DOI: 10.17725/rensit.2020.12.207

A Survey of Software Radios: Reconfigurable Platforms, Development Tools, and Future Directions

[1]Hassan Nasser, [1]Abdelrazak Badawieh, [2]Abdulkarim Assalem

[1]Damascus University, http://damascusuniversity.edu.sy/
Damascus, Syrian Arab Republic
[2]Al-Baath University, http://albaath-univ.edu.sy/
Homs - p. 77, Syrian Arab Republic
Email: hsnsy.aa@gmail.com, Gbadawil@gmail.com, assalem1@gmail.com

Received March 11, 2020; peer reviewed March 20, 2020; accepted March 27, 2020

Abstract. **Software-Defined Radio (SDR) approaches for rapid prototyping of radio systems using reconfigurable hardware platforms offer significant advantages over traditional analog and hardware-centered methods. In particular, time and cost savings can be achieved by reusing tested design artefacts; this translates to supporting various features and functionalities, such as updating and upgrading through reprogramming, without the need to replace the hardware on which they are implemented. This opens the doors to the possibility of realizing multi-band and multi-functional wireless devices. Progress in the SDR field has led to the escalation of protocol development and a wide spectrum of applications, with more emphasis on programmability, flexibility, portability, and energy efficiency, in Mobile technology, Wi-Fi, and M2M communication. Consequently, SDR has earned a lot of attention and great significance to both academia and industry. SDR designers intend to simplify the realization of communication protocols while enabling researchers to experiment with prototypes on deployed networks. This Research is a survey of the state-of-the art SDR platforms and development tools in the context of wireless communication which presented an overview of SDR architecture and its basic components; and discussed the significant design trends and development tools. In addition, we reviewed available SDR platforms with an analytical comparison based on a set of metrics as a guide to developers. Finally, we offered some predictable future Directions for SDR Researches**

Keywords: **Software Defined Radio, Reconfigurability, FPGAs, Platforms, System on Chip, Radio Communications**
PACS: 01.20.+×

For citation: Hassan Nasser, Abdelrazak Badawieh, Abdulkarim Assalem. A Survey of Software Radios: Reconfigurable Platforms, Development Tools, and Future Directions. *RENSIT*, 2020, 12(2):207-218; DOI: 10.17725/rensit.2020.12.207.

Contents

1. INTRODUCTION

According to Cisco Systems, 500 Billion wireless devices for 7 Billion people are expected to be connected to the internet by 2030 [1]. Each device includes sensors that collect data, interact with environment, and communicate over a network. Therefore, the first challenge is to adjust the basic connectivity and networking layers to handle the large numbers of ends [2, 3]. There is an increasing number of wireless protocols that have been developed, such as BLE, LTE, and new Wi-Fi protocols for Machine-to-Machine (M2M) communication purposes due to different demanding requirements, one of which is high-energy efficiency. Wireless standards, generally,

are adapting quickly in order to accommodate different user needs and hardware specifications. This has called for a transceiver design with the ability to handle several protocols, including the existing ones and those being developed. In order to accomplish this task, one needs to realize the protocols need for a flexible, reconfigurable, and programmable framework.

The National Aeronautics and Space Administration (NASA) is developing an on-orbit, adaptable, Software Defined Radio (SDR)/Space Telecommunications Radio System (STRS)-based testbed facility to conduct a suite of experiments to advance technologies, reduce risk, and enable future mission capabilities on the International Space Station (ISS). The Space Communications and Navigation (SCaN) Testbed Project will provide NASA, industry, other Government agencies, and academic partners the opportunity to develop and field communications, navigation, and networking technologies in the laboratory and space environment based on reconfigurable, SDR platforms and the STRS Architecture. Led by the NASA Glenn Research Center (GRC), the SCaN Testbed was developed to move SDR space technology forward, as the advancements of Appendix Table 2 indicate. The project was previously known as the Communications, Navigation, and Networking reconfigurable Testbed (CoNNeCT) [5].

Global Industry Analysts highlights some of the market trends for SDR as follows [4]: (a) increasing interest from the military sector in building communication systems and large-scale deployment in developing countries, (b) growing demand for public safety and disaster preparedness applications, and (c) building virtualized base stations (BSs). SDRs are also ideal for developing future space communications [6], Global Navigation Satellite System (GNSS) sensors [7], Vehicle-to-Vehicle (V2V) communication [8, 9], and Internet of Things applications [10, 11], where relatively small and low-power SDRs can be utilized.

SDRs are implemented through employing various types of hardware platforms, such as General Purpose Processors (GPPs), Graphics Processing Units (GPUs), Digital Signal Processors (DSPs), and Field Programmable Gate Arrays (FPGAs). Each of these platforms is associated with their own set of challenges. Some of these challenges are: (a) Utilizing the computational power of the selected hardware platform, (b) keeping the power consumption at a minimum, ease of design process, (c) Cost of tools and equipment. The research community and industry have both developed SDRs that are based on the aforementioned hardware platforms.

2. RESEARCH METHODOLOGY AND GOALS

In this Research, we first introduced some principles and criterions that wireless communications based on. Then we presented an overview of SDR architecture as well as the analog and digital divides of the system and interconnection of components. Furthermore, we reviewed the SDR platforms developed by both industry and academia, and provided an analytical comparison of hardware platforms as a guide for design decision making. Moreover, we discuss the use of respective development tools and present a summary to help explain their functionalities and the platforms they support.

This Research is organized as follows: Section 3 provides a description of SDR architecture. Section 4 Design Mythologies & Platforms and comparison of the commercially and academically developed SDR platforms. Section 5 we review the common development tools that are typically used in the process of SDR design and implementation for different design approaches. Finally, we offered some predictable future directions for SDR researches, and concluded the paper in Section 6.

3. PRINCIPLES AND ARCHITECTURE

First, we will overview some basic principles:

- **Table 1** shows the Electromagnetic Spectrum Based on the International Telecommunications Union (ITU) [12]:
- The term Radio Spectrum Policy generally refers to "electromagnetic frequencies between 9 KHz and 3000 GHz with wavelengths between one millimeter and thousands of kilometers".
- In conventional Radio: Radio Components are implemented as analog parts or static silicon, while in Software Radio reconfigurable parts are used instead; and make the components generic so they could be used to implement several types of Radios. **Figure 1** shows this idea:
- The Institute of Electrical and Electronic Engineers (IEEE) P1900.1 Working Group [13] has created the following definition for the

Table 1

Electromagnetic Spectrum and Wavelength types

Electromagnetic spectrum						
Gamma rays	X-rays	Ultraviolet	Visible	Infrared	Microwave	Radio
ν ZHz	EHz	PHz	THz	GHz	MHz	
fm	pm	nm	μm	mm	m	km λ
← higher frequencies			longer wavelengths →			

X-rays	Ultraviolet	Visible (optical)	Microwaves	Radio spectrum (ITU)		Wavelength types
• soft X-ray • hard X-ray	• Extreme ultraviolet • Vacuum ultraviolet • Lyman-alpha • FUV • MUV • NUV • UVC • UVB • UVA	• Violet • Blue • Cyan • Green • Yellow • Orange • Red	• W band • V band • Q band • Ka band • K band • Ku band • X band • C band • S band • L band	THF	300 GHz/1 mm 3 THz/0.1 mm	• Microwave • Shortwave • Medium wave • Longwave
				EHF	30 GHz/10 mm 300 GHz/1 mm	
				SHF	3 GHz/100 mm 30 GHz/10 mm	
				UHF	300 MHz/1 m 3 GHz/100 mm	
				VHF	30 MHz/10 m 300 MHz/1 m	
				HF	3 MHz/100 m 30 MHz/10 m	
				MF	300 kHz/1 km 3 MHz/100 m	
				LF	30 kHz/10 km 300 kHz/1 km	
				VLF	3 kHz/100 km 30 kHz/10 km	
				ULF	300 Hz/1 Mm 3 kHz/100 km	
				SLF	30 Hz/10 Mm 300 Hz/1 Mm	
				ELF	3 Hz/100 Mm 30 Hz/10 Mm	

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

As shown in **Figure 2**, at a high level, a typical SDR transceiver consists of the following components: Signal Processing, Digital Front End, Analog RF Front End, and an antenna.

1. Antenna: SDR platforms usually employ several antennas to cover a wide range of frequency bands [15]. Antennas are generally referred to as "intelligent or smart" due to their ability to select a frequency band and adapt with mobile tracking or interference cancellation [14]. In the case of SDRs, an antenna needs to meet a certain list of requirements such as self-adaptation (flexibility to tuning to several bands), self-alignment (beamforming capability), and self-healing (interference rejection) [16].

2. RF Front End: This is a RF circuitry that its

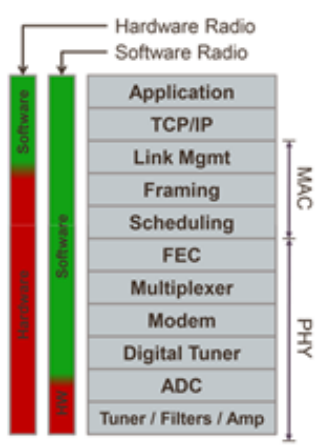

Fig. 1. *Hardware Radio vs Software Radio.*

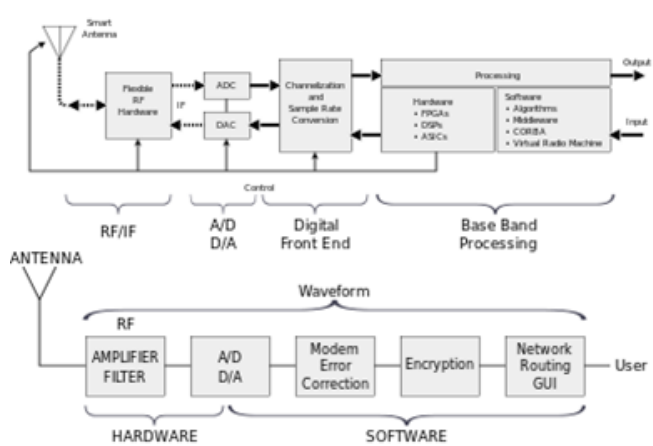

Fig. 2. *SDR transceiver architecture.*

main function is to transmit and receive the signal at various operating frequencies. Its other function is to change the signal to/from the Intermediate Frequency (IF). The process of operation is divided into two, depending on the direction of the signal (Tx or Rx mode):

- In the transmission path, the Digital-to-Analog Converter (DAC), which in turn feeds the RF Front-End, converts digital samples into an analog signal. This analog signal is mixed with a preset RF frequency, modulated, and then transmitted.
- In the receiving path, the antenna captures the RF signal. The antenna input is connected to the RF Front End using a matching circuitry to guarantee an optimum signal power transfer. It then passes through a Low Noise Amplifier (LNA), which resides in a close proximity to the antenna, to amplify weak signals and minimize the noise level. This amplified signal, with a signal from the Local Oscillator (LO), are fed into the mixer in order to down convert it to the IF [17].

3. Analog-to-Digital and Digital-to-Analog Conversion: The DAC, as mentioned in the previous section, is responsible for producing the analog signal to be transmitted from the digital samples. On the receiver side, the ADC resides, and is an essential component in radio receivers. The ADC is responsible for converting continuous-time signal to a discrete-time, binary-coded signals. ADC performance can be described by various parameters [18] including: **(1)** Signal-to-Noise Ratio (SNR): the ratio of signal power to noise power in the output, **(2)** resolution: number of bits per sample, **(3)** Spurious-free Dynamic Range (SFDR): the strength ratio of the carrier signal to the next strongest noise component or spur, and (4) power dissipation. Advances in SDR development have provided momentum for ADC performance improvements. For example, since ADC's power consumption affects the lifetime of battery-powered SDRs, more energy efficient ADCs have been developed [20].

4. Digital Front End: The Digital Front End has two main functions [19]:

- Sample Rate Conversion (SRC), which is a functionality to convert the sampling from one rate to another. This is necessary since the two communication parties must be synchronize.
- Channelization, which includes up/down conversion in the transmitter and receiver side, respectively. It also includes channel filtering, where channels that are divide by frequency are extracted. include interpolation and low-pass filters.
- In a SDR transceiver, the following tasks are executed in the digital front end:
- In the transmitting side, the Digital Up Converter (DUC) translates the aforementioned baseband signal to IF. The DAC that is connected to the DUC then converts the digital IF samples into an analog IF signal. Afterwards, the RF up-converter converts the analog IF signal to RF frequencies.
- In the receiving side, the ADC converts the IF signal into digital samples. These samples are subsequently fed into the next block, which is the Digital Down Converter (DDC). The DDC includes a digital mixer and a numerically controlled oscillator. DDC extracts the baseband digital signal from ADC, and after being processed by the Digital Front End, this digital baseband signal is forwarded to a high-speed digital signal-processing block [21].

5. Signal Processing: The signal-processing block is referred to as the baseband processing block. When discussing SDRs, the baseband block is at the heart of the discussion, because it makes up the corpus of the digital domain of the implementation. This implementation runs on top of a hardware circuitry that is capable of processing signals efficiently as ASICs, FPGAs, DSPs, GPPs, and GPUs.

4. DESIGN MYTHOLOGIES & PLATFORMS

4.1. Field Programmable Gate Array-based:

1. Zu7/5/4 Zynq UltraScale+ MPSoC development kit.

2. Arria 10 SoC/FPGA FMC+ Development platform.

3. Zynq 7000 SODIMM Development kit.

4. Altera Cyclone V SoC Development Platform.

5. Airblue [35]: It is a method to implement radios on FPGA to achieve configurability by using an HDL language called Bluespec, through which all hardware blocks of a radio transceiver are written. In Bluespec, a developer describes the execution semantics of the design through Term Rewriting

Systems (TRS). TRS is a computational paradigm based on the repeated application of simplification rules [22]. The next step is compiling the TRS into RTL codes. TRS has the capability of generating efficient hardware designs. The main difference between a Verilog interface and a Bluespec interface is that the former is merely a collection of wires with no semantic meaning, while the latter includes handshake signals for blocks communication. Therefore, Bluespec facilitates latency insensitive designs, which are essential to system construction via modular composition. Using Airblue, developers may find the need to modify the building blocks, or modules, to add new features, make algorithmic modifications, and tune the performance to meet throughput or timing requirements, or make FPGA-specific optimizations.

4.2. Digital Signal Processor-based

1. Imagine Processor-based SDR: One of the earliest SDR solutions that is fully based on a DSP developed at Stanford University in 2001 [23]. The Stanford Imagine project aimed at providing a signal and image processor that was C programmable and was able to match the high performance and density of an ASIC. It is based on stream processing [24], which is similar to dataflow programming in exploiting data parallelism and is suitable for signal processing applications. This work paved the way to the development of GPUs.

2. University of Michigan's **Software On-Demand Architecture (SODA)**: A high performance SDR platform based on multi-core DSPs with one ARM Cortex-M3 processor for control purposes and multiple processing elements for DSP operations. Using four processing elements can meet the computational requirements of 802.11a and W-CDMA.

3. ARM Ardbeg: A commercial prototype based on SODA architecture. The main enhancements of Ardbeg compared to SODA are optimized SIMD design, Very Long Instruction Word (VLIW) support, and a few special ASIC accelerators, which are dedicated to certain algorithms such as Turbo Encoder/Decoder, Floating Point and Arithmetic Operations.

4. Atomix [34]: It is a declarative language describe the software in blocks, named atoms. We can implement any operation (signal processing or system handling) by an atom. Atoms can be used for realizing blocks, flow graphs, and states in wireless stacks. In addition, simple control flow makes atoms composability. It is important to note that an Atomix signal processing block implements a fixed algorithmic function, operates on fixed data lengths, is associated with a specific processor type, and uses only the memory buffers passed to it during invocation. The blocks will run fixed sets of instructions executing uninterrupted on fixed resources using fixed memories. This results in having fixed execution times. Atoms can also be composed to build larger atoms. Using Atomix, radio software can be built entirely out of atoms and is easily modifiable. Atomix based radio also meets throughput and latency requirements. The want of Atomix is that it is intended only for synthesis on a variety of DSPs, but not for GPPs, GPUs, or FPGAs.

5. BeagleBoard-X15: A cooperative project between Texas Instruments [25], Digi-Key [26], and Newark element14 [27], BeagleBoard is an open-source SoC computer [28]. It features TI Sitara AM5728 [29], which includes two C66x DSPs, two ARM Cortex-A15, two ARM M4 cores [30], and two PowerVR SGX544 GPUs [31]. With its relatively low price, the DSPs along with the co-processors make a powerful platform for implementing standalone SDRs.

4.3. General Purpose Processor-based:

1. Universal Software Radio Peripheral N-Series [32].
2. Ziria: A programming platform uses a domain specific language (DSL) Called Ziria.
3. Sora: A fully programmable software radio platform on PC architecture [33].
4. Lime Microsystems SDR: (Field Programmable RF transceiver (FPRF) technology).
5. Kansas University Agile Radio (KUAR): A Flexible Software-Defined Radio Development Platform. In addition, a Project for an M2M application platform that provides a Java/OSGi-based container for application running in service gateways. KURA covers I/O access, data service, watchdog service, network configuration and remote [64].

4.4. Graphics Processing Unit-based:

1. OFDM for Wi-Fi Uplink SDR.
2. WiMAX SDR.
3. Signal Detection SDR.

4.5. Hybrid Design (Co-design):

1. Wireless open-Access Research Platform (WARP) [36].
2. USRP Embedded (E) Series.
3. PSoC 5LP.
4. Zynq-based SDR.

4.6. SDR Reference guide

Table 2 compares a list of available software-defined radio indicated above and others more in an effort to provide a reference guide for developers according to some standards:

Table 2

Comparison between available SDR

SDR-Name	Frequency	Bandwidth	ADC (Bit)	DAC (Bit)	Sample rate	Interface	FPGA	Price US$
Apache Labs ANAN-8000DLE [37]	0 kHz - 61.44 MHz	x	16	16	x	Gigabit Ethernet	Altera Cyclone IV	4,395
AD-FMCOMMS5-EBZ [38]	70 MHz – 6 GHz	54 MHz	12	12	61.44 MSPS	FMC (to Xilinx board) then USB 2.0 or Gigabit Ethernet.		1,125
ADALM-PLUTO [39]	325 MHz – 3.8 GHz	20 MHz	12	12	61.44 MSPS	USB 2.0, Ethernet&WLAN with USB-OTG	Xilinx Zynq Z-7010	148
AFEDRI SDR [40]	30 kHz – 35 MHz, 35 MHz – 1700 MHz	2.3MHz	12		80 MSPS	USB 2.0, 10/100 Ethernet		249
Aaronia SPECTRAN V6 [41]	20 MHz – 6 GHz	Up to 490 MHz	16	16	2 GSPS	Embedded or True IQ data via 2 x USB 3.2 Gen1, 1 x USB 3.1 GEN2	XC7A200T-2 (930 GMACs)	3,400
AirSpy R2 [42]	24 – 1700 MHz	10 MHz	12	N/A	10 MSPS ADC sampling, up to 80 MSPS for custom applications	USB	None	170
AirspyHF+ [43]	9 kHz -31 MHz (60 -260) MHz	660 kHz	18	N/A	36 MSPS	USB		199
AOR AR-2300 [44]	40 kHz – 3.15 GHz		x	N/A	65 MSPS	Embedded system (no computer needed), USB		3,299
ASR-2300 [45]	300 MHz – 3.8 GHz		x	X	<40 MHz (Programmable)	USB 3.0 SuperSpeed		1,500
Bitshark Express RX [46]	300 MHz – 4 GHz		x		105 MSPS (RX only)	PCIe		4,300
bladeRF 2.0 micro [47]	47 MHz – 6 GHz	56 MHz	12	12	61.44 MSPS	USB 3.0 SuperSpeed	Altera Cyclone V	480
ColibriDDC [48]	10 kHz – 62.5 MHz	38 – 312 kHz	14	N/A	125 MSPS	10/100 Ethernet		650
COM-3011 [49	20 MHz – 3 GHz		Ext		External ADC required (I/Q output)	USB		345
Cyan [50]	100 kHz – 18 GHz	1 – 3 GHz	12 – 16	16	1–3 GSPS ADCs 2.5 GSPS DACs	4x 40Gbps QSFP, Ethernet	Intel Stratix 10 SoC	73,500
DRB 30 [51	30 kHz – 30 MHz		Ext		External ADC required (I/Q output)	LPT parallel port		390
DX Patrol [52]	100 kHz–2GHz		8		2.4 Msps	USB		115
ELAD FDM-S1 [53]	20 kHz–30 MHz	192- 3072 kHz	14	N/A	61.44 MHz	USB	Xilinx	420
ELAD FDM-S2 [54]	HF:9 kHz – 52 MHz / FM:74 MHz - 108 MHz / VHF:135 MHz - 160 MHz	192 kHz – 6 MHz	16	N/A	122.88 MHz	USB 2.0	Xilinx Spartan-6	600
ELAD FDM-DUO [55]	HF:10 kHz – 54 MHz	192 kHz – 6 MHz	16	X	122.88 MHz	Embedded system + 3x USB 2.0	Xilinx Spartan-6	1300
Elecraft KX3 [56]	0.5 – 54 MHz (144–148 MHz optional)		14	X	30 kHz	USB or embedded system (no computer needed)		900
FLEX-6700 [57]	0.01 – 73, 135 – 165 MHz	24-192kHz	16	16	245.76 MSPS	Gigabit Ethernet	Xilinx XC6VLX130T	7000
CDRX-3200 [58]	0.01 – 100 MHz	48 – 250 kHz	24	-	48-250 kSPS	Gigabit Ethernet	Xilinx XC5VLX30T	
LBRX-24 [59]	950 – 2150 MHz	150 kHz–80 MHz	16	-	150 KSPS – 80 MSPS	10 Gigabit Ethernet (x4)	Xilinx XC6VHX380T	
FLEX-6600M [60]	0.01 – 54 MHz	24 – 192 kHz	16	16	245.76 MSPS	Gigabit Ethernet	Xilinx XC6VLX130T or XC7A200T	5000
FLEX-1500 [61]	0.01 – 54 MHz	48 kHz	16	16	48 kHz	USB	-	650
Hermes-Lite2 [62]	0 to 38.4 MHz	1.536 MHz	12	12	76.8 MSPS	Ethernet	Altera Cyclone IV	300
Iris-030 [63]	50 MHz – 3.8 GHz	122.88 MHz	12	12	122.88 Msps (SISO) 61.44 Msps (MIMO)	Gigabit Ethernet or 24.6 Gbps High-Speed Bus	Xilinx Zynq 7030	2,400
KUAR Radio [64, 65]	5 GHz	TX 30 MHz RX 600 MHz.	6	16	160 Msps	Embedded PC built + ComExpress 1.4 GHz Pentium M	Xilinx Virtex II Pro P30	-
KerberosSDR [66]	24MHz - 1.7GHz	4* sample rate	8		2.4 Msps	USB		150
LimeSDR [67]	100 kHz – 3.8 GHz	61.44 MHz (120 MHz internally)	12	X	61.44 Msps	USB 3.0, PCIe	Altera Cyclone IV	800
LimeSDR-Mini [68]	10 MHz – 3.5 GHz	30.72 MHz	12	X	30.72 Msps	USB 3.0, PCIe	Altera MAX 10	160
LD-1B [69]	100 kHz – 30 MHz		Ext		External ADC required (I/Q output)	USB		285
Matchstiq [70]	300 MHz – 3.8 GHz		x	X	40 MSPS (RX/TX)	Embedded System or USB	Xilinx Spartan 6	4,500
MB1 [71]	10 kHz – 160 MHz	38 – 312 kHz	16	14	160 MSPS (RX), 640 MSPS (TX)	10/100 Ethernet, WLAN (optional)		5,600
Mercury [72]	0.1 – 55 MHz		x		122.88 MSPS	USB (via Ozy) or Ethernet (via Metis)		469
Myriad-RF [73]	300 MHz – 3.8 GHz		x		Programmable (16 selections); 0.75 – 14 MHz, Bypass mode	standard connector FX10A-80P	None	300
Noctar [74]	100 kHz – 4 GHz	200 MHz	x		x	PCI Express ×4		2,500
Odyssey TRX [75]	0.5 – 55 MHz		x		122.880 Msps ADC sampling, 48k-960k output sample rate	LAN, Wi-Fi, USB	Altera Cyclone IV	450
Perseus [76]	10 kHz – 40 MHz	1.6 MHz	16		80 MS/s (16 bit ADC)	USB 2.0		1200
PCIe SDR MIMO [77	70 MHz – 6 GHz		x		61.44 Msps	PCIe (1x)		1700

Table 2

Comparison between available SDR (prolongation)

SDR-Name	Frequency	Bandwidth	ADC (Bit)	DAC (Bit)	Sample rate	Interface	FPGA	Price US$
PM-SDR [78]	100 kHz – 50 MHz	192 kHz	Ext		External ADC	USB		220
PrecisionWave Embedded SDR [79	1 MHz – 9.7 GHz	2x RX: 155 MHz	x		310 MSPS	Embedded System Gigabit Ethernet / USB / JTAG / Audio	Xilinx Zynq Z-7030	4000
QS1R [80	10 kHz–62.5 MHz		x		130 MHz	USB	Altera Cyclone III	1010
Quadrus [81]	0.1 – 440 MHz		x		80 Msps ADC sampling, 48k-1.536M output sample rate	PCI		1550
Realtek RTL2832U DVB-T tuner [82]	24 – 1766 MHz (R820T tuner)	Matches sampling rate	8		2.8 MHz	USB		10
RDP-100 [83]	RX, 0 – 125 MHz; TX, 0–200 MHz		x		RX: 250 MSPS TX - 800 MSPS	Embedded System		x
RTL-SDR V3 Receiver Dongle [84]	0.5 – 1766 MHz	Matches sampling rate	8		2.4 MHz	2.4 MHz		21.95-25.5
SDRplay [85]	1kHz – 2 GHz	10 MHz	14		Two independent tuners, each with 11 built-in preselection filters. 3 antenna ports	USB	None	290
SDR-IQ [86]	0.1 kHz – 30 MHz		x		66.666 MHz	66.666 MHz		525
SDR-IP [87]	0.1 kHz – 34 MHz		x		80.0 MHz	Ethernet		3000
SDR-LAB SDR04 [88]	0.4 – 4 GHz		x		40 MHz	USB 3.0 SuperSpeed		x
SDRX01B [89]	50 kHz – 200 MHz		Ext		< 2 MHz External ADC required (I/Q output)	Ethernet or USB usually, but other interfaces are available in MLAB modular system		100
SDR Minor [90]	0.1 – 55 MHz		x		122.880 Msps ADC sampling, 48k-960k output sample rate	LAN 10/100		200
SDRstick UDPSDR-HF2 [91]	0.1 – 55 MHz		x		122.88 Msps	1G Ethernet via BeMicroCV-A9	Altera	400
SDR MK1.5 Andrus [92]	5 kHz – 31 MHz (1.7 GHz downconverter opt.)		x		64 MSPS	USB 2.0, 10/100 Ethernet		480
SDR-4+ [93]	0.85 – 70.5 MHz		x		48 kHz (integrated soundcard)	USB × 2		260
SDR(X) HF, VHF & UHF [94]	0.1 – 1850 MHz (R820T tuner)		x		Optimized for HF amateur bands with 4 user selectable pre-select HF filters	USB		100
SoftRock RX Ensemble II LF [95]	180 kHz – 3.0 MHz		Ext		External ADC required (I/Q output)	USB		97
Spectre [96]	0.4 – 4 GHz	200 MHz	16		310 MSPS	USB, Serial, JTAG, 10Gbit/s SFP+ Ethernet		10,000
SunSDR2 Pro [97]	10 kHz – 160 MHz	38 – 312 kHz	16	14	160 MSPS (RX), 640 MSPS (TX)	10/100 Ethernet, WLAN (embedded)		1,595
ThinkRF WSA5000 [98]	50 MHz – 8 GHz, 18 GHz or 27 GHz		x		125 MSPS	10/100/1000 Ethernet		3,500-14,140
USRP B210 [99]	70 MHz – 6 GHz	56 MHz	x		56 Msps	USB 3.0	Xilinx Spartan 6 XC6SLX150	1,100
USRP N210 [100]	DC – 6 GHz	Up to 40 MHz	16		25 Msps for 16-bit samples; 50 Msps for 8-bit samples	Gigabit Ethernet	Xilinx Spartan 3A-DSP 3400	1,717
USRP X310 [101]	DC – 6 GHz	Up to 160 MHz	x		200 Msps	Gigabit Ethernet, 10 Gigabit Ethernet, PCIe	Xilinx Kintex-7 XC7K410T	4,800
UmTRX [102]	300 MHz – 3.8 GHz	Up to 28 MHz	12	12	13 MSPS x2	Gigabit Ethernet	Spartan 6 LX75	1,300
WARPv3 [103]	2.4 GHz and 5.8 GHz	40 MHz	12	12	40 Msps	Dual Gigabit Ethernet	Xilinx Virtex-6 LX240T	7000
WinRadio [104]	9 kHz – 50 MHz		x	N/A	100 MSPS	USB		950
XTRX Pro [105]	30 – 3700 MHz	120 MHz	12	12	120 MSRP SISO, 90 MSRP MIMO	Mini PCIe	Xilinx Artix7 50T	599

5. Development Tools

In this paragraph, we offer the common development tools, which Researches typically use in the process of SDR design and implementation for different design approaches. We also provide an overall comparison between them to highlight the differences as shown in **Table 4.**

5.1. High Level Synthesis (HLS)

Table 3 presents a summary of HLS tools. The Tools in table all provide a set of area and timing optimizations such as resource sharing, scheduling, and pipelining. Nevertheless, not all of them are capable of generating testbenches for the design.

Table 3: HLS Tools

	Xilinx Vivado HLS	Intel FPGASDK OpenCL	Cadence Stratus HLS	Synopsys Synphony C Compiler	Maxeler MaxCompiler	MATLAB HDL Coder HLS
Input	C/C++/SystemC	C/C++/SystemC	C/C++/SystemC	C/C++	MaxJ	Algorithm & Modeling
Output	VHDL/Verilog/SystemC	VHDL/Verilog	VHDL/Verilog	VHDL/Verilog/SystemC	VHDL	VHDL/Verilog/SystemC
Test bench	Yes	No	Yes	Yes	No	Yes
Optimizations	Yes	Yes	Yes	Yes	Yes	Yes
Compatibility	Xilinx FPGA	Intel FPGA	All	All	All	All

5.2. Tools

1. MATLAB & HDL Coder.

2. LabVIEW.

3. GNU Radio.

4. Vivado HLS & SDSoC.

5. Compute Unified Device Architecture (CUDA) [106].

6. Joint Tactical Radio System (JTRS) [107]: Was a program of the US military to produce radios that provide flexible and interoperable communications. Examples of radio terminals that require support include hand-held, vehicular, airborne and dismounted radios, as well as base-stations (fixed and maritime). Allow radio components to be distributed across heterogeneous computer hardware, including FPGAs, DSPs, and GPPs. This goal is achieved using SDR systems based on an internationally endorsed open Software Communications Architecture (SCA). This standard uses CORBA on POSIX operating systems to coordinate various software modules. The program is providing a flexible new approach to meet diverse soldier communications needs through software programmable radio technology. All functionality and expandability is built upon the SCA.

Table 4

Development Tools & Platforms

	MATLAB & Simulink	Vivado HLS & SDSoC	LegUP	GNU Radio	Lab View	CUDA
Input	MATLAB/Graphical	C/ C++/ System C	C	Graphical/Python/C++	Graphical	C/C++/Fortran/Python
Output	MATLAB/C++/RTL	C/RTL	C/RTL	C/RTL	C/RTL	Machine Code
Platform	GPP/GPU/DSP/FPGA	GPP/FPGA	GPP/FPGA	GPP/GPU/DSP/FPGA	GPP/GPU/DSP/FPGA	GPU
License	commercial	commercial	open-source	open-source	commercial	commercial

The SCA, despite its military origin, is under evaluation by commercial radio vendors for applicability in their domains. The adoption of general-purpose SDR frameworks outside of military, intelligence, experimental and amateur uses, however, is inherently hampered by the fact that civilian users can more easily settle with a fixed architecture, optimized for a specific function, and as such more economical in mass market applications. Still, software-defined radio's inherent flexibility can yield substantial benefits in the longer run, once the fixed costs of implementing it have gone down enough to overtake the cost of iterated redesign of purpose built systems. This then explains the increasing commercial interest in the technology.

The Open Source SCA Implementation – Embedded (OSSIE [108] project, provides SCA-based infrastructure software and rapid development tools for SDR education and research. The Wireless Innovation Forum funded the SCA Reference Implementation project, an open source implementation of the SCA specification. (SCARI) can be downloaded for free.

According to what previously mentioned, we conclude that each SDR is unique concerning the design methodology, development tools, performance, and end application.

5.3. Case Study: (Hardware-Software Co-Design Workflow for System on Chip Platforms)

This paragraph Explain an academic Example Design supports our Article idea using MATLAB & Simulink & HDL Coder to develop an SDR with a desktop computer and SoC platforms. **Figure 3** shows the design flow for SoC platforms that the aforementioned tools offer and how they are connected.

The HDL Coder Hardware-Software Co-design workflow helps automate the deployment of MATLAB and Simulink design to a Zynq-7000

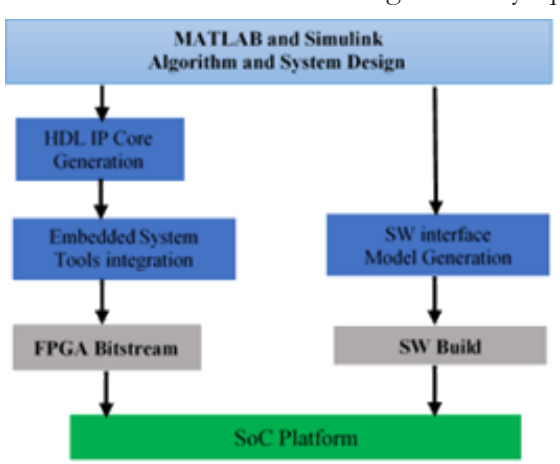

Fig. 3. *Hardware-Software Co-Design Workflow for SoC.*

platform or Intel SoC platform. We can explore the best ways to partition and deploy our design by iterating through the following workflow.

1. MATLAB and Simulink Algorithm and System Design: we begin by implementing our design in MATLAB or Simulink. When the design behavior meets requirements, decide how to partition our design: which parts we want to run in hardware, and which parts to run in embedded software. The part of the design that we want to run in hardware must use MATLAB syntax or Simulink blocks that are supported and configured for HDL code generation.

2. HDL IP Core Generation: Enclose the hardware part of our design in an atomic Subsystem block or MATLAB function, and use the HDL Workflow Advisor to define and generate an HDL IP core.

3. Embedded System Tool Integration: As part of the HDL Workflow Advisor IP core generation workflow, we insert our generated IP core into a reference design, and generate an FPGA bitstream for the SoC hardware.

The reference design is a predefined embedded system integration project. It contains all elements the Intel or Xilinx software needs to deploy our design to the SoC platform, except for the custom IP core and embedded software that we generate.

4. SW Interface Model Generation (requires a Simulink license and Embedded Coder license): In the HDL Workflow Advisor, after we generate the IP core and insert it into the reference design, we can optionally generate a software interface model. The software interface model is our original model with AXI driver blocks replacing the hardware part.

If the designer or researcher have an Embedded Coder license, he/she can automatically generate the software interface model, generate embedded code from it, and build and run the executable on the Linux kernel on the ARM processor. The generated embedded software includes AXI driver code generated from the AXI driver blocks that controls the HDL IP core.

However, if do not have an Embedded Coder license or Simulink license, he/she can write the embedded software and manually build it for the ARM processor. The following diagram shows the difference between the original model and the software interface model as shown in **Figure 4.**

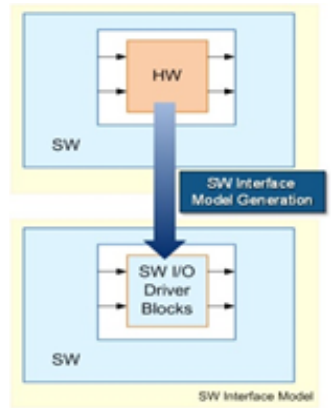

Fig. 4. *The difference between the original model and the software interface* [110].

5. SoC Platform and External Mode PIL: Using the HDL Workflow Advisor, we program our FPGA bitstream to the SoC platform. We can then run the software interface model in external mode, or processor-in-the-loop (PIL) mode, to test our deployed design.

Finally, if our deployed design does not meet the design requirements, we should repeat the workflow with a modified model, or a different hardware-software partition.

6. FUTURE DIRECTIONS & CONCLUSIONS

We thing that future Directions of SDR Researches support the goals and key actions of the Europe 2020 initiative and the Digital Single Market and in particular focuses on:

- Eliminating the digital divide;
- Efficient use of spectrum;
- Promoting investments, competition and innovation; and
- Protecting general interest objectives such as cultural diversity and media pluralism.

In addition, on the EU level, radio spectrum policy has three main goals, which are:

- The harmonization of spectrum access conditions across the Union's internal market, enabling interoperability and economies of scale for wireless equipment.
- A more efficient use of spectrum;
- Improve availability of information about the current use, plans for use and availability of spectrum.

In our Research, we presented a comprehensive overview of the various design approaches and reconfigurable platforms adopted for SDR solutions.

This includes GPPs, GPUs, DSPs, FPGAs, and Co-design. We explained the basic architectures. Then reviewed the major current and early SDR platforms, whether they were developed by the industry or in academia; Due to the different features of design approaches and development tools, we found that it was important to compare them against each other based on a set of metrics as a guide to developers. Finally, we offered some of the SDR research challenges and topics that are predicted to improve in the near future, helping to advance SDRs and their wide adoption.

Finally, we think that SDR solutions are going to be mainstream and that their ability to implement different wireless communication standards with high levels of flexibility and reprogrammability will be considered the norm. In a few last words this Research concluded to: that the driving factors for the high demand of SDR include network interoperability, readiness to adapt to future updates and new protocols, and more importantly, lower hardware and development costs.

REFERENCES

1. Cisco–Global Home Page, www.cisco.com, 2020.
2. Buyer. Software Defined Radio Market by Application, Component, End User, Type - Global Forecast to 2021. Tech. Rep., 2016. [Online]. Available: https://www.reportbuyer.com.
3. A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash. Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications. *IEEE Communications Surveys and Tutorials*, 2015.
4. Global Industry Analysts Inc. Software Defined Radio (SDR) - Global Strategic Business Report. Tech. Rep., 2017. [Online]. Available: https://www.researchandmarkets.com/research/pc9x7g/software defined.
5. Dale J. Mortensen, Daniel W. Bishop. Space Software Defined Radio Characterization to Enable Reuse. *JAN*, 2020.
6. Gara Quintana-Diaz, Roger Birkeland. *Software Defined Radioi IN SATELLITE COMMUNICATIONS*, Research-Gate, Jan 2019.
7. J. Seo, Y.-H. Chen, D.S. De Lorenzo, S. Lo, P. Enge, D. Akos, and J. Lee. A real-time capable software-defined receiver using GPU for adaptive anti-jam GPS sensors. *Sensors,* 2011, 11(9):8966-91.
8. W. Xiang, F. Sotiropoulos, and S. Liu. *Radio: An Novel Software Defined Radio (SDR) Platform and Its Exemplar Application to Vehicle-to-Vehicle Communications*. Springer, Cham, 2015, pp. 404-415.
9. M. Kloc, R. Weigel, and A. Koelpin. SDR implementation of an adaptive low-latency IEEE 802.11p transmitter system for real-time wireless applications. *IEEE Radio and Wireless Symposium,* Jan 2017, pp. 207-210.
10. Y. Chen, S. Lu, H.-S. Kim, D. Blaauw, R. G. Dreslinski, and T. Mudge. A low power software-defined-radio baseband processor for the Internet of Things. *IEEE International Symposium on High Performance Computer Architecture* (HPCA-2016), mar 2016, pp. 40-51.
11. Y. Park, S. Kuk, I. Kang, and H. Kim. Overcoming IoT Language Barriers Using Smartphone SDRs. *IEEE Transactions on Mobile Computing*, 2017, 16(3):816-828.
12. https://Imagine.gsfc.NASA.gov.
13. *IEEE Project 1900.1* - Standard Definitions and Concepts for Dynamic Spectrum Access: Terminology Relating to Emerging Wireless Networks, System Functionality, and Spectrum Management. https://standards.ieee.org/develop/project/1900.1.html.
14. A. Haghighat. A review on essentials and technical challenges of software defined radio. *Proceedings MILCOM*, 2002, pp. 377-382.
15. U.L. Rohde and T.T.N. Bucher. *Communications Receivers: Principles and Design*. McGraw-Hill Education, 1989.
16. T.J. Rouphael. *RF and digital signal processing for software-defined radio: a multi-standard multi-mode approach*. Newness, 2009.
17. J.J. Carr. *The technician's radio receiver handbook: wireless and telecommunication technology*. Newness, 2001.
18. R. Walden. Analog-to-digital converter survey and analysis. *IEEE Journal on Selected Areas in Communications*, 2000, 17(4):539-550.
19. T. Hentschel, M. Henker, and G. Fettweis. The digital front-end of software radio terminals. *IEEE Personal Communications*, 2000, 6(4):40-46.
20. C. Bowick, J. Blyler, and C. J. Ajluni. *RF circuit design*. Newness/Elsevier, 2012.
21. M.N.O. Sadiku and C.M. Akujuobi. Software-defined radio: a brief overview. *IEEE Potentials,* 2004, 23(4):14–15.

22. M.M. Bezem, J.W. Klop, R.de Vrijer. *Term rewriting systems*. Cambridge University Press, 2003.
23. B. Khailany, W.J. Dally, U.J. Kapasi, P. Mattson, J. Namkoong, J.D. Owens, B. Towles, A. Chang, and S. Rixner. Imagine: media processing with streams. *IEEE Micro*, 2002, 21(2):35-46.
24. B.K. Khailany, T. Williams, J. Lin, E.P. Long, M. Rygh, D.W. Tovey, and W.J. Dally. A Programmable 512 GOPS Stream Processor for Signal, Image, and Video Processing. *IEEE Journal of Solid-State Circuits*, 2008, 43(1):202-213.
25. *SMJ320C80 Digital Signal Processor - TI.com*. [Online]. Available: http://www.ti.com/product/SMJ320C80.
26. Digi-Key Electronics - Electronic Components Distributor. [Online]. Available: https://www.digikey.com.
27. *Newark element14 Electronics – Electronic Components Distributor*. [Online]. Available: https://www.newark.com
28. A.S. Fayez. Designing a Software Defined Radio to Run on a Heterogeneous Processor. P*h.D. dissertation, Virginia Tech*, Apr. 2011.
29. D.A. Patterson and J.L. Hennessy. *Computer organization and design ARM Edition: The Hardware Software Interface*. Morgan Kaufmann Publ., 2016.
30. *Architecting a Smarter World Arm*. [Online]. Available: https: //www.arm.com.
31. I*magination Technologies - Developing and Licensing IP cores*. [Online]. Available: https://www.imgtec.com.
32. *Ettus Research-Networked Software Defined Radio (SDR*). [Online]. Available: https://www.ettus.com.
33. K. Tan, H. Liu, J. Zhang, Y. Zhang, J. Fang, and G.M. Voelker. Sora: high-performance software radio using general-purpose multicore processors. Communications of the ACM, 2011, 54(1):99.
34. M. Bansal, A. Schulman, and S. Katti. Atomix: A Framework for Deploying Signal Processing Applications on Wireless Infrastructure. *12th USENIX Symposium on Networked Systems Design and Implementation* (NSDI 15), 2015, pp. 173–188.
35. M.-C. Ng, K. Fleming, M. Vutukuru, S. Gross, Arvind, and H. Balakrishnan. Airblue: A system for cross-layer wireless protocol development. *Symp. Architectures for Networking & Communications Syst.* (ANCS), pp. 1–11, 2010.
36. *WARP Project*. [Online]. Available: https://warpproject.org.
37. *Apache Labs*. Www. apache-labs.com. 2018.
38. http://www.analog.com/ad-fmcomms5-ebz http://wiki.analog.com/resources/eval/user-guides/ad-fmcomms5-ebz http://www.analog.com/ad9361.
39. http://www.analog.com/media/en/news-marketing-collateral/product-highlight/ADALM-PLUTO-Product-Highlight.pdf.
40. *Archived copy*. Archived from the original on 2013.
41. *SPECTRAN V6*. Www.aaronia.com. January 2020.
42. *Airspy SDR#*|Low Cost High Performance Software Defined Radio. Www.Airspy.com.
43. *Airspy HF+*. Www.Airspy.com. Retrieved 2018.
44. AR2300|RECEIVERS|AOR U.S.A., INC. *Authority On Radio Communications*. Www.Aorusa.com.
45. *Archived copy*. Archived from the original on 2014.
46. *Bitshark Express RX|Epiq Solutions*. Www. Epiqsolutions.com. July 2016.
47. *Nuand|bladeRF 2.0 micro*. Www.Nuand.com. Nov 2018.
48. *Expert Electronics - ColibriDDC*. Www.Eesdr.com. Dec 2017.
49. *COM-3011 [20 MHz - 3 GHz] Receiver/SDR*. Www.Comblock.com. 2020.
50. *Per Vices Home – Per Vices*. Www.Pervices.com. Feb 2019.
51. *Software Defined Radio - NTi Rudolf Ille Communications Technology - Products - DiRaBox*. Online 2020.
52. Www.dxpatrol.pt. 2020.
53. *FDM-S1 Receiver*. Www.Ecom.eladit.com. 2020.
54. *ELAD FDM-S2 SDR Receiver*. Www.Ecom.eladit.com. 2020.
55. *FDM-DUO SDR TRANSCEIVER*.www. Eecom.eladit.com.
56. *Elecraft® Hands-On Ham Radio™*. www.Elecraft.com.
57. *FLEX-6700-FlexRadio Systems*. Www. flexradio.com.
58. *CDRX-3200-FlexRadio Systems*. Www. flexradio.com. 2019.
59. *LBRX-24-FlexRadio Systems*. Www. flexradio.com.
60. *Lunaris SDR based on HERMES SDR Transceiver design*. www.Ceda-labz.com.
61. *FLEX-1500-FlexRadio Systems* .Www. flexradio.com.
62. *HL2 build9 specs*. 2019.
63. *Products-Skylark Wireless*. 2019
64. https://www.eclipse.org/proposals/tecnology.Kura. Sep 2016.
65. G.J. Minden, J.B. Evans, L. Searl, D. DePardo,

V.R. Petty, R. Rajbanshi, T. Newman, Q. Chen, F. Weidling, J. Guffey, D. Datla, B. Barker, M. Peck, B. Cordill, A.M. Wyglinski and A. Agah. *KUAR: A Flexible Software-Defined Radio Development Platform*. The University of Kansas, Lawrence, KS 66045.

66. *KerberosSDR - 4 Channel Coherent RTL-SDR*.
67. *LimeSDR*: Flexible, Next-generation, Open Source Software Defined Radio, Crowd Supply. Www.limesdr.org.
68. *LimeSDR-Mini*: Flexible, Next-generation, Open Source Software Defined Radio, Crowd Supply.
69. *LD-1B Software-Defined Radio*.
70. *Matchstiq s10*. Www. Epiqsolutions.com.
71. *Expert Electronics - MB1*. Www.Eesdr.com.
72. *TAPR Webmaster*, Site Design by Greg Jones, WD5IVD. "TAPR - HPSDR Mercury". Www.Tapr.org.
73. *Reference Development Kit-Myriad*. Www.Myriadrf.org.
74. *Archived copy*.
75. *Odyssey,* New open source 16-bit HF DDC SDR Transceiver Odyssey|New open source 16-bit HF DDC SDR Transceiver. Www.Ody-sdr.com.
76. *Perseus SDR Home Page*. Www.Microtelecom.it.
77. *Archived copy* (PDF).
78. *IW3AUT-HAM RADIO PROJECTS*. Www. Iw3aut.altervista.org.
79. *PrecisionWave SDR*. www.precisionwave.com.
80. Software Radio Laboratory LLC. *QS1R Software Defined Receiver*. Www.Srl-llc.com.
81. QUADRUS SDR hardware digitizer|*SDR software receiver*. Www.Spectrafold.com.
82. http://sdr.osmocom.org/trac/wiki/rtl-sdr http://www.rtlsdr.com/ http://www.rtl-sdr.com.
83. *Archived copy*.
84. *Buy RTL-SDR Dongles* (RTL2832U). Www.Rtl-sdr.com.
85. *SDRplay*. www.sdrplay.com.
86. *SDR-IQ Receiver*. Www. Rfspace.com.
87. *SDR-IP*. Www.Rfspace.com.
88. http://www.sdr-lab.com http://amitec.co.
89. http://wiki.mlab.cz/doku.php?id=en:sdrx http://www.ust.cz/shop/product_info.php?cPath=29&products_id=76.
90. КВ приемник SDR-Minor - Мои статьи - *Каталог статей - Персональный сайт*. Www.Sdr-deluxe.com.
91. *SDRstick*. Www.Sdrstick.com.
92. *UVB-76 Live Stream Blog*. Uvb-76.net.
93. *Cross Country Wireless SDR-4+ general coverage receiver*. Www.Crosscountrywireless.net.
94. *Welcome to 6V6 Electronics - 6V6 Electronics Company*.
95. *SoftRock RX Ensemble II LF Receiver Kit*. Www. Fivedash.com.
96. *Spectre|Clearbox Systems*. Www.Clearboxsystems.com.au.
97. *Expert Electronics - SunSDR2 Pro*. eesdr.com.
98. *WSA5000|ThinkRF*. Www.Thinkrf.com.
99. USRP B210 USB Software Defined Radio (SDR) - *Ettus Research*. Www.Ettus.com.
100. USRP N210 Software Defined Radio (SDR) - *Ettus Research*. Www.Ettus.com.
101. USRP X310 High Performance Software Defined Radio (SDR) - *Ettus Research*. Www.Ettus.com.
102. *Google Code Archive* - Long-term storage for Google Code Project Hosting.
103. *Mango Communications - WARP v3 Kit. Www.* Mangocomm.com.
104. *WiNRADiO WR-G31DDC 'EXCALIBUR' Receiver*. Www.Winradio.com.
105. XTRX - A Fairwaves tiny SDR. *XTRX - A Fairwaves tiny SDR*.
106. CUDA Toolkit Documentation. [Online]. Available: *http://docs. Nvidia.com/CUDA*.
107. Dsca.mil/major-arms-scales/Canada-multifunctional-information-distrbution-System-*JTRS*.
108. https://www.openhup.net.
109. https://ieeexplore.ieee.org.
110. www.MathWorks.com.

Хасан Санад Насер
Ph.D.
Дамасский университет, кафедра электроники и инженерии связи
Дамаск, Сирия
E-mail: hsnsy.aa@gmail.com, hm228844@gmail.com

Бадави Абдельразак
Dr. Eng , Professor
Дамасский университет, кафедра электроники и инженерии связи
Дамаск, Сирия
E-mail: Gbadawil@gmail.com

Ассалем Абдулкарим
Dr. Eng , Professor
Аль-Баат университет, факультет механики и электротехники
Хомс, Сирия
E-mail: assalem1@gmail.com