AIRFPGA: A SOFTWARE DEFINED RADIO PLATFORM BASED ON NETFPGA

Hongyi Zeng, John W. Lockwood, G. Adam Covington

Stanford University
Stanford, CA, USA
email: {hyzeng, jwlockwd, gcoving}@stanford.edu

Alexander Tudor

Agilent Labs
Santa Clara, CA, USA
email: alex_tudor@agilent.com

ABSTRACT

This paper introduces AirFPGA, a scalable, high-speed, remote controlled software defined radio (SDR) platform implemented using a NetFPGA board, a digitizer and a High Frequency (HF) receiver (integrated into a single device).

The AirFPGA is a system built on the NetFPGA that allows baseband recording and playback of wireless signals, as well as distributed processing. It captures radio signals, processes them in reconfigurable hardware, and sends the data via high speed Gigabit Ethernet to PCs or other NetFPGAs for additional computations. This paper describes the system architecture, data path, testbed implementation and test results. The paper demonstrates the system's verification using a signal generator. It also describes an application consisting of monitoring a commercial AM station.

1. INTRODUCTION

Software Defined Radio (SDR), which replaces typical hardware components of a radio system (e.g. mixers, filters, detectors, etc.) with personal computer software or other embedded computing devices, has recently attracted much attention in research and education.

SDRs traditionally combine data acquisition (DAQ) and signal processing on the same device or in close proximity (e.g. on the same bus). This limits the scale of DAQ(s) and strains computing capacity due to space and power availability. It is therefore desirable that DAQ and processing be performed separately, with only minimal signal processing at the acquisition point. This enables locating computing intensive signal processing resources away from DAQ.

We implemented the AirFPGA on the NetFPGA card. The AirFPGA uses the NetFPGA to transmit baseband signals over high-speed Ethernet. Each NetFPGA card has 4 Gigabit Ethernet ports. The NetFPGA is collocated with DAQ. It is connected to signal processing nodes via UDP/IP stack.

2. SYSTEM ARCHITECTURE

The AirFPGA enables the construction of a signal processing network distributed platform consisting of NetFPGAs, PCs and other computing devices. The architecture consists of four parts: receiver and digitizer (A/D converter) integrated into a single device, the NetFPGA card, the radio server, and radio clients. This is shown in Figure 1. The Radio Frequency (RF) signal received by the antenna is down-converted, digitized to IQ pairs and transferred to the NetFPGA card. The NetFPGA packetizes the received data and sends it over Gigabit Ethernet. On the other side of the network radio clients receive the packets for further processing.
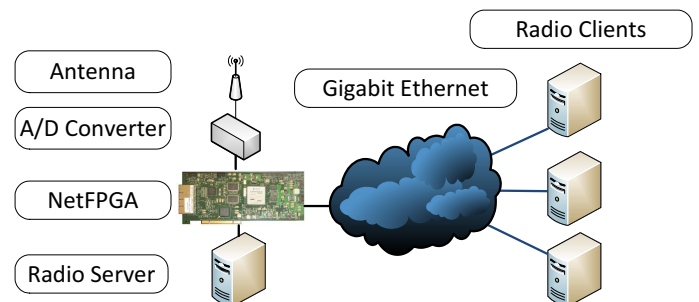

Fig. 1. AirFPGA System Architecture

2.1. NetFPGA

The NetFPGA card [1, 2] is the core of the AirFPGA system. It is a network hardware accelerator that augments the function of a standard computer. The card has four Gigabit Ethernet ports, Static RAM (SRAM) and Dynamic RAM (DRAM). The NetFPGA attaches to the Peripheral Communication Interconnect (PCI) bus on a PC. A Xilinx Virtex-II Pro FPGA on the NetFPGA performs all media access control (MAC) and network layer functions.

2.2. Radio Server

The radio server PC hosts the NetFPGA card. The packetization and forwarding are all accomplished on the FPGA.

A USB cable connects the radio [3] to the PC in which the NetFPGA card is installed. The radio server first reads data from the radio via USB cable, then loads it to NetFPGA's SRAM using the memory access tool provided by NetFPGA gateware. NetFPGA can then treat the data as though it was directly acquired from the radio, which is what we expect to accomplish in the next generation of this system. Further details will be given in Section 4.2.

2.3. Radio Client

The radio client is the receiver of packets generated by the NetFPGA card on the radio server PC. A radio client is a PC with or without a NetFPGA card. It can also be another NetFPGA card in a serial signal processing chain, or another device that has a large amount of Digital Signal Processing (DSP) resources, such as the ROACH [4].

3. FEATURES

The AirFPGA has the functionality that enables its novel use as a network distributed signal processing platform.

3.1. Scalability

The NetFPGA card has 4 Gigabit Ethernet ports. Several clients may participate in the computation needed for signal processing. Signals buffered by the NetFPGA card can be sent through Gigabit Ethernet to devices that have processing resources. In this case, which is the system's current implementation, NetFPGA is a data transport device. Other processing topologies of this "DSP Network" can be envisioned.

3.2. High speed

The AirFPGA's current design uses a single NetFPGA Gigabit Ethernet port. Up to 190kHz baseband bandwidth can be captured and streamed by the radio. Even though the on-board ADC samples 14bits at 66MS/s, the data is decimated according to the chosen narrow-band demodulation in order to fit the USB connection's 400Mb/s speed. Future work will use all four Gigabit Ethernet ports to enable wide-band modulations and MIMO.

Xilinx Virtex II Pro is suitable for processing narrow-band signals; its use for partial processing for wide-band is the subject of further investigation. For that purpose, Xilinx's DSP IP cores (e.g. numerically controlled oscillators (NCO), mixers, filters, FIFOs, etc.) will be considered.

3.3. Remote Controlled

AirFPGA separates data acquisition and signal processing. Radio clients and the radio server are logically and physically separated. You can listen to an AM radio far away from your town by deploying a radio server there. With more bandwidth you can watch HDTV received on your roof when you are abroad, or you can monitor the electromagnetic environment in a particular region.

4. IMPLEMENTATION

4.1. Reference Pipeline

The original reference pipeline of NetFPGA, as shown in Figure 2, is comprised of eight receive queues, eight transmit queues, and the user data path. The receive and transmit queues are divided into two types: MAC and CPU. The MAC queues are assigned to one of the four interfaces on the NetFPGA, and there is one CPU queue associated with each of the MAC queues.
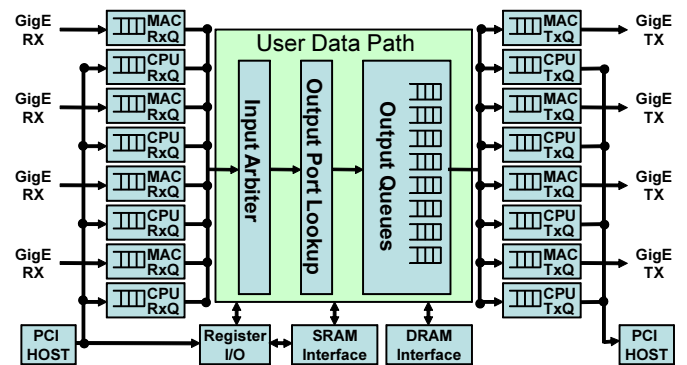

Fig. 2. NetFPGA Reference Pipeline

Users add and connect their modules to the User Data Path. The Input Arbiter and the Output Queues modules are present in almost all NetFPGA designs. The source code for these modules are provided in the NetFPGA Verilog library [5]. The Input Arbiter services the eight input queues in a round robin fashion to feed a wide (64 bit) packet pipeline.

The register system allows software to read and write the contexts of key registers, counters and the contents of SRAM from the host PC. The architecture allows modules to be inserted into the pipeline with minimal effort. The register interface allows software programs running on the host system to send data to and receive data from the hardware modules.

CTRL	NetFPGA 64bit Data Path							
-	Bits 0-7	Bits 8-15	Bits 16-23	Bits 24-31	Bits 32-39	Bits 40-47	Bits 48-55	Bits 56-63
0xFF	port_dst 16		word_length 16		port_src 16		byte_length 16	
0x00	mac_dst 48						mac_src_hi 16	
0x00	mac_src_lo 32				mac_ethertype 16		ip_version 4 + ip_header_length 4	ip_ToS 8
0x00	ip_total_length 16		ip_id 16		ip_flags 3 + ip_flag_offset 13		ip_TTL 8	ip_prot 8
0x00	ip_header_checksum 16		ip_src 32				ip_dst_hi 16	
0x00	ip_dst_lo 16		udp_src 16		udp_dst 16		udp_length 16	
0x00	udp_checksum 16		airfpga_reserved 16		airfpga_seq_num 32			
0x00	IQ 32				IQ 32			
0x00	IQ 32				IQ 32			
0x00	. . .				. . .			
0x01	IQ 32				IQ 32			

Table 1. Packet Format of AirFPGA

4.2. AirFPGA Data Path

The User Data Path of the AirFPGA is shown in Figure 3. We remove all of 8 input queues, the input arbiter, and output queue lookup module, since NetFPGA is served as a transport an data acquisition device in the current architecture.
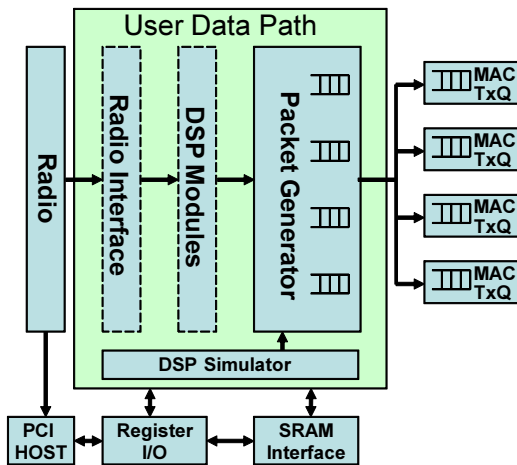

Fig. 3. AirFPGA data path. The envisioned path is shown in dotted lines, through the radio interface and DSP modules. The current path is through the DSP simulator that feeds the packet generator with "DSP results" directly from SRAM.

We are investigating additional NetFPGA functionality consisting of the radio control interface and possibly a PHY (physical layer decoder) and a PHY FEC (Forward Error Correction). The radio output is baseband IQ pairs or real numbers. Depending on how much processing takes place on the NetFPGA, IQ pairs, demodulated and decoded sig-nals, or even MAC bits are packetized according to the packet format described in Section 4.3, and sent out the Gigabit Ethernet port(s). Only IQ pairs are now packetized.

The above describes a theoretical data path, still under investigation. The current AirFPGA implementation uses an alternative data path to circumvent the radio interface and DSP modules, while preserving the overall architecture. The radio server PC loads the radio signals to the NetF-PGA's SRAM. The DSP simulator module reads the data from SRAM and feeds it to the packet generator as "DSP results". The remaining part of data path is the same as the previous path.

Radio clients receive packets using standard UDP/IP network sockets.

4.3. Packet Format

The NetFPGA data width is 64 bits, along with 8 bit control (CTRL) signals. Table 1 shows the packet format using for packetizing process. We choose UDP as transport layer protocol to achieve highest throughput and reduce the complexity of state machine design. Our applications are tolerant with potential packet loss with UDP flow. In UDP payload, we reserved first 16 bits for storing information used by the AirFPGA software, such as central frequency, power, antenna status, and data width. A 32 bit sequence number is embedded to maintain the order of packets and discover packet loss.

The headers from top down are: IOQ module headers defined in reference design for NetFPGA to route the packet correctly, IEEE 802.3 ethernet MAC header, IPv4 header, UDP header, AirFPGA header (reserved bits and sequence number). The AirFPGA payload consists a serial of 32 bit IQ data, where 16 bit I (in-phase) is preceded by 16 bit Q (quadrature).

4.4. Registers

The NetFPGA register interface exposes hardware's registers, counters and tables to the software and allows software to modify them. [6] This is achieved by memory-mapping the internal hardware registers. The memory-mapped registers appear as I/O registers to software. The software can then access the registers using ioctl calls.

The AirFPGA employs 11 main registers to realize run-time parameters modification and DSP simulator controlling. They are listed in Table 2.

Register Name	Function
Packet Parameters	
AIRFPGA_MAC_SRC_HI	High 16 bits:source MAC
AIRFPGA_MAC_SRC_LO	Low 32 bits:source MAC
AIRFPGA_MAC_DST_HI	High 16 bits:dest. MAC
AIRFPGA_MAC_DST_LO	Low 32 bits:dest. MAC
AIRFPGA_IP_SRC	Source IP
AIRFPGA_IP_DST	Destination IP
AIRFPGA_UDP_SRC	Source UDP port
AIRFPGA_UDP_DST	Destination UDP port
DSP Simulator Control	
AIRFPGA_SIM_ADDR_LO	Start addr of SRAM
AIRFPGA_SIM_ADDR_HI	End addr of SRAM
AIRFPGA_SIM_ENABLE	Enable DSP Simulator

Table 2. Registers for AirFPGA

5. TESTBED AND RESULTS

We have implemented AirFPGA on the NetFPGA board and built a testbed for verification. The architecture of the testbed is shown in Figure 5.
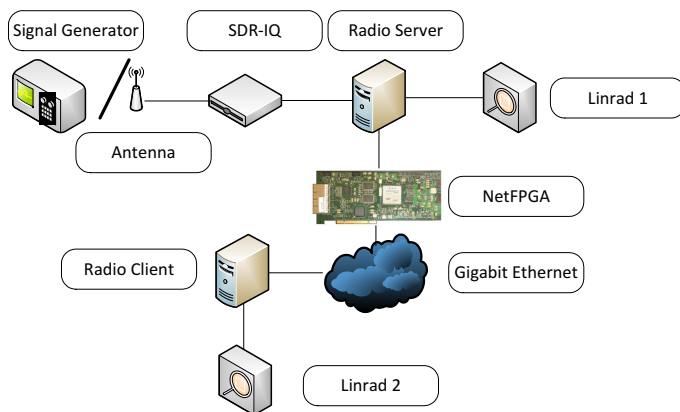

Fig. 5. AirFPGA Testbed

The AirFPGA testbed receives signals from an antenna or a signal generator. The receiver, an HF (30MHz) direct digital down-converter, can continuously stream up to 190kHz of spectrum. The product, called SDR-IQ, is manufactured by RFSpace Inc. The signal generator outputs an AM modulated RF signal. An antenna can also feed the SDR-IQ. The SDR-IQ down-converts the RF signal, digitizes it and converts it to IQ pairs. The samples are then sent to the radio server via USB. The radio server loads the IQ pairs to the NetFPGA's SRAM via the PCI interface. NetFPGA packetizes and sends them over the Gigabit Ethernet.

5.1. SDR-IQ

SDR-IQ [3] (Figure 6) converts the RF signal into IQ pairs. It features a 14 bits analog to digital converter. The device is supported by several platforms, for either Windows or Linux, as a front end for spectrum analysis and dozens narrowband modulations, both analog and digital.

Fig. 6. SDR-IQ

The hardware directly converts 30MHz to IQ pairs using a direct digital converter (DDC) chip from Analog Devices (AD6620) running at 66.6MHz.

The SDR-IQ comes with an HF amplified front-end with switched attenuators, switched filters and 1Hz tuning.

5.2. Linrad

The testbed has two Linrad's [7] running on the server and client side. Linrad is a software spectrum analyzer and demodulator running under Linux (as well as Microsoft Windows). For signal integrity verification we compare the displayed spectrum between two Linrad(s) as shown in Section 5.3.

Linrad operates with any sound card when audible modulations are used. Linrad supports the SDR-IQ and other hardware platforms.

The Linrad DSP software is independent of the hardware. It can process any bandwidth produced by the hardware subject to the computing resources of the PC on which Linrad is running. Linrad has a general purpose architecture and can be seen as a receiver design kit.
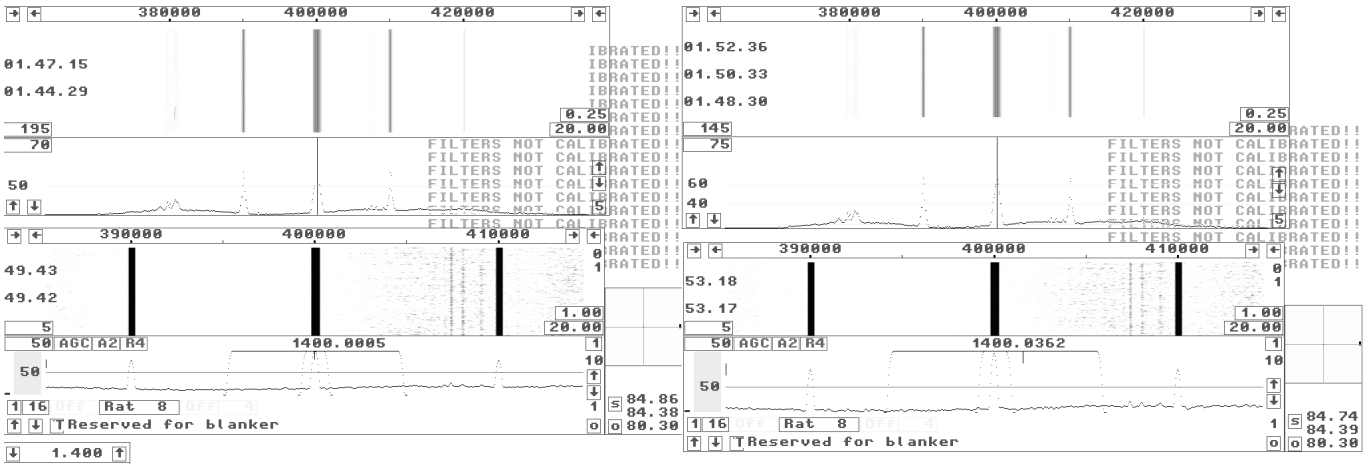
Fig. 4. 10kHz single tone AM at carrier frequency 1.4MHz. 3 main spectrum are shown on the waterfall: carrier at 1.4MHz, sidebands at 1.39MHz and 1.41MHz. The spectrum are shown on Linrad 1 (left) and Linrad 2 (right).

5.3. Signal Generator + SDR-IQ + AirFPGA

In the first scenario, we connect an Agilent E8267D PSG Vector Signal Generator to the SDR-IQ. It generates analog (AM, FM) and digital (ASK, FSK, MSK, PSK, QAM) modulated signals.

Figure 4 shows how a single tone AM signal looks like on Linrad's running at the radio server (left) and the radio client (right). The upper half of Linrad is the wideband waterfall and spectrum. The carrier is at 1.4MHz and two modulating sidebands are on 1.39MHz and 1.41MHz respectively. The baseband waterfall and spectrum is on the lower part.
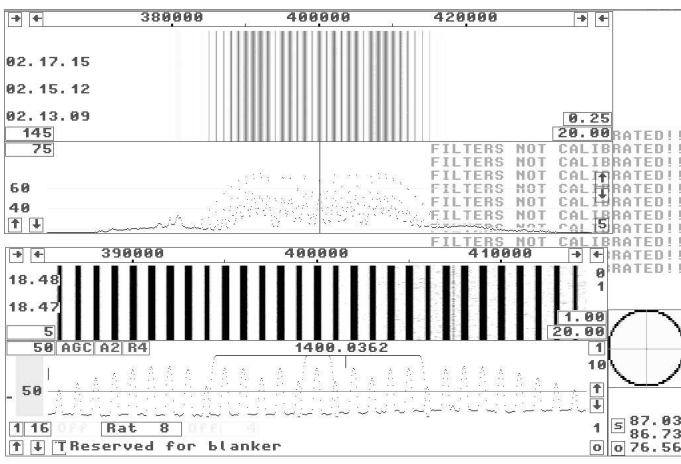

Fig. 7. 1kHz single tone ($\beta = 10$) Wideband FM signal at carrier frequency 1.4 MHz shown on client side Linrad. Several spectrum spaced by 1kHz.

Figure 7 shows how an FM signal looks like on client side Linrad. It is a 1kHz single tone signal at carrier frequency 1.4MHz. We can see several spectrum spaced by 1kHz with magnitude of n-order modified Bessel function evaluated at $\beta = 10$.

5.4. Antenna + SDR-IQ + AirFPGA

In this scenario an AM antenna is connected to the SDR-IQ. Due to the limitation of SDR-IQ (30MHz), we are only able to receive commercial AM stations (520kHz-1,610kHz).

The AM antenna is a thin wire plugged into the RF input jack of the SDR-IQ.

Figure 8 shows the spectrum of a local AM station (KLIV 1590kHz) in Bay Area. Linrad is able to demodulate the received AM signal and send the waveform to the sound card. We can listen to this station either at the server side or the client side.

A demonstration video is available online[8].

6. DEVICE UTILIZATION

Table 3 describes the device utilization of AirFPGA. AirFPGA uses 38% of the available slices on the Xilinx Virtex II Pro 50 FPGA. The largest use of the slices are from the packet generator that packetizes DSP results into IP packets. 19% of the block RAMs available are used. The main use of block RAMs occurs in the FIFOs used between the modules and the main input and output queues of the system.

7. RELATED WORK

The AirFPGA is not the first system designed for SDR. There are a number of popular platforms for the amateur radio
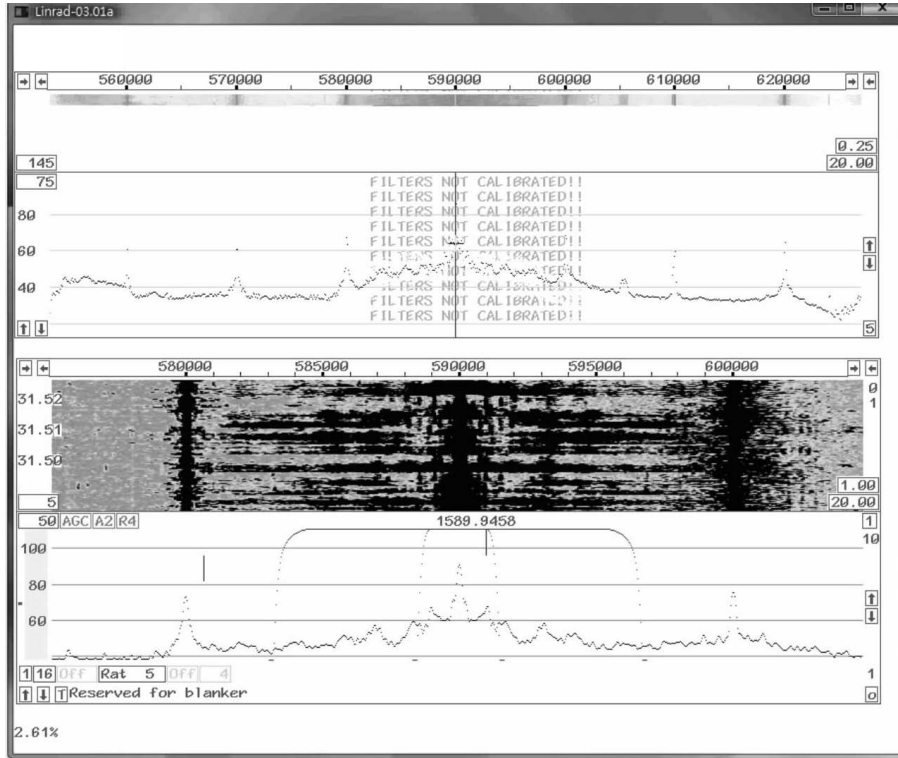
Fig. 8. KLIV (1590kHz) spectrum on AirFPGA

Resources	XC2VP50 Utilization	Utilization Percentage
Slices	9,210 out of 23,616	38%
4-input LUTS	10,204 out of 47,232	23%
Flip Flops	9,148 out of 47,232	19%
Block RAMs	46 out of 232	19%
External IOBs	356 out of 692	51%

Table 3. Device utilization for AirFPGA

community, such as SDR1000 from FlexRadio Systems [9], Softrock40 from American QRP [10]. These commercial platforms have no open source design, thus are difficult to modify by users.

The HPSDR [11] is an open source hardware and software SDR project for use by Radio Amateurs ("hams") and Short Wave Listeners (SWLs). It is being designed and developed by a group of SDR enthusiasts with representation from interested experimenters worldwide. The discussion list membership currently stands at around 750 and includes such SDR enthusiasts. However, HPSDR is designed for radio-amateur analog and digital communications.

GNU Radio [12] is a free software development toolkit that provides the signal processing runtime and processing blocks to implement software radios using readily-available, low-cost external RF hardware and commodity processors. It is widely used in hobbyist, academic and commercial environments to support wireless communications research as well as to implement real-world radio systems.

Rice University's WARP [13, 14, 15] is a scalable, extensible and programmable wireless platform to prototype wireless networks. The open-access WARP repository allows exchange and sharing of new physical and network layer architectures, building a true community platform. Xilinx FPGAs are used to enable programmability of both physical and network layer protocols on a single platform, which is both deployable and observable at all layers.

These two systems are designed for future digital wireless research, and popular in the academic world. GNU Radio and WARP both rely on centralized digital signal processing in one chip, thus lacking scalability.

Some platforms, although not designed for SDR, provide superior computing ability that can be used in SDR processing. UC Berkeley's BEE2 and ROACH are two examples. These boards can serve as radio clients in the AirFPGA architecture.

The BEE2 board [16] was originally designed for high-end reconfigurable computing applications such as ASIC design. It has 500 Gops/sec of computational power provided by 5 Xilinx XC2VP70 Virtex-II Pro FPGAs. Each FPGA

connects to 4GB of DDR2-SDRAM, and all FPGAs share a 100Mbps Ethernet port.

The ROACH board [4] is intended as a replacement for BEE2 boards. A single Xilinx Virtex-5 XC5VSX95T FPGA provides 400 Gops/sec of processing power and is connected to a separate PowerPC 440EPx processor with a 1 Gigabit Ethernet connection. The board contains 4GB of DDR2 DRAM and two 36Mbit QDR SRAMs.

8. FUTURE WORK

8.1. Multicast

The current implementation supports multicast IP address [17]. The destination IP address can be specified as a 224.x.y.z multicast IP address. Another solution is to implement 4 unicast UDP connection at the same time. The NetFPGA card has 4 Gigabit Ethernet ports. The additional ports could allow packets to be delivered to more than one port if there are multiple recipients of the data, each at a unique destination IP address. Four copies of each packet can be sent. They are different only in the destination IP address and the corresponding next-hop MAC address. The MAC address can be determined by ARP protocol, which has been implemented in the NetFPGA reference router.

8.2. Radio Daughterboard

We are developing a radio daughterboard and interface module that connects the radio directly to the NetFPGA. The daughterboard is the physical interface between the radio (consisting of RF front-end and analog to digital converter) and NetFPGA. Data is transmitted from the radio to the NetFPGA card through the daughterboard, and NetFPGA controls the radio through the same interface. The NetFPGA card provides several General Purpose Input-Output (GPIO) pins, which are suitable for interfacing with the daughterboard.

8.3. DSP Modules

We have not yet implemented any DSP modules on the NetFPGA. For the next stage we are investigating partial PHY processing of wide-band signals on the NetFPGA in order to reduce the transmitted data rate.

9. CONCLUSION

The AirFPGA is a network based SDR platform that performs distributed signal processing over high speed Ethernet. It is a novel server-client architecture designed for complex digital signal processing in wireless communication. The radio on the server side can be remotely controlled and the data it captures can be sent through Gigabit Ethernet to

remote machines. The prototype of AirFPGA has been implemented on NetFPGA along with SDR-IQ and Linrad for capturing and displaying of analog modulations. Future versions of AirFPGA will include multicast, a radio daughterboard, and local digital signal processing, etc. Further information and demonstration on the AirFPGA can be found on AirFPGA's website. [8]

10. REFERENCES

[1] G. Gibb, J. W. Lockwood, J. Naous, P. Hartke, and N. McKeown, "NetFPGA: an open platform for teaching how to build gigabit-rate network switches and routers," *IEEE Transactions on Education*, vol. 51, no. 3, pp. 364–369, Aug. 2008.

[2] J. W. Lockwood, N. McKeown, G. Watson, G. Gibb, P. Hartke, J. Naous, R. Raghuraman, and J. Luo, "NetFPGA– an open platform for gigabit-rate network switching and routing," in *Microelectronic Systems Education, 2007. MSE '07. IEEE International Conference on*, San Diego, CA, June 2007, pp. 160–161.

[3] Rfspace Inc., "SDR-IQ Receiver," http://www.rfspace.com/SDR-IQ.html.

[4] ROACH Group, "ROACH Homepage," http://casper.berkeley.edu/.

[5] NetFPGA Team, "NetFPGA website," http://netfpga.org/.

[6] G. A. Covington, G. Gibb, J. Naous, J. Lockwood, and N. McKeown, "Methodology to contribute NetFPGA modules," in *International Conference on Microelectronic Systems Education (submitted to)*, 2009.

[7] Leif Asbrink, SM5BSZ, "Linrad Website," http://www.sm5bsz.com/linuxdsp/linrad.htm.

[8] AirFPGA Group, "AirFPGA Homepage," http://www.netfpga.org/airfpga.

[9] FlexRadio Systems, "FlexRadio Website," http://www.flex-radio.com/.

[10] "SoftRock-40," http://www.amqrp.org/kits/softrock40/.

[11] HPSDR, "High Performance Software Defined Radio," http://hpsdr.org/.

[12] GNU Radio Group, "GNU Radio Website," http://www.gnu.org/software/gnuradio/.

[13] WARP Group, "WARP Website," http://warp.rice.edu.

[14] P. Murphy, A. Sabharwal, and B. Aazhang, "Design of WARP: a wireless open-access research platform," in *EURASIP XIV European Signal Processing Conference*, September 2006.

[15] K. Amiri, Y. Sun, P. Murphy, C. Hunter, J. R. Cavallaro, and A. Sabharwal, "WARP, a unified wireless network testbed for education and research," in *Microelectronic Systems Education, 2007. MSE '07. IEEE International Conference on*, San Diego, CA, June 2007, pp. 53–54.

[16] C. Chang, J. Wawrzynek, and R. W. Brodersen, "BEE2: a high-end reconfigurable computing system," *IEEE Design & Test of Computers*, vol. 22, no. 2, pp. 114–125, Mar./Apr. 2005.

[17] S. Deering, "Host extensions for IP multicasting," Internet Engineering Task Force, RFC 1112, Aug. 1989. [Online]. Available: http://www.rfc-editor.org/rfc/rfc1112.txt