



## GnuRadio

**CONTACT INFORMATION:**

phone: +1.301.527.1629

fax: +1.301.527.1690

email: [whitepaper@hsc.com](mailto:whitepaper@hsc.com)

web: [www.hsc.com](http://www.hsc.com)



---

## PROPRIETARY NOTICE

All rights reserved. This publication and its contents are proprietary to Hughes Systique Corporation. No part of this publication may be reproduced in any form or by any means without the written permission of Hughes Systique Corporation, 15245 Shady Grove Road, Suite 330, Rockville, MD 20850.

Copyright © 2006 Hughes Systique Coporation

---

## TABLE OF CONTENTS

<u>SECTION</u>	<u>PAGE</u>
1.0 INTRODUCTION.....	4
1.1 HARDWARE REQUIREMENTS.....	4
1.2 THE UNIVERSAL SOFTWARE RADIO PERIPHERAL .....	4
1.3 WHAT GOES IN THE FPGA? .....	5
1.4 GNU RADIO APPLICATIONS.....	5

## 1.0 INTRODUCTION

GNU Radio is a collection of software that when combined with minimal hardware, allows the construction of radios where the actual waveforms transmitted and received are defined by software. What this means is that it turns the digital modulation schemes used in today's high performance wireless devices into software problems.

GNU Radio provides a library of signal processing blocks and the glue to tie it all together. The programmer builds a radio by creating a graph (as in graph theory) where the vertices are signal processing blocks and the edges represent the data flow between them. The signal processing blocks are implemented in C++. Conceptually, blocks process infinite streams of data flowing from their input ports to their output ports. Blocks' attributes include the number of input and output ports they have as well as the type of data that flows through each. The most frequently used types are short, float and complex.

Some blocks have only output ports or input ports. These serve as data sources and sinks in the graph. There are sources that read from a file or ADC, and sinks that write to a file, digital-to-analog converter (DAC) or graphical display. About 100 blocks come with GNU Radio. Writing new blocks is not difficult. Graphs are constructed and run in Python.

Graphical interfaces for GNU Radio applications are built in Python. Interfaces can be built using any toolkit we can access from Python. GNU Radio provides blocks that use interprocess communication to transfer chunks of data from the real-time C++ flow graph to Python-land.

(<http://www.gnu.org/software/gnuradio/>)

### 1.1 Hardware Requirements

GNU Radio is reasonably hardware-independent. Today's commodity multi-gigahertz, super-scalar CPUs with single-cycle floating-point units mean that serious digital signal processing is possible on the desktop. A 3 GHz Pentium or Athlon can evaluate 3 billion floating-point FIR taps/s. We now can build, virtually all in software, communication systems unthinkable only a few years ago.

Our computational requirements depend on what we are trying to do, but generally speaking, a 1 or 2 GHz machine with at least 256 MB of RAM should suffice. We also need some way to connect the analog world to our computer. Low-cost options include built-in sound cards and audiophile quality 96 kHz, 24-bit, add-in cards. With either of these options, we are limited to processing relatively narrow band signals and need to use some kind of narrow-band RF front end.

Another possible solution is an off-the-shelf, high-speed PCI analog-to-digital board. These are available in the 20M sample/sec range, but they are expensive, about the cost of a complete PC. For these high-speed boards, cable modem tuners make reasonable RF front ends.

Finding none of these alternatives completely satisfactory. Hardware piece Universal Software Radio Peripheral (USRP) designed by GNU solves the problem.

(<http://www.gnu.org/software/gnuradio/>)

### 1.2 The Universal Software Radio Peripheral

The USRP is an extremely flexible USB device that connects PC to the RF world. The USRP consists of a small motherboard containing up to four 12-bit 64M sample/sec ADCs, four 14-bit, 128M sample/sec DACs, a million gate-field programmable gate array (FPGA) and a programmable USB 2.0 controller. Each fully populated USRP motherboard supports four daughterboards, two for receive and two for transmit. RF front ends are implemented on the daughterboards. A variety of daughterboards is available to handle different frequency bands. For amateur radio use, low-power daughterboards are available that receive and transmit in the 440 MHz band and the 1.24 GHz band. A receive-only daughterboard based

---

on a cable modem tuner is available that covers the range from 50 MHz to 800 MHz. Daughterboards are designed to be easy to prototype by hand in order to facilitate experimentation.

The flexibility of the USRP comes from the two programmable components on the board and their interaction with the host-side library. To get a feel for the USRP, let's look at its boot sequence. The USRP itself contains no ROM-based firmware, merely a few bytes that specify the vendor ID (VID), product ID (PID) and revision. When the USRP is plugged in to the USB for the first time, the hostside library sees an unconfigured USRP. It can tell it's unconfigured by reading the VID, PID and revision. The first thing the library code does is download the 8051 code that defines the behavior of the USB peripheral controller. When this code boots, the USRP simulates a USB disconnect and reconnect. When it reconnects, the host sees a different device: the VID, PID and revision are different. The firmware now running defines the USB endpoints, interfaces and command handlers. One of the commands the USB controller now understands is load the FPGA. The library code, after seeing the USRP reconnect as the new device, goes to the next stage of the boot process and downloads the FPGA configuration bitstream.

FPGAs are generic hardware chips whose behavior is determined by the configuration bitstream that's loaded into them. We can think of the bitstream as object code. The bitstream is the output of compiling a high-level(VHDL or Verilog) description of the design.

### 1.3 What Goes in the FPGA?

Using a good USB host controller, the USRP can sustain 32 MB/sec across the USB. The USB is halfduplex.

Based on our needs, we can partition the 32 MB/sec between the transmit and the receive directions. In the receive direction, the standard configuration allows us to select the part or parts of the digitized spectrum we are interested in, translate them to baseband and decimate as required. This is exactly equivalent to what's happening in the RF front end, only now we're doing it on digitized samples. The block of code that performs this function is called a digital down converter . One advantage of performing this function in the digital domain is we can change the center frequency instantaneously, which is handy for frequency hopping spread spectrum systems. In the transmit direction, the exact inverse is performed. The FPGA contains multiple instances of the digital up and down converters. These instances can be connected to the same or different ADCs, depending on our needs.

(<http://www.gnu.org/software/gnuradio/>)

### 1.4 GNU Radio Applications

In addition to the examples discussed above, GNU Radio comes with a complete HDTV transmitter and receiver, a spectrum analyzer, an oscilloscope, concurrent multichannel receiver and an evergrowing collection of modulators and demodulators.

Projects under investigation or in progress include:

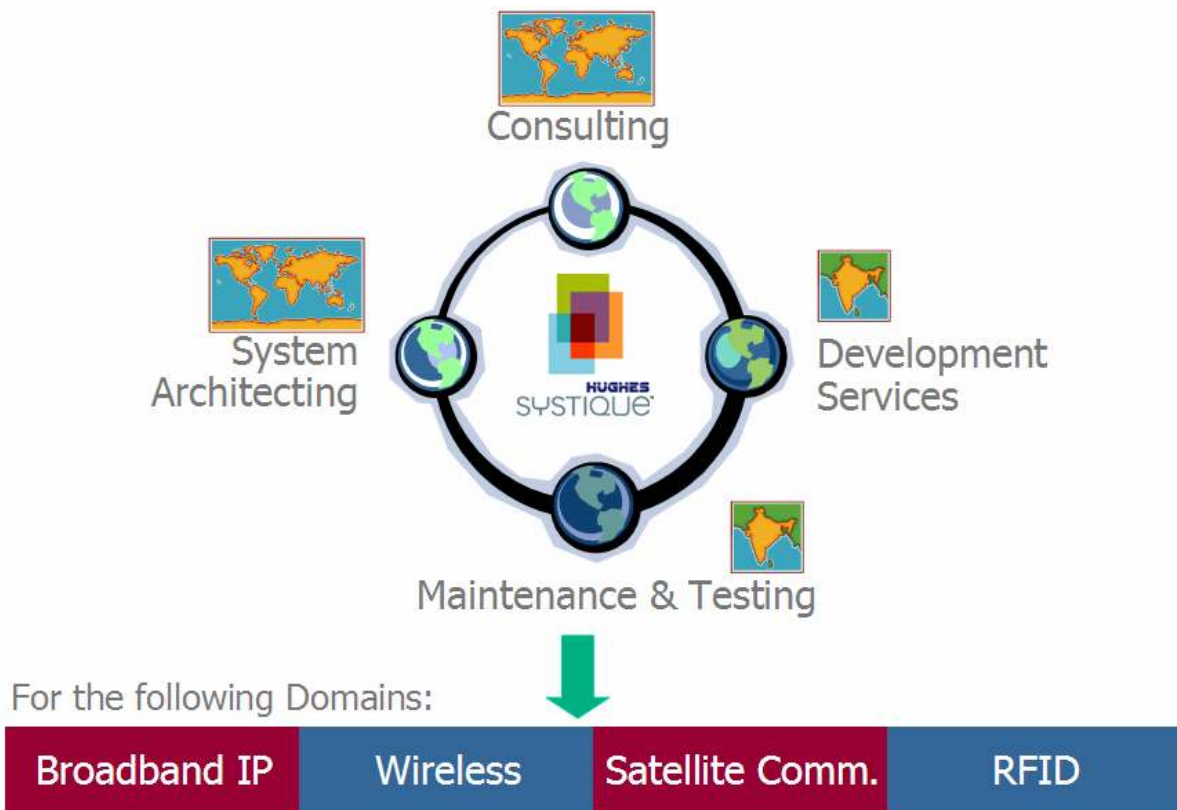
- 1 A **TIVO** equivalent for radio, capable of recording multiple stations simultaneously.
- 2 Time Division Multiple Access (TDMA) waveforms.
- 3 A passive radar system that takes advantage of broadcast TV for its signal source. For those of you with old TVs hooked to antennas, think about the flutter you see when airplanes fly over.
- 4 Radio astronomy.
- 5 TETRA transceiver.
- 6 Digital Radio Mundial (DRM).
- 7 Software GPS.
- 8 Distributed sensor networks.
- 9 Distributed measurement of spectrum utilization.
- 10 Amateur radio transceivers.
- 11 Ad hoc mesh networks.

12 RFID detector/reader.  
13 Multiple input multiple output (MIMO) processing.  
(<http://www.gnu.org/software/gnuradio/>)

## APPENDIX A ABOUT HUGHES SYSTIQUE CORPORATION

HUGHES Systique Corporation, part of the HUGHES group of companies, is a leading communications Consulting and Software company. We provide Consulting, Systems Architecture, and Software Engineering services to complement our client's in-house capabilities. Our "Best Shore" model coupled with an experienced management and technical team team is capable of delivering a total solution to our clients, from development to deployment of complex systems, thus reducing time, risk and cost

### HSC Solution Space:



#### CONTACT INFORMATION:

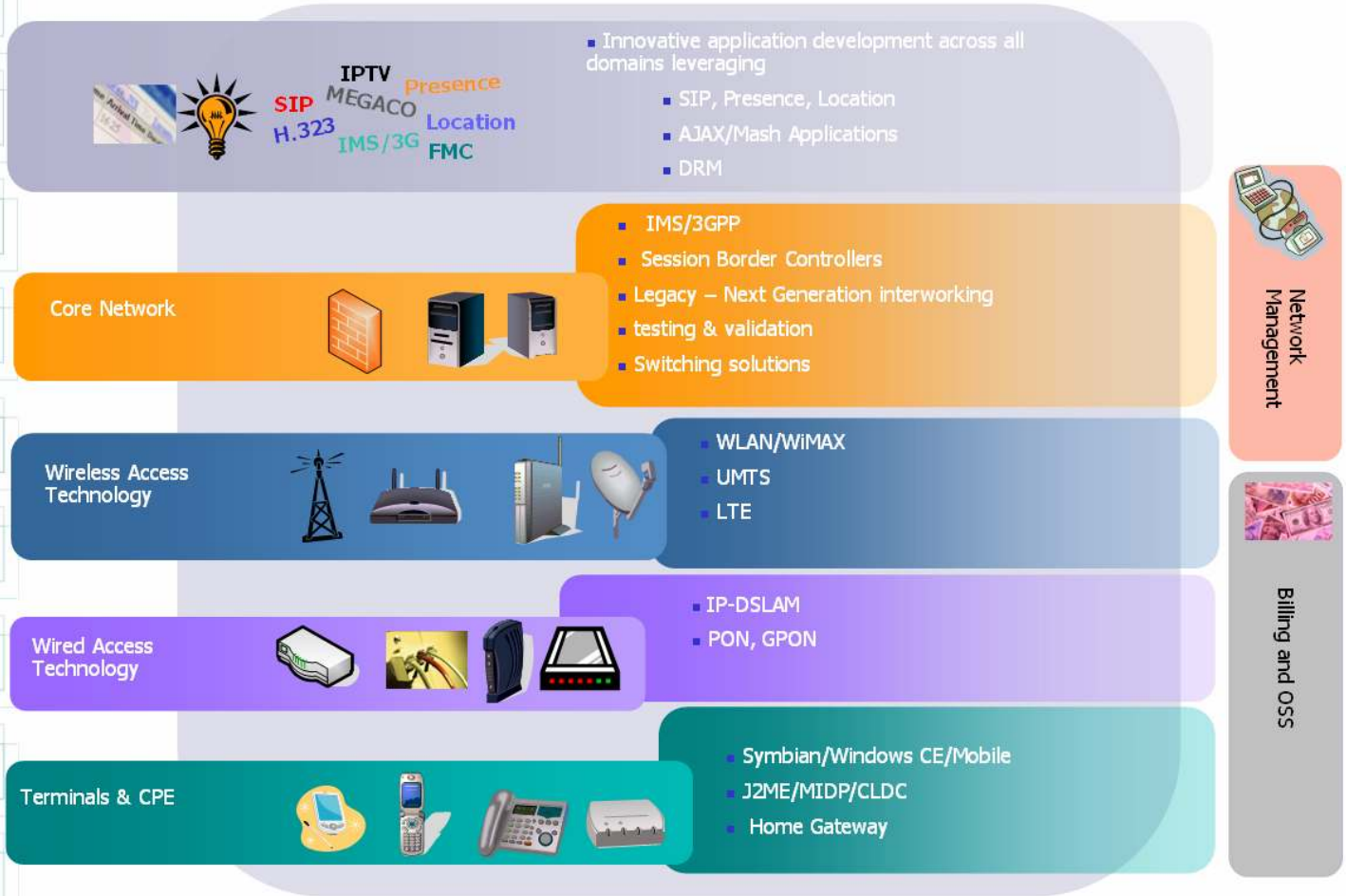
phone: +1.301.527.1629

fax: +1.301.527.1690

email: [whitepaper@hsc.com](mailto:whitepaper@hsc.com)

web: [www.hsc.com](http://www.hsc.com)

**HSC Expertise Areas in Brief:**



**CONTACT INFORMATION:**  
 phone: +1.301.527.1629  
 fax: +1.301.527.1690  
 email: [whitepaper@hsc.com](mailto:whitepaper@hsc.com)  
 web: [www.hsc.com](http://www.hsc.com)