Introduction to Software-Defined Radio

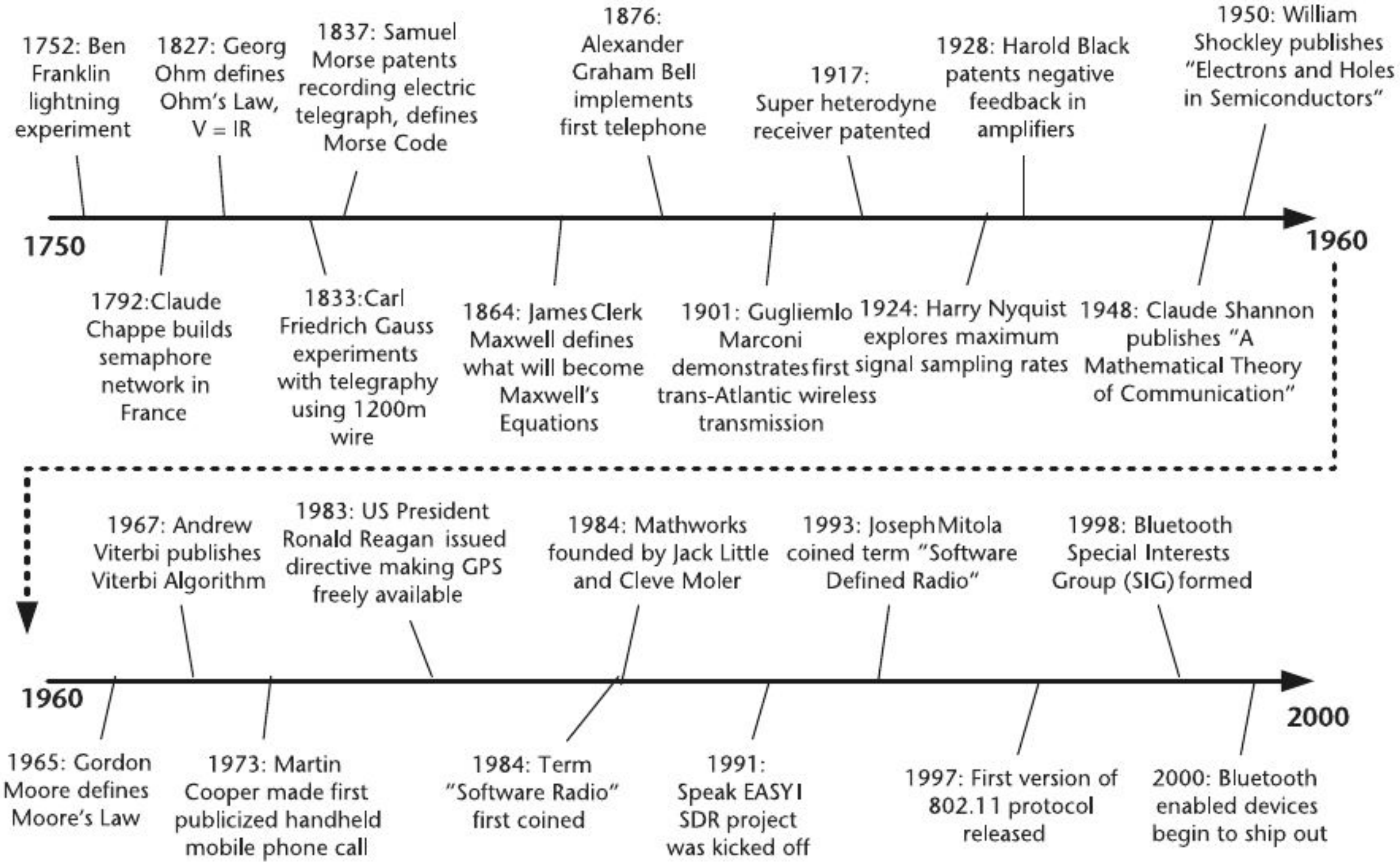
Chapter 1

Introduction

With modern advances in computing technologies, digital signal processing and digital communication algorithms, artificial intelligence, radio frequency (RF) hardware design, networking topologies, and many other elements have evolved modern communication systems into complex, intelligent, high-performance platforms that can adapt to operational environments and deliver large amounts of information in real-time, error-free. The latest step in communication systems technology is the software-defined radio, or SDR, which adopts the most recent advances in all fields to yield the ultimate transmitter and receiver.

1.1 Brief History

This history is dominated by various people investigating ideas or concepts, publishing the results, then allowing their peers and colleagues to build on their work. Many turned their work into commercial products and became famous and rich; some became neither.


1750
1960
1752: Ben Franklin lightning experiment
1792:Claude Chappe builds semaphore network in France
1827: Georg Ohm defines Ohm's Law, V = IR
1833:Carl Friedrich Gauss experiments with telegraphy using 1200m wire
1837: Samuel Morse patents recording electric telegraph, defines Morse Code
1864: James Clerk Maxwell defines what will become Maxwell's Equations
1876: Alexander Graham Bell implements first telephone
1901: Gugliemlo Marconi demonstrates first trans-Atlantic wireless transmission
1917: Super heterodyne receiver patented
1924: Harry Nyquist explores maximum signal sampling rates
1928: Harold Black patents negative feedback in amplifiers
1948: Claude Shannon publishes "A Mathematical Theory of Communication"
1950: William Shockley publishes "Electrons and Holes in Semiconductors"
1960
2000
1965: Gordon Moore defines Moore's Law
1967: Andrew Viterbi publishes Viterbi Algorithm
1973: Martin Cooper made first publicized handheld mobile phone call
1983: US President Ronald Reagan issued directive making GPS freely available
1984: Term "Software Radio" first coined
1984: Mathworks founded by Jack Little and Cleve Moler
1991: Speak EASY I SDR project was kicked off
1993: Joseph Mitola coined term "Software Defined Radio"
1997: First version of 802.11 protocol released
1998: Bluetooth Special Interests Group (SIG) formed
2000: Bluetooth enabled devices begin to ship out

1.2 What is a Software-Defined Radio?

Software Controlled: Software controlled refers to the use of software processing within the radio system or device to select the parameters of operation.

Software Defined: Software defined refers to the use of software processing within the radio system or device to implement operating (but not control) functions.

Software Controlled Radio: Radio in which some or all of the physical layer functions are software controlled.

Software-Defined Radio (SDR): Radio in which some or all of the physical layer functions are software defined.

1.2 What is a Software-Defined Radio?

In today's modern systems signal processing has progressed to such an extent that a majority of baseband functionality is being implemented in software. The flexibility of the RF hardware to be re purposed and reconfigured has led to one radio front-end handling most RF systems. Normally the RF front-end is software controlled rather than software defined. This modern combination of flexible RF front-ends and signal processing in software has lead the birth of software-defined radio.

1.2 What is a Software-Defined Radio?

An SDR system is a complex device that performs several complicated tasks simultaneously in order to enable the seamless transmission and reception of data.

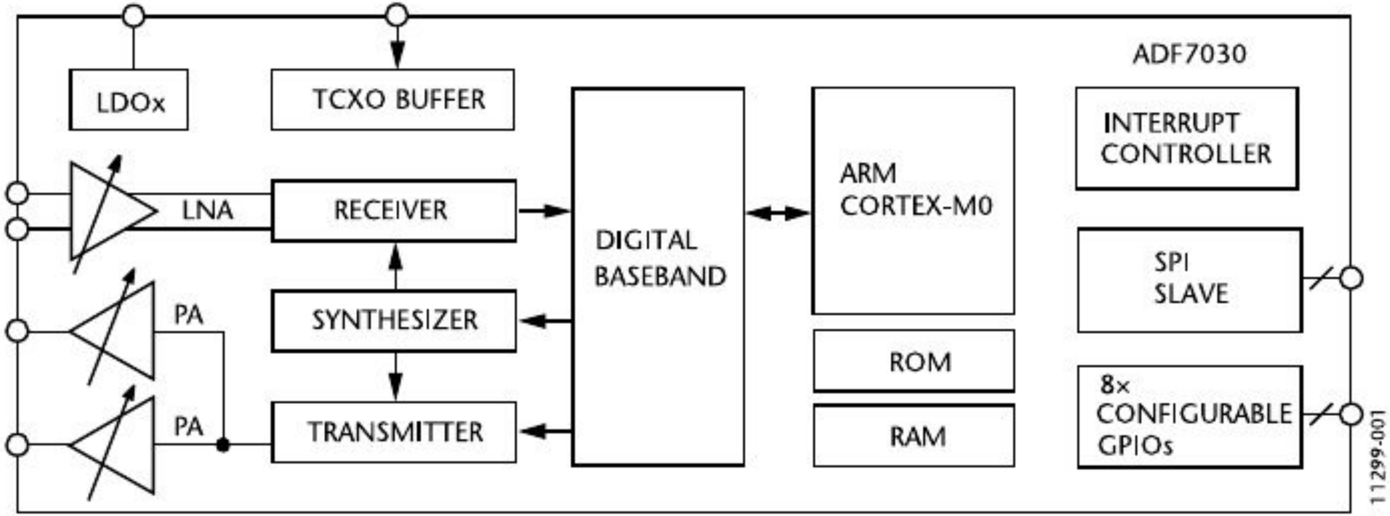
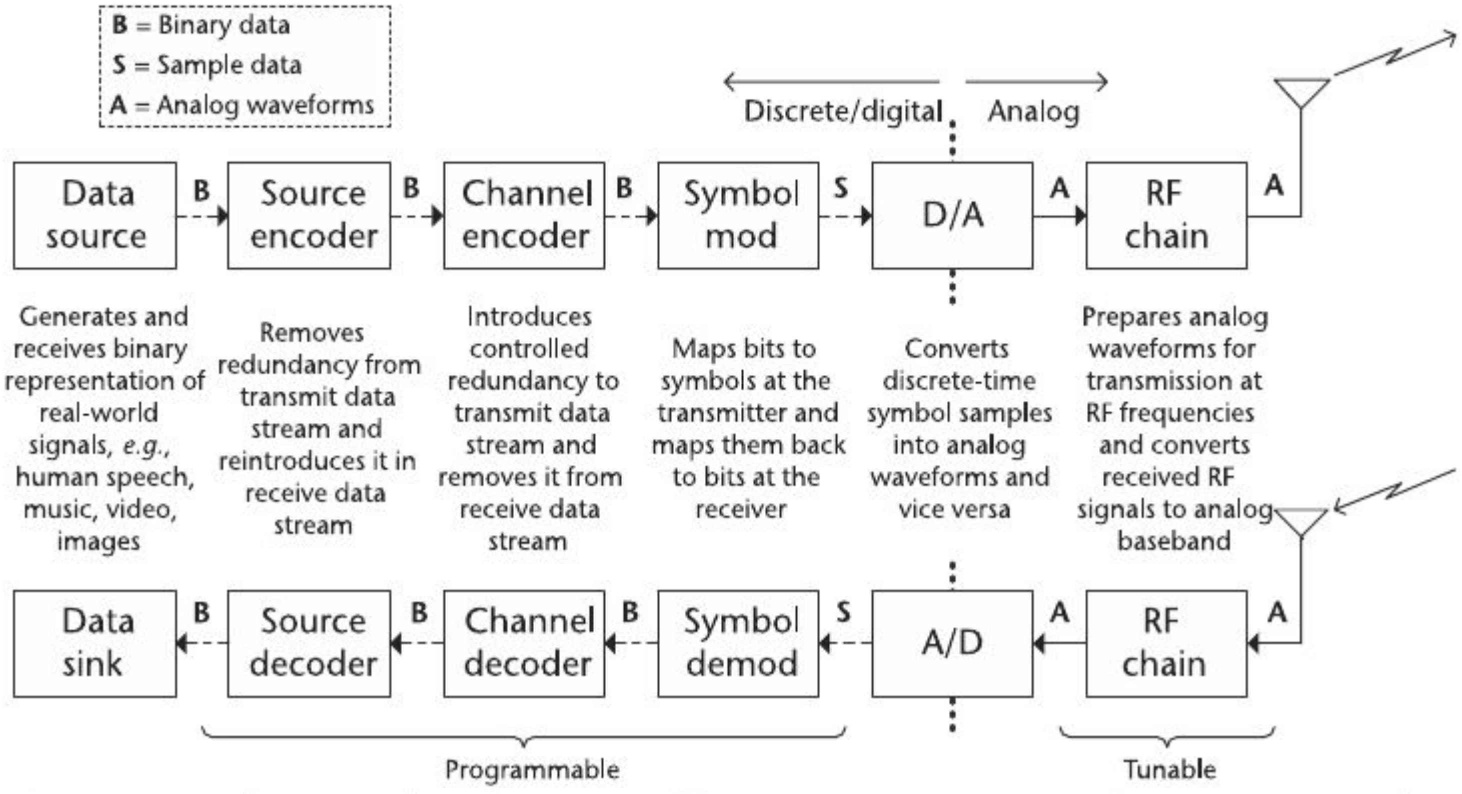


Figure 1.2 ADF7030 block diagram [3].

1.2 What is a Software-Defined Radio?



1.3 Networking and SDR

There are two models for dividing up a communication system into layers: the Open System Interconnection (OSI) 7-layer model and the Transmission Control Protocol (TCP)/Internet Protocol (IP) 5-layer model.

1.3 Networking and SDR

	Units	Protocol	OSI 7-Layer Model	TCP/IP 5-Layer Model	
Level 7	Data	Resource sharing, file access, SMTP, HTTP	**Application** Network process to application	**Application** (FTP, SMTP, HTTP, etc)	Host layers
Level 6	Data	Compression, JPEG, ASCII, TIFF, GIF	**Presentation** Data representation and encryption		
Level 5	Data	Security, name recognition, RPC, SQL, NFS	**Session** Interhost communication		
Level 4	Segments	Segmentation and traffic, TCP, SPX, UDP	**Transport** End to End connections and reliability	**TCP**	
Level 3	Packets	Routing, fragmentation, IP, IPX, IMCP	**Network** Path determination and logical addressing (IP)	**Network**	Media layers
Level 2	Frames	Sequencing, traffic control, Media Access	**Data Link** Physical addressing (MAC & LLC)	**Data Link**	
Level 1	Bits	Bits and volts data encoding	**Physical** Media, signal and binary transmission	**Physical**	

Figure 1.4 Seven-layer OSI model compared to five-layer TCP/IP model.

1.3 Networking and SDR

From the perspective of a radio system and its implementation, much of the system design will focus on the physical layer (as does this text), but it can’t be forgotten how the link layer may affect the physical layer. Nevertheless, given that the baseband processing is all conducted in software, it is possible for the communications system to implement the higher layers of the stack in software as well.

Many communication standards have adopted this scheme, where the entire communication system across all the layers are implemented in software, although depending on data rate requirements, this can require significant computational capabilities on the part of the system to achieve real-time operation.

1.3 Networking and SDR

All software implementations enable functional updates without hardware replacement. In practice, this is normally only done on emerging standards or where data rates are relatively low.

The trade-offs of which function or layer is done in fixed hardware versus flexible software is an engineering decision based on volume, cost, power, complexity, and many other factors.

1.3 Networking and SDR

The link layer will also affect the physical (PHY) layer of a wireless communication system as shown in Figure 1.5.

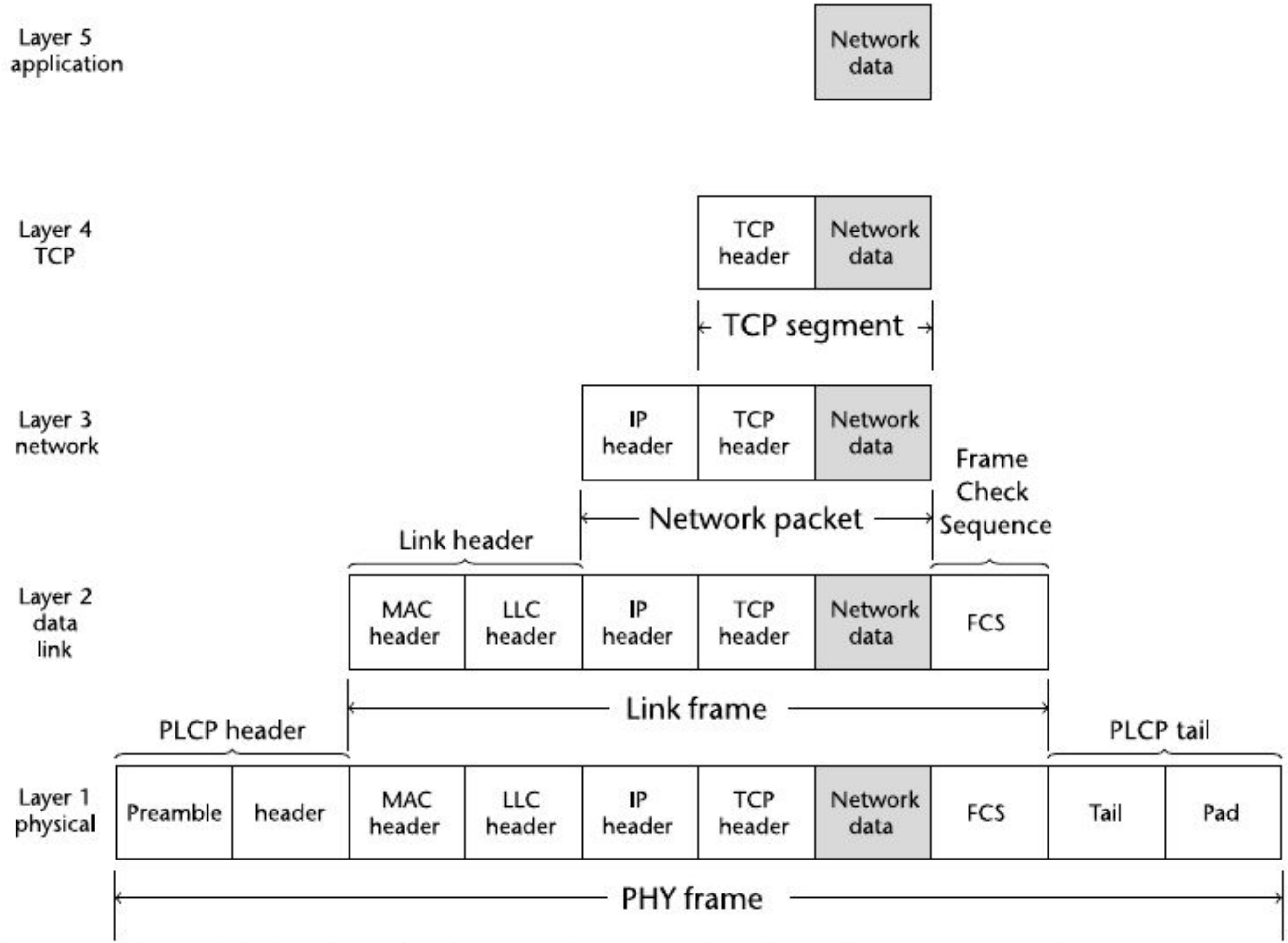
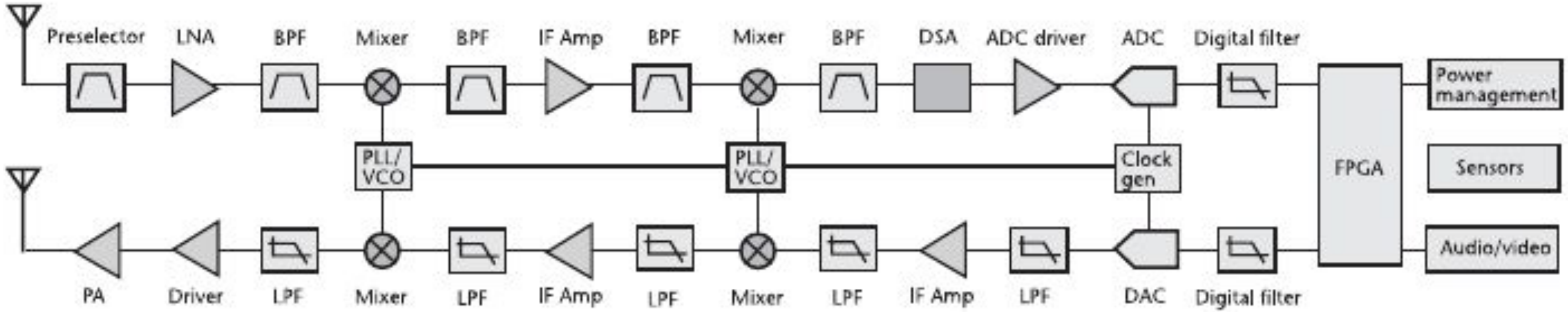


Figure 1.5 Packet structure effects SDR, PLCP = Physical Layer Convergence Protocol.

1.4 RF architectures for SDR

The superheterodyne architecture has been the backbone of radio design.



1.4 RF architectures for SDR

The single or dual mixing stage superheterodyne architecture is the common choice (see Figure 1.6). The benefits of this design are clear: proper frequency planning can allow for very low spurious emissions, the channel bandwidth and selectivity can be set by the intermediate frequency (IF) filters, and the gain distribution across the stages allows for a trade-off between optimizing the noise figure and linearity.

1.4 RF architectures for SDR

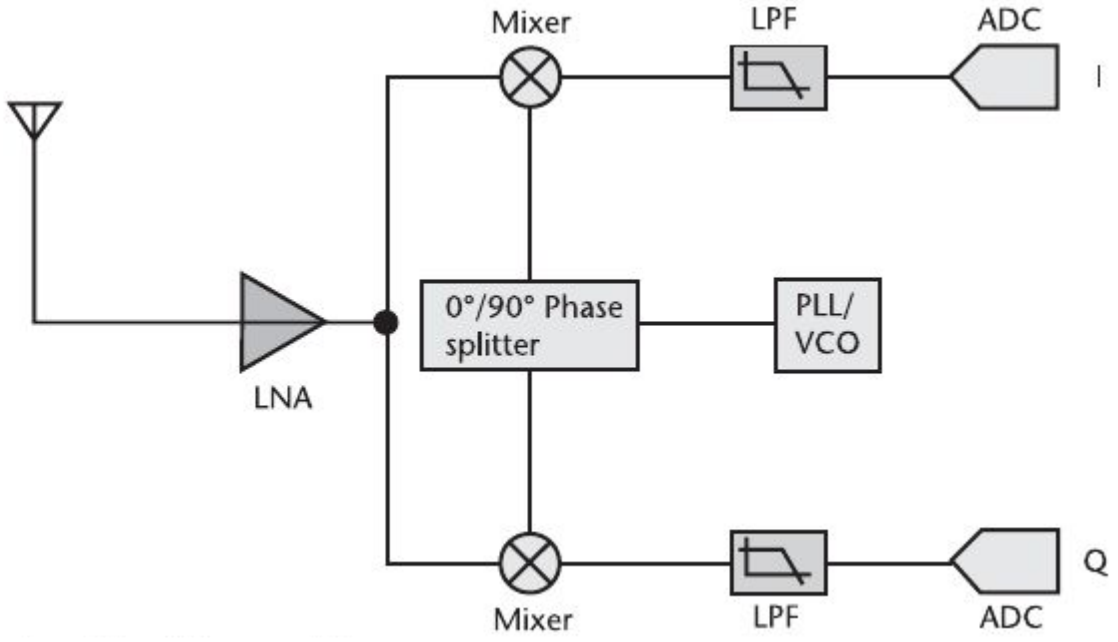
Device-specific improvements are beginning to reach the point of diminishing returns. While the RF components have followed a reduced size, weight, and power (SWaP) trend, high-performance filters remain physically large and are often custom designs, adding to overall system cost.

Additionally, the IF filters set the analog channel bandwidth of the platform, making it difficult to create a common platform design that can be reused across a wide range of systems.

For package technology, most manufacturing lines will not go below a 0.65-mm or 0.8-mm ball pitch, meaning there is a limit on how physically small a complex device with many I/O requirements can become.

1.4 RF architectures for SDR

An alternative to the superheterodyne architecture, which has reemerged as a potential solution in recent years, is the zero-IF (ZIF) architecture.



1.4 RF architectures for SDR

A ZIF receiver (see Figure 1.7) utilizes a single frequency mixing stage with the local oscillator (LO) set directly to the frequency band of interest, translating the received signal down to baseband in phase (I) and quadrature (Q) signals.

This architecture alleviates the stringent filtering requirements of the superheterodyne since all analog filtering takes place at baseband, where filters are much easier to design and less expensive than custom RF/IF filters.

The ADC and DAC are now operating on I/Q data at baseband, so the sample rate relative to the converted bandwidth can be reduced, saving significant power.

For many design aspects, ZIF transceivers provide significant SWaP reduction as a result of reduced analog front-end complexity and component count.

1.4 RF architectures for SDR

Due to real-world factors it is impossible to maintain a perfect 90◦ phase offset between the I and Q signals.

Historically, the I/Q imbalance has limited the range of applications that were appropriate for the ZIF architecture. This was due to two reasons: first, a discrete implementation of the ZIF architecture will suffer from mismatches both in the monolithic devices and also in the printed circuit board (PCB).

In addition to this, the monolithic devices could pull from different fabrication lots, making exact matching very difficult due to native process variation. A discrete implementation will also have the processor physically separated from the RF components, making a quadrature correction algorithm very difficult to implement across frequency, temperature, and bandwidth.

1.4 RF architectures for SDR

Moore's law, or integration of the ZIF architecture into a monolithic transceiver device provides the path forward for next-generation systems. By having the analog and RF signal chain on a single piece of silicon, process variation will be kept to a minimum. Digital signal processing (DSP) blocks can be incorporated into the transceiver, removing the boundary between the quadrature calibration algorithm and the signal chain.

1.5 Processing architectures for SDR

With the latest advances in microelectronics and microprocessor systems, this has given rise to software-defined radio (SDR) technology, where baseband radio functionality can be entirely implemented in digital logic and software. There are several different types of microprocessor systems for SDR implementations, including.

- General-purpose microprocessors
- Digital signal processors (DSPs)
- Field programmable gate arrays (FPGAs)
- Graphics processing units (GPUs)
- Advanced RISC Machines (ARMs)

1.6 Software Environments for SDR

At their most fundamental level, most commercially available SDR platforms convert live RF signals to samples at digital baseband, and use a software-defined mechanism for modulation and demodulation techniques to transfer real-world data.

The boundary between the analog and digital worlds for a communication system is located at the analog-to-digital converter (ADC) and the digital-to-analog converter (DAC).

1.6 Software Environments for SDR

Typically, the radio can be configured to select center frequency, sampling rate, bandwidth, and other parameters to transmit and receive signals of interest. This leaves the modulation and demodulation techniques, which are developed using a two-step development process.

1. Develop, tune, and optimize the modulation and demodulation algorithms for a specific sample rate, bandwidth, and environment.
2. Take the above algorithm, which may be implemented in a high-level language in floating point, and code it in a production worthy environment.

1.6 Software Environments for SDR

One software environment that meets this requirement is MATLAB from MathWorks.

An additional product, Communications Systems Toolbox, adds physical layer algorithms, channel models, reference models, and connectivity to SDR hardware to transmit and receive live signals.

Another SDR software architecture is the popular open-source GNU Radio software. In GNU Radio, a variety of C++ libraries modeling different digital communications and digital signal processing algorithms are integrated together using Python and SWIG.