A High-performance and High-programmability Reconfigurable Wireless Development Platform

Jiahua Chen[1], Tao Wang[1,3], Haoyang Wu[1], Jian Gong[1], Xiaoguang Li[1], Yang Hu[1],
Gaohan Zhang[1], Zhiwei Li[1], Junrui Yang[1], Songwu Lu[2,3]
[1]Center for Energy-Efficient Computing and Applications, School of EECS, Peking University, Beijing, China
[2]UCLA Computer Science Department, Los Angeles, CA, USA
[3]PKU-UCLA Joint Research Institute in Science and Engineering
{chenjiahua, wangtao, wuhaoyang, jian.gong, xiaoguangli2010, hy1021, gh.zhang, zhiwei.li, yjr}@pku.edu.cn
slu@cs.ucla.edu

(Demonstration Paper)

***Abstract*—The ongoing mobile Internet revolution calls for quick adoptions of new wireless communication and networking technologies. To enable such fast innovations, a software-defined platform is needed to validate and refine new algorithms, protocols, and architectures in communications and networking. Unfortunately, no current systems can meet both requirements of high programmability and high performance. In this work, we report our recent effort on building such a reconfigurable platform. We show that our proposed platform, GRT, can support both high-performance and high-programmability in a unified framework. Moreover, GRT is seamlessly integrated into the standard TCP/IP network protocol stack under Linux, and can act as a WiFi-capable, network interface card. Furthermore, it ensures backward compatibility with the popular GNU Radio platform, a user-friendly, yet low-performance system. In the demo, we will demonstrate the full functionalities of the 802.11a/g WiFi on GRT, including (1) wireless file transfer between two GRT systems at the speed of tens of Mbps; (2) execution of default Linux TCP/IP applications without changes (e.g. SSH); (3) access point (AP) operation mode, where commodity WiFi devices access the Internet via the GRT-converted AP over the WiFi channel.**

Keywords*—*wireless, FPGA, performance, programmability, reconfigurable

I. INTRODUCTION

The recent mobile Internet revolution is driven by both user demand and technology push. While user demand is readily accelerated by numerous mobile applications and services, the technology push requires fast adoption of new wireless communication and networking techniques. Indeed, communication and computer science researchers have proposed many innovative algorithms, protocols, and architectures for wireless communication and networking during the past two decades. What is missing is an appropriate development platform to prototype and validate the numerous paper designs.

Fundamentally, such wireless development platform should possess the following four features: (1) High performance - it needs to support high-throughput and low-latency wireless transmission in real time; (2) High programmability - it should be easy to program and debug to enable fast prototype and modular design; (3) Backward compatibility - it is highly desirable to remain compatible to the popular GNU Radio system (a slow yet user-friendly platform) [3], to minimize the learning curve and facilitate code reuse of legacy prototypes; (4) Seamless integration to the TCP/IP protocol stack - It should work with the default TCP/IP network protocol suite, thus facilitating cross-layer optimizations and enabling complete networking system spanning all layers.

Unfortunately, none of the existing systems can meet all four requirements. For example, GNU Radio [3] is a pure software-based system and its performance is quite low. Sora [11] uses resource over-provisioning to ensure high performance, thus requiring high-end, multi-core servers. Its programmability is only available under Windows OS. WARP [5] is an FPGA-based system and does not provide much programmability support for programming the processing pipeline in FPGA. It also works as a stand-alone component, hard to be integrated into the standard TCP/IP network protocol suite. Moreover, both Sora and WARP are not compatible to GNU Radio, and cannot reuse its large code base.

In our recent work [12], we describe a reconfigurable SDR platform called GRT. In this paper, we present a much anhanced version of GRT and demonstrate the full functionalities of the 802.11a/g WiFi on GRT. GRT meets all four requirements in a single unified framework. Our current prototype uses the full protocol family of 802.11a/g WiFi as the showcase. Our demo includes the following three scenarios: (1) wireless file transfer between two GRT systems over the 2.4GHz radio channel with the data throughput up to tens of Mbps; (2) execution of standard TCP/IP applications (e.g SSH) without any modification on GRT; (3) access point (AP) operation mode, where commodity WiFi devices access the Internet via the GRT-converted AP over the WiFi channel.

The rest of this paper is organized as follows. Section II describes the system architecture of GRT and the key techniques to support the features of high performance, high programmability, compatibility and seamless integration. Section III presents the current prototype of GRT and shows three demo cases. Section IV concludes the paper.

*This paper is supported by National Natural Science Foundation of China (61370056, 61103028)

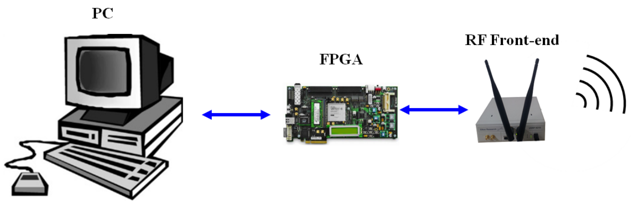


Fig. 1: GRT system architecture

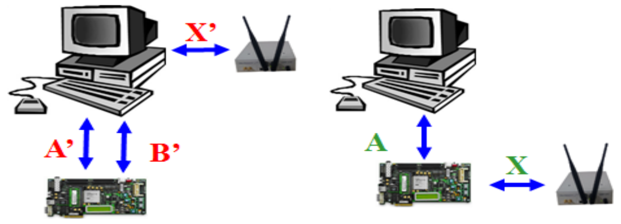


Fig. 2: Different structures for connecting the host PC, the FPGA board and the RF front-end.

II. GRT System Architecture

In this section, we describe the system components of GRT and the main techniques to ensure high performance, high programmability, backward compatibility, and seamless integration to the TCP/IP protocol stack.

A. System Components

As shown in Figure 1, a GRT system has three main components: a host PC, an FPGA broad, and a radio frequency (RF) front-end.

At the host PC, GRT works with the standard TCP/IP protocol suite under Linux through a customized device driver. It thus behaves similar to a commodity wireless network adaptor. Moreover, GRT offers a set of application programmable interfaces (APIs), in order to configure and tune various parameters, control the FPGA/RF subsystem, and monitor the system status. Furtheremore, GRT supports a user-friendly, graphical programming interface, to facilitate fast prototyping and debugging of new algorithms/protocols both at the host PC and in the FPGA board.

In the FPGA board, GRT follows the modular design. It implements throughput-intensive modules for the physical (PHY) layer and latency-sensitive functions for the low-level media access control (MAC) layer. The FPGA board is connected to the host PC via the standard PCIe interface.

At the RF front-end, radio signals (i.e., electromagnetic waves) are transmitted and received over the air. The RF front-end also converts high-frequency signals (e.g. 2.4GHz or 5GHz) to baseband ones bidirectionally, and transmits/receives the baseband signals to/from the FPGA board.

B. Techniques for High Performance

GRT applies the following three techniques to improve the throughput and latency of the entire processing pipeline.

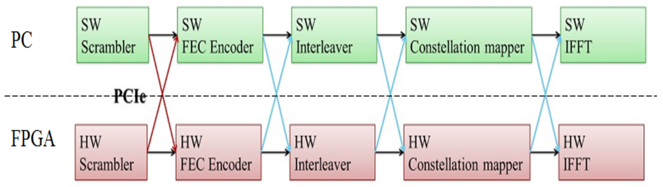


Fig. 3: A processing pipeline in GRT (software/hardware co-design)

Multi-clock processing pipeline in FPGA: The throughput of a processing pipeline in an FPGA can be improved by carefully designing/tuning each individual module. However, to make the throughput of different modules (e.g., bit-level modules and massive parallel modules) match each other, these modules may not operate at the same clock frequency. Therefore, we support multi-clock processing pipelines in FPGA in GRT, and we provide the necessary building blocks for synchronizing the modules across different clock domains as well as the programming support.

High-efficiency interconnection logic for modules in FPGA: In FPGA programming, FIFO is often used to interconnect modules for its simplicity and capability of isolating different modules. However, FIFO, especially asynchronous FIFO, is quite expensive in terms of both latency and area consumption. In GRT, we carefully design an interconnection glue logic based on registers. The FPGA modules are mostly connected with such glue logic. Moreover, the glue logic is automatically generated by the GRT programming framework.

Direct linking between FPGA and RF front-end: The RF front-end (e.g., USRP series [1], [2]) is typically connected to the host PC via the standard Ethernet/USB port, as shown in the left part of Figure 2. Therefore, the host PC has to route massive data (i.e., RF signals) between the RF front-end and the FPGA for signal processing. Such a linking practice incurs large latency overhead, introduced by both the PC driver (UHD) for the USRP (see X' in Figure 2) and the massive time-domain data transfer between the host PC and the FPGA (see B' in Figure 2).

In GRT, we have devised a customized driver module in FPGA for the RF front-end (USRP). It enables the RF front-end to directly connect to the FPGA board, as shown in the right part of Figure 2. This scheme results in 10x less traffic (A versus A'+B') from the point of view of the host PC and 20x latency reduction (X versus X'+B'). The overall system performance is thus greatly improved.

C. Techniques for High Programmability

GRT supports modular design for the processing pipeline. Each module can be implemented either as a software module on the host PC or as a hardware module on the FPGA, as shown in Figure 3. These software and hardware modules can call each other. The resulting processing pipeline can be composed of a mixture of software modules and hardware modules. This design thus supports progressive refinement in GRT, starting from an all-software implementation for fast validation of initial designs, ending up with a full-speed

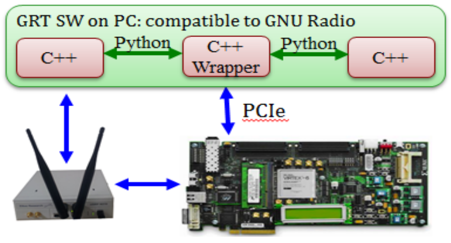


Fig. 4: GRT modules in GNU-Radio

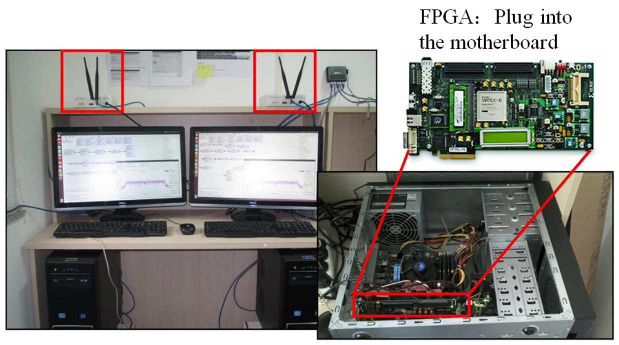
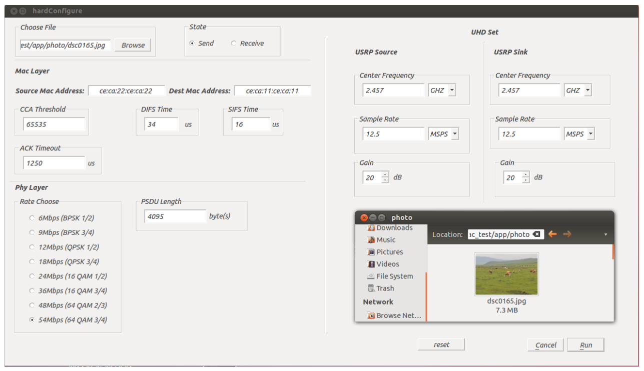


Fig. 5: Current GRT prototype



Choose File: est/app/photo/dsc0165.jpg

State: [✓ Send] Receive

Mac Layer

Source Mac Address: ce:ca:22:ce:ca:22 Dest Mac Address: ce:ca:11:ce:ca:11

CCA Threshold: 65535

DIFS Time: 34 us

SIFS Time: 16 us

ACK Timeout: 1250 us

Phy Layer

Rate Choose:
6Mbps (BPSK 1/2)
9Mbps (BPSK 3/4)
12Mbps (QPSK 1/2)
18Mbps (QPSK 3/4)
24Mbps (16 QAM 1/2)
36Mbps (16 QAM 3/4)
48Mbps (64 QAM 2/3)
[✓ 54Mbps (64 QAM 3/4)]

PSDU Length: 4095 byte(s)

UHD Set

USRP Source: Center Frequency 2.457 GHZ; Sample Rate 12.5 MSPS; Gain 20 dB

USRP Sink: Center Frequency 2.457 GHZ; Sample Rate 12.5 MSPS; Gain 20 dB

Location: ac_test/app/photo

dsc0165.jpg 7.3 MB

Fig. 6: GRT configuration panel

hardware implementation for final prototype and operation in real-world usage scenarios.

GRT offers programmability support for coding and debugging FPGA modules. To design a processing pipeline in FPGA, researchers with different levels of expertise on hardware programming can code each FPGA module with different programming languages: hardware description language (such as Verilog), the IP cores provided by FPGA vendors, or in C/C++ using a high-level-synthesis tool (e.g. [4]). The GRT programming framework supports all three programming methods.

GRT provides a software tool in the programming framework to automatically generate the glue logic for modules. A developer can focus on the functions of each module, whereas the interconnection/synchronization among the modules is handled by the GRT programming framework. This reduces programming complexity and human errors.

D. Techniques for Compatibility and Integration to TCP/IP

GRT is compatible to GNU Radio, the most popular software-based wireless development system. As shown in Figure 4, the GRT software and hardware modules can be invoked in C++ wrappers which obey the coding/execution conventions of GNU Radio C++ processing blocks. GRT modules can thus be directly used in the GNU Radio platform.

GRT is also compatible with various types of FPGA boards and RF front-end devices. GRT currently works on Xilinx ML605 [8], KC705 [7], AC701 [6] and VC707 [9] evaluation boards, and supports USRP N210 [1] and X300 [2] as the RF front-end.

GRT provides a standard Linux network interface card (NIC) driver, which allows for it to be seamlessly integrated into the Linux TCP/IP protocol suite. With this driver, GRT behaves similar to a commodity wireless network interface. In the customization mode of GRT, researchers can configure various wireless parameters (e.g., channel width, error correction level, and monitoring mode) at both PHY and MAC layers and access rich information extracted from the PHY and MAC layers. This facilitates cross-layer optimizations and facilitates coordination with upper-layer research effort such as software-defined network (SDN).

III. Prototype and Demonstration overview

The current GRT prototype is illustrated in Figure 5. It includes a normal host PC running Linux OS, a Xilinx ML605 FPGA evaluation board, and a USRP N210 RF front-end. The host PC communicates with the FPGA board via a new PCIe library [10]. The USRP N210 connects to the FPGA board via the Gigabit Ethernet port.

We use the entire protocol family of 802.11a/g WiFi as the showcase for the GRT prototype. Our demo includes the following three scenarios: (1) wireless file transfer between two GRT systems over the 2.4GHz radio channel with the data throughput up to tens of Mbps; (2) execution of standard TCP/IP applications (eg. SSH) without any modification on GRT; (3) access point (AP) operation mode, where commodity WiFi devices access the Internet via the GRT-converted AP over the WiFi channel.

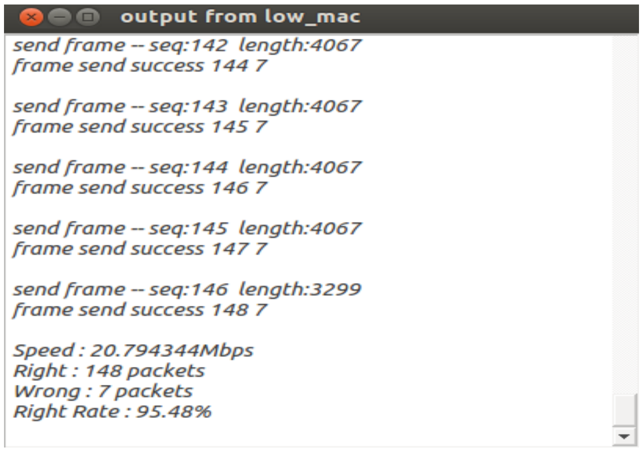
A. Wireless File Transfer over the 2.4GHz Radio Channel

In this demo, we use two GRT systems to transfer files over the 2.4GHz radio channel, following the 802.11a/g WiFi standard. In addition, certain parameters of the PHY and MAC layers can be configured during the demo, such as the center frequency, channel width, modulation scheme, frame length, slot time, acknowledgment timeout, etc. Figure 6 shows the configuration panel.

Figure 7 shows the user-observed throughput during the file transfer, up to 20.79 Mbps over the radio channel with the channel width being 12.5 MHz.

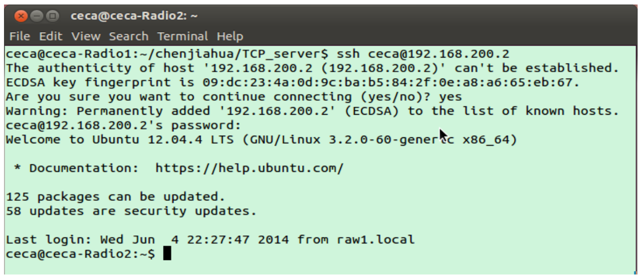
B. Running standard TCP/IP applications on GRT

GRT provides a standard Linux network interface card (NIC) driver. It can thus be readily integrated with the Linux



```
output from low_mac
send frame -- seq:142 length:4067
frame send success 144 7

send frame -- seq:143 length:4067
frame send success 145 7

send frame -- seq:144 length:4067
frame send success 146 7

send frame -- seq:145 length:4067
frame send success 147 7

send frame -- seq:146 length:3299
frame send success 148 7

Speed : 20.794344Mbps
Right : 148 packets
Wrong : 7 packets
Right Rate : 95.48%
```

Fig. 7: User-observable throughput in file transferring over the 2.4GHz radio channel with a 12.5MHz bandwidth



```
ceca@ceca-Radio2: ~
File Edit View Search Terminal Help
ceca@ceca-Radio1:~/chenjiahua/TCP_server$ ssh ceca@192.168.200.2
The authenticity of host '192.168.200.2 (192.168.200.2)' can't be established.
ECDSA key fingerprint is 09:dc:23:4a:0d:9c:ba:b5:84:2f:0e:a8:a6:65:eb:67.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.200.2' (ECDSA) to the list of known hosts.
ceca@192.168.200.2's password:
Welcome to Ubuntu 12.04.4 LTS (GNU/Linux 3.2.0-60-generic x86_64)

 * Documentation:  https://help.ubuntu.com/

125 packages can be updated.
58 updates are security updates.

Last login: Wed Jun  4 22:27:47 2014 from raw1.local
ceca@ceca-Radio2:~$
```

Fig. 8: SSH on GRT

TCP/IP protocol stack. Using this driver, GRT can behave similar to a commodity wireless network interface. This enables the execution of default Linux TCP/IP applications without modifications on GRT.

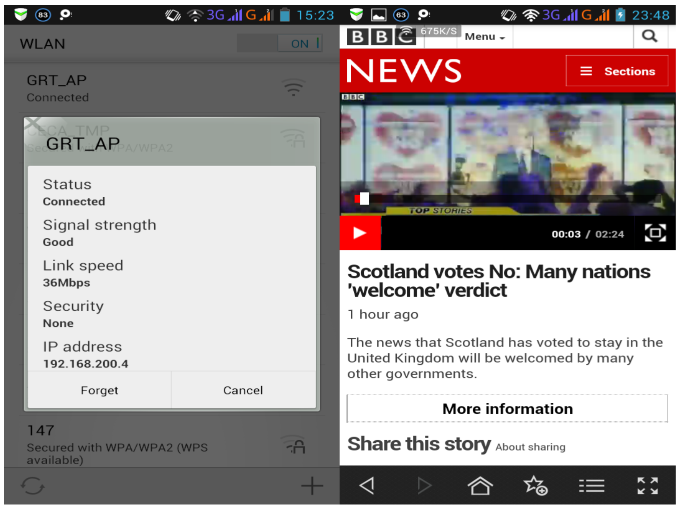
Figures 8 shows the execution of the standard application SSH on GRT.

C. GRT-converted AP for Commodity WiFi Devices

In this demo, we configure GRT to work in the access point (AP) operation mode, following the 802.11a/g WiFi standard. We set the SSID of the GRT-converted AP as GRT_AP. Commodity mobile phones with turn-on WiFi interfaces can then observe the SSID, and are associated with the GRT-converted AP. Afterwards, these smartphones can access the Internet over WiFi via the GRT AP. Figure 9 shows an associated mobile phone, which is playing an online video clip over the WiFi channel through the GRT-converted AP.

IV. Conclusion

In this work, we have reported our recent effort on building a reconfigurable platform GRT, which facilitates the research and development community to quickly validate and refine new algorithms, protocols, and architectures in communications and networking. To the best of our knowledge, GRT is the first system that can meet all four requirements of high programmability, high performance, backward compatibility, and seamless integration with TCP/IP. We have described the



GRT_AP

Status
Connected

Signal strength
Good

Link speed
36Mbps

Security
None

IP address
192.168.200.4

Forget | Cancel

147
Secured with WPA/WPA2 (WPS available)

Scotland votes No: Many nations 'welcome' verdict

1 hour ago

The news that Scotland has voted to stay in the United Kingdom will be welcomed by many other governments.

More information

Share this story About sharing

Fig. 9: An associated mobile phone playing an online video clip over the WiFi channel through the GRT-converted AP

system architecture of GRT and the key techniques used by GRT. We believe that GRT can greatly assist the researchers in the field of the wireless communication and mobile networking and enable fast renovation during the mobile Internet revolution.

In the demo, we demonstrate the full functionalities of the 802.11a/g WiFi on GRT. They include (1) wireless file transfer between two GRT systems at the speed of tens of Mbps; (2) execution of default Linux TCP/IP applications without changes (e.g. SSH); (3) access point (AP) operation mode, where commodity WiFi devices access the Internet via the GRT-converted AP over the WiFi channel.

References

[1] Ettus USRP N210. https://www.ettus.com/product/details/UN210-KIT.

[2] Ettus USRP X300. https://www.ettus.com/product/details/X300-KIT.

[3] GNU-Radio. http://gnuradio.org.

[4] Vivado-HLS. http://www.xilinx.com/products/design-tools/vivado/integration/index.htm.

[5] Warp v3 Kit. http://mangocomm.com/products/kits/warp-v3-kit.

[6] Xilinx AC701 evaluation board. http://www.xilinx.com/products/boards-and-kits/EK-A7-AC701-G.htm.

[7] Xilinx KC705 evaluation board. http://www.xilinx.com/products/boards-and-kits/EK-K7-KC705-G.htm.

[8] Xilinx ML605 evaluation board. http://www.xilinx.com/products/boards-and-kits/EK-V6-ML605-G.htm.

[9] Xilinx VC707 evaluation board. http://www.xilinx.com/products/boards-and-kits/EK-V7-VC707-G.htm.

[10] J. Gong, T. Wang, J. Chen, H. Wu, F. Ye, S. Lu, and J. Cong. An Efficient and Flexible Host-FPGA PCIe Communication Library. In *the 24th International Conference on Field Programmable Logic and Applications (FPL 2014)*, Munich, Germany, September 2-4, 2014.

[11] K. Tan, H. Liu, J. Zhang, Y. Zhang, J. Fang, and G. M. Voelker. Sora: High-performance software radio using general-purpose multi-core processors. *Commun. ACM*, 54(1):99–107, Jan. 2011.

[12] T. Wang, G. Sun, J. Chen, J. Gong, H. Wu, X. Li, S. Lu, and J. Cong. GRT: a Reconfigurable SDR Platform with High Performance and Usability. In *the 5th International Symposium on Highly Efficient Accelerators and Reconfigurable Technologies (HEART 2014)(will be published in ACM SIGARCH Computer Architecture News)*, Sendai Miyagi, Japan, June 9-11, 2014.