- Parker Pederson: Poor understanding of toolflow use for hdl in project. Need to understand the overall architecture of the link layer
- Seth Klaassen: still needs to finish toolflow tutorial and grasp link layer in a more in depth manner
- Akash Gojuru: Still learning and understanding the high-level design of the LL and how the overall system works.

Individual contributions

NAME	Hours this week	HOURS cumulative
Zane Salti	7	12
Jackson Doyle	3	7
Daniel Pytel	3	5
Tyler Bibus	4	7
Aiden Han-Lindemyer	4	8
Parker Pederson	4	8
Akash Gorjuru	3	7
Seth Klaassen	3	6

Comments and extended discussion

In the upcoming week we must decide how to break down areas of responsibility for this project. At the moment the divide between link layer and PHY has become well defined but subdivisions must be established for both efficiency and accountability.

Plans for the upcoming week

- Zane Salti: Work on designing a top level architecture of the Link Layer and map how data can flow from HCI to PHY. Analyze available bare GFSK transceivers that could potentially be used for physically prototyping the Link Layer.
- Jackson Doyle:Solidify understandings of both RF components and the down conversion plan. Construct a high level plan to move forward in terms of priority for phy layer team.
- Daniel Pytel: Finish up tutorials and break down the previous team's progress on the PLL, hopefully figure out what can be reconfigured (Zigbee -> BLE).
- Tyler Bibus: Create an initial signal list for LL to HCI and PHY interfaces. Decompose state machine into manageable portions to create digital schematics and initial HDL.
- Aiden Han-Lindemyer: Write a document for LL architecture to help my understanding and everyone in LL on board for architecture and overall understanding.
- Parker Pederson: Create a sample project that can be simulated on a caravel board. Define the connection state behaviour for the link layer. Work on LL top level architecture and potentially state machine portions.
- Akash Gojuru: Understand the LL high level design, support breaking down the state machine, and help with writing the first HDL code.
- Seth Klaassen: finish toolflow tutorial and begin rf switch design and high frequency testing

Summary of weekly advisor meeting

The meeting focused on refining the RF system architecture and clarifying project scope. Key technical discussion addressed the TX/RX signal path configuration, including proper gate inversion and antenna routing. Frequency instability concerns were reviewed, with feedback indicating that excessive PLL gain may be causing oscillation. Reducing PLL gain and carefully managing VCO behavior were identified as necessary to meet tight frequency precision requirements.

Discussed the Caravel and Caravan IC templates. While the Caravel template is the default recommended template, Caravan offers high speed bare pads. Though a change to using Caravan would need to be justified as it might no longer share a platform with the other ChipForge projects.

Faculty expertise alignment was discussed, noting that Dr. Geiger and Dr. Chen focus on signal generation and RF design, while Dr. Wong specializes in system-level implementation. Duwe emphasized questions need to be appropriately targeted as they will improve feedback quality; provide as much documentation to the professors. Duwe advised the team to seek feedback from Dr. Neihart specifically in order to learn more about RF design on ICs.

The team was advised to keep the project scope specific when seeking feedback and to prepare structured documentation outlining design requirements, insertion loss analysis, testing methodology, and simulation plans (e.g., SPICE). Clear documentation will enable more actionable faculty feedback.

Finally, the group discussed framing the project within specific IoT use cases (e.g., wireless mice) to ensure the final panel has something tangible that they can recognize as a bluetooth device, rather than proving a broad “IoT” definition which is too broad.