AN UNIVERSAL FRAMEWORK FOR SCALABLE SOFTWARE DEFINED OR COGNITIVE RADIOS RUNNING ON DESKTOP COMPUTERS

Piotr Szegvari (Ph.D. Student: Brandenburg University of Technology Cottbus, Cottbus, Germany; Piotr.Szegvari@tu-cottbus.de);
Christian Hentschel (Chairholder: Brandenburg University of Technology Cottbus, Cottbus, Germany; Christian.Hentschel@tu-cottbus.de)

ABSTRACT

Software Defined Radios (SDRs) and increasingly Cognitive Radios (CRs) are attractive to replace common digital signal processing hardware solutions. Current SDRs are mostly part of communication systems using hardware front ends containing DSPs or FPGAs. Processing on Desktop Computers only is not common due to the huge amount of processing resources required. Most current Desktop Computers are not able to handle this.

We present an approach to realize SDRs on desktop computers with distributed resources. This approach is optimized for multi-core CPUs and already available General Purpose Graphics Processing Unit (GPGPU). In addition, the signal processing blocks are scalable to guarantee real-time performance at varying processing resources at runtime. The novel universal framework can be used to control the output quality of signal processing blocks related to the resource consumption. CR realization could benefit from this approach, too.

1. INTRODUCTION

Software defined radios aim for more flexibility by shifting hardware specific signal processing to software solutions. The software part becomes significantly more complex compared to hardware defined radios where just control software is needed. These radios are mostly called software controlled radios (SCRs). The processing ratio between tasks realized in hardware and in software determines the SDR software complexity. With more specific hardware components flexibility decreases while increasing software processing tasks result in higher resource demands. Due to the high resource demands, processing units such as digital signal processors (DSPs) or field programmable gate arrays (FPGAs) are used typically.

There are only few applications which include CPU resources for signal processing. The most important open source project is the GNURadio-Project [1] in combination with the universal software radio peripheral (USRP1 or USRP2) developed by Ettus Research LLC [2].

Commercial companies like CrestaTech [3] already announced solutions for universal broadcast receivers. They offer a programmable integrated circuit on an USB stick or on a PCI-express card including a multi-threaded signal processing software.

The reason for these few applications is the huge amount of data which has to be processed in real-time on a CPU. Dependent on the service to be realized, more CPU cores should run in parallel, but current SDR software hardly supports multicore CPUs. Known solutions emphasize the use of multi-threaded signal processing software algorithms following the single instruction multiple data (SIMD) principle. GNURadio supports multi-threaded signal processing and a Cell Broadband Engine (BE) processor framework. Dependent on the implemented signal processing chain, usage of both or similar principles will become necessary. Even TV receivers or transmitters could be realized with real-time performance using these architecture principles and future generations of multicore CPUs [4, 5].

We developed a universal framework exemplarily for video and audio broadcast services. As a first service a scalable FM stereo radio receiver including a RDS decoder has been implemented and evaluated concerning the scaling properties and the resource consumption [6]. This new framework runs on a desktop computer without the need of DSPs or FPGAs, similar to the GNURadio. Two hardware interfaces are supported, a PCI prototype board and the USRP2.

2. NEW UNIVERSAL FRAMEWORK

Unlike all existing solutions the new architecture takes advantage of three important concepts:

- The Resource/Quality Scalability concept, in the following referred to as software scalability,
- A concurrent programming pattern to execute multiple instructions on single data,

- An interface for using GPGPUs to execute single instructions on multiple data, in the following referred to as hardware scalability.

A combination of these concepts makes it possible to realize software defined and cognitive radios on desktop computers with highly complex processing tasks. The advantages of this combination and the concepts themselves will be explained in the following chapters including the necessity of the novel framework.

2.1. Resource/Quality Scalability

The first part contains the principle of resource dependent scalable systems. The principle has been introduced in [7, 8]. It is based on the fact that a system with fixed resources can only process a limited number of algorithms. In the case that more algorithms should be executed on such a system, it must be possible to reduce the resource consumption of already running algorithms. This property is known from resource-quality scalable algorithms.

Resource-quality scalable algorithms can adapt their resource consumption on a system by changing the output quality. Such a scalable system could be used on platforms with different resources without changing any source code. It also could be used to realize a dependable and stable system on platforms with varying resources. These concept properties are most suitable for software defined radio receivers and especially cognitive radios on desktop computers. An available system based on these principles is currently unknown.

2.2. Concurrent Programming Pattern

Multicore support should be implemented for all new software solutions in general [9, 10, 11], since the trend in the processor industry leads to multicore instead of faster single-core processing units. These processors are efficient only, if the application can be divided in highly parallelized tasks. The selected programming scheme discussed in [12] is used for the new architecture and is called the producer and consumer model. It was implemented with the "wxWidgets" library [13] to decrease the communication and polling overhead of idle threads. Each of these threads can be paused and resumed enabling this architecture to switch between different kinds of receivers. An example of measurement results for four different FM-radio receiver implementations can be seen in Table II.

2.3. Using a General Purpose GPU

The last part is the integrated GPGPU interface. Nowadays it is possible to execute concurrent general purpose processing tasks on a GPGPU [14]. The integration of such an interface into a software module which follows a single instruction multiple data scheme extends the existing architecture significantly. An additional hardware interface with an onboard DSP or FPGA is not absolutely necessary anymore. The processing tasks of these embedded processing units can be ported to a GPGPU. The principle of modularity is still available. Each module can be executed on the GPGPU, on the CPU or not at all in a bypass mode.

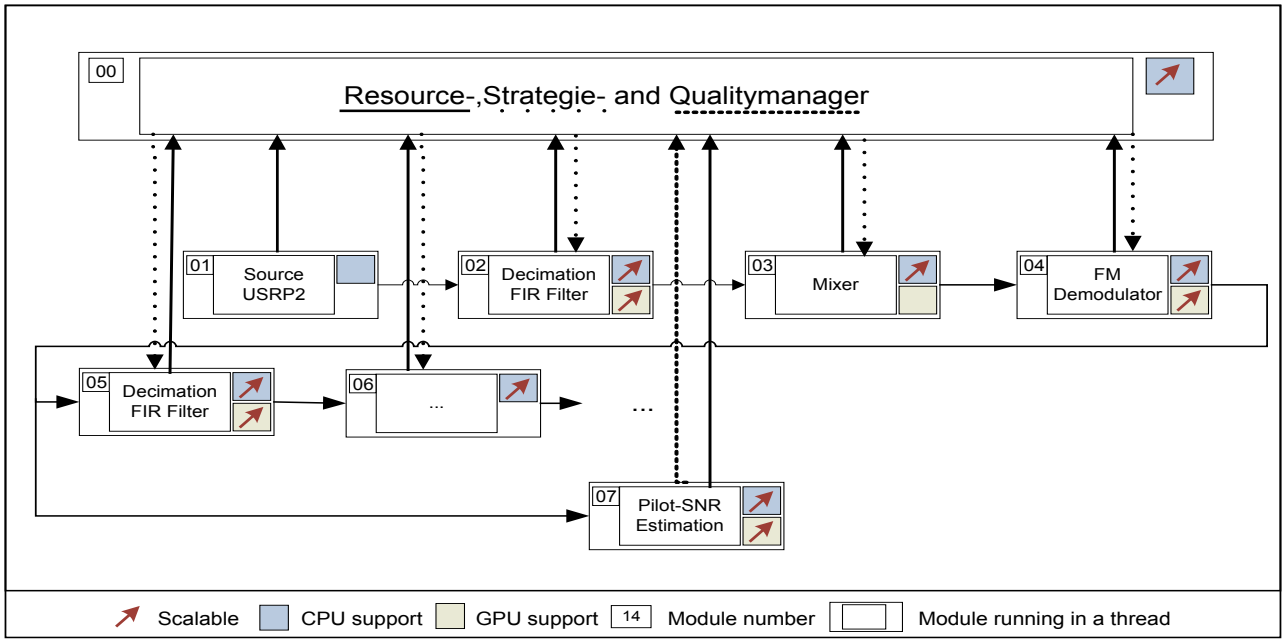

Fig. 1. Block diagram of a scalable cognitive radio receiver principle with some FM radio receiver modules.

Nevertheless, in the beginning of a signal processing chain it is still an option to run computationally complex tasks on an existing onboard DSP or FPGA.

2.4. Implementation Details

Putting all three concepts together, a new system equal to the system shown in Fig. 1 will occur. Next to a resource- and strategy manager many scalable and not scalable modules can be seen. The scalable modules could be separated into two descriptions of scalability. The first one is scalable on a CPU. The second type is scalable on both a CPU and a GPGPU. The non-scalable modules running on a CPU or on both processing units are based on algorithms with no possibility to change the resource consumption.

Still, there are some similarities between these types of modules. Each of them is working in the context of an own thread, and all of them have an internal local buffer. Communication between a module and the global resource- and strategy manager is bidirectional. Thus each module can report his measured runtime to the resource manager. Then the resource manager is reporting his information to the strategy manager. Now, the strategy manager decides on runtime which module will be executed, sets the chosen resource level, and selects the kind of processing unit. The strategy manager has the main goal to get the best quality dependent on maximum available resources.

2.5. Advantages and Drawbacks

All described concepts of this new architecture are already known and some of them are used quite often. A combination of these concepts is unknown and therefore developing a new framework was necessary. This is the biggest drawback, because every non-scalable module had to be implemented as well. In return the implementation take advantage of highly optimized arithmetic, signal processing and memory management libraries like Intels® Math Kernel Library or Integrated-Performance-Primitives [15] and Micro-Quills SmartHeap™ [16]. A second drawback is that only Nvidia® CUDA™ GPGPUs are supported currently. Integration of OpenCL is a next important step towards heterogeneous software architectures.

3. CREATING A COGNITIVE RADIO

Using a controller module based on evolutionary algorithms or reinforcement learning strategies results in a real-time capable software and hardware scalable system. Changing the quality level of each scalable module reflects the software scalability.

Changing the processing unit of scalable or not scalable modules reflects the hardware scalability. The manager modules are executed on fixed timer intervals. The resource manager monitors used resources while the strategy manager decides on quality levels depending on the available resources.

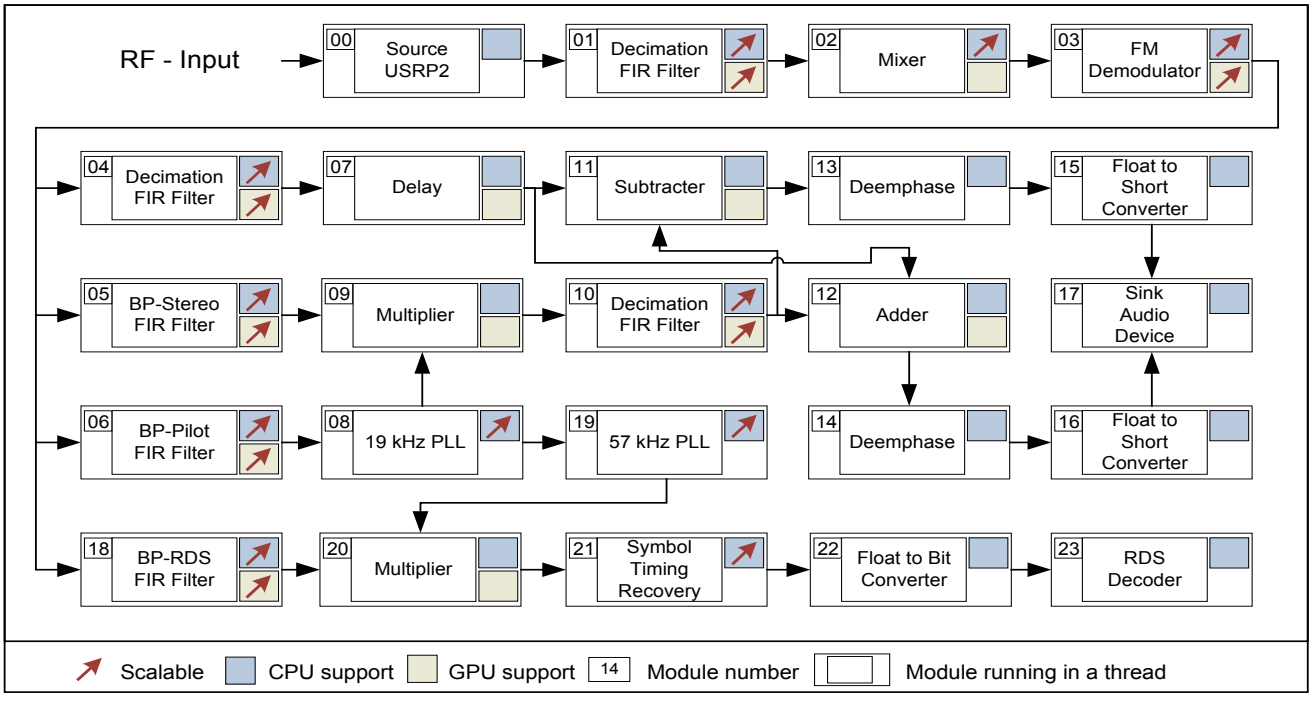

Fig. 2. Block diagram with CPU-, CPU/GPU- and Non-scalable modules of a FM-Stereo receiver including a RDS decoder.

The quality manager is optional and provides the strategy manager with additional information. The quality information is based on received measurement values from modules which can calculate either objective or instrumental quality values. In Fig. 1 an example of a signal or carrier to noise ratio estimator can be seen. This module calculates the CNR/SNR of the FM stereo pilot tone and sends this information to the quality manager.

Scaling properties can be archived by changing the number of calculations per second. If resources are left the manager calculates the CNR/SNR in a range of very long intervals providing almost continuous measurements. Dependent on the broadcasting service therewith it is possible to determine the receiver output quality. The strategy manager can detect with these calculation results, if module settings should be changed to improve the system output quality. Using such quality estimation modules converts a scalable software defined radio receiver into a scalable cognitive radio.

4. FIRST IMPLEMENTATION - A SCALABLE SOFTWARE DEFINED FM STEREO RADIO RECEIVER

For the measurements presented in this article we developed an FM-Stereo receiver including a RDS-Decoder. This receiver consists of 22 software and 2 hardware modules as shown in Figure 2. 50% of the software modules are scalable on the CPU or on the GPGPU as well. All other software modules are non scalable and will be executed on the CPU and some parallelized modules could be executed on the GPGPU, too. Each module will be executed in a context of a thread. In this example the signal source interface is the USRP2 with an input sample rate of 10 MSamples/s. Our internal PCI board solution uses an input sample rate of 27 MSamples/s.

4.1. Experimental Results

More than 50% of all implemented software modules for the FM-Stereo receiver can be executed on a GPGPU. In turn 50% of these modules are scalable on the GPGPU as well. To compare the time consumption of these modules on a GPGPU with the time consumption on a CPU it is important to differentiate three possible situations and therefore three different time measurements for each module. The first one contains the execution time including a host to device and a device to host copy of N samples. This happens if neither the previous module nor the following module is executed on the GPGPU. The second situation contains the execution time and only one copying - host to device or device to host. This happens if either the previous or the following module is executed on the GPGPU. The last situation appears if the previous and the following modules are already executed on a GPGPU.

Table I shows the different GPGPU measurement results for situation 1 and 3. Although with large block sizes the benefit of a GPU is not as high as expected for some of the modules in this receiver - for example the polyphase decimation filters. On the contrary, some modules - like the mixer, the FM-demodulator and the FIR filters could greatly benefit from the GPGPU. The advantage of GPGPU processing increases especially for higher output sample rates. And this is necessary for all kind of video broadcasting service receivers. Another advantage of the HW/SW scalable FM-receiver is the possibility to switch modules into a bypass mode which results in different receiver modes.

TABLE I
CPU VERSUS GPU RESOURCE (TIME) MEASUREMENTS

Module	States	CPU time in ms		GPU time in ms	
Name		min	max	min[b]	max[b]
Source (USRP2)	1				
Decimator	100 [a]	33	260	125|138	115|120
Mixer	3	3	28	38|30	38|30
FM-Demodulator	3	8	25	23|45	28|45
Decimator	100 [a]	3	14	40|100	30|60
Bandpass-Stereo	100 [a]	19	162	36|100	26|46
Bandpass-Pilot	100 [a]	19	162	36|120	26|50
Delay	1	0,1	0,1	38|32	38|32
PLL-Pilot	16	75	170	-	-
+ Freq. Multiply	2	9	45	29|63	29|63
Multiplier	1	1,3	1,3	29|50	29|50
Decimator	100 [a]	2,3	13	32|80	25|40
Subtracter	1	0,5	0,5	22|72	22|72
Adder	1	0,4	0,4	22|72	22|72
Deemphase	2	0	0,6	-	-
Deemphase	2	0	0,6	-	-
Float to Short	1	0,02	0,02	-	-
Float to Short	1	0,02	0,02	-	-
Sink (Sound)	1				
Bandpass-RDS	100[a]	19	163	36|100	26|50
Freq. Multiply	2	10	48	29|63	29|63
Multiplier	1	1,6	1,6	29|50	29|50
SymbolRecovery	2	3	6	-	-
Float to Bit	1	1,4	1,4	-	-
RDS-Decoder	1	14	14	-	-

a. Could be infinite but mostly 1-50 states are only noticeable.
b. X|Y – X represents the time consumption including host to device and device to host copy. Y represents the execution time if all modules running on a GPGPU with only a few important copies.

Table II shows four different receivers with the corresponding time range on a CPU and on a CPU+GPGPU.

A time consumption improvement of factor 2 could be achieved

TABLE II
MAIN RESOURCE LEVELS AND ITS SCALING RANGE

Level		Range in ms CPU only		Range in ms CPU+GPU	
Name	**Nr.**	**min**	**max**	**min**	**max**
FM-Mono	1	47	328	47	162
FM-Stereo	2	139	783	139	340
FM-Mono+RDS	3	155	724	155	346
FM-Stereo+RDS	4	188	1017	188	413

5. CONCLUSIONS

Receiving broadcasting services with high input and output sample rates using software defined radios on desktop computers is still a challenge. Dependent on the desktop computer some services could be demodulated and decoded in real-time and some of them cannot. Our experimental results show that using an available graphic processing unit in combination with a resource-quality scalable multi-core supporting framework as described in this paper could close this gap. Integration of quality measuring algorithms, like for example SNR or no-reference perceptual blur metric estimators [17] can result in a cognitive radio additionally.

6. REFERENCES

[1] GNURadio Project, http://www.gnuradio.org

[2] LLC Ettus Research, http://www.ettus.com

[3] CrestaTech, http://www.crestatech.com

[4] P. Szegvari, C. Hentschel, A. Roy, „A Scalable software Defined Radio Receiver for audio and video Broadcasting on personal computers," IEEE Internationally symposium on Consumer Electronics 04/08, ISBN 978-1-4244-2422-1, Algarve, Portugal.

[5] V. Pellegrini, G. Bacci, M. Luise, "Soft-DVB: A Fully-Software GNURadio-based ETSI DVB-T Modulator", 2008

[6] P. Szegvari, C. Hentschel, „Scalable Software Defined FM-Radio Receiver Running on Desktop Computers" ISCE 25th-28th May, in 2009, IEEE Internationally symposium on Consumer Electronics 05/09, ISBN 978-1-4244-2976-9, Kyoto, Japan.

[7] C. Hentschel, M. Gabrani, K. Van Zon, R. Bril, L. Steffens, "Scalable video algorithms and quality-of-service resource management for consumer terminals," Consumer Electronics, in 2001. ICCE. Internationally Conference on, IN 2001, 338-339

[8] C. Hentschel, R. Braspenning, M. Gabrani, "Scalable algorithms for media processing," Image Processing, in 2001. Proceedings. In 2001 Internationally Conference on, IN 2001, 3, 342-345vol/3

[9] T. Rauber, G. R., "Multicore: Parallel programming", jumper in 2008

[10] D.R. Buttenhof, "Programming with POSIX(R) Threads," Addison-Wesley, 1997

[11] A. F. Lochovsky, "Analysis of parallel algorithms using pipeline architectures in computer vision applications", Springer Netherlands, Volume 4, Numbers 1-2, March 1991, ISSN 1012-2443

[12] M.O.Thokhi, M.H.S. M.A.Hoassain H. M.A.Hossain, "the Parallel Computing for Real-time Signal Procesing and Control," jumper, in 2003

[13] J. Smart, K. Crouch, S. Csomor, „Cross-Platform GUI Programming with wxWidgets," Prentice Hall PTR, July 25, in 2005, ISBN 0-13-147381-6, Pearson Education

[14] Nvidia® CUDA™ , http://www.nvidia.co.uk/object/cuda.html

[15] MircroQuill, http://www.microquill.com

[16] Intel C++ Compiler, http://www.intel.com

[17] P. Marziliano, F. Dufaux, S. Winkler, T. Ebrahimi, „A no-reference perceptual blur metric", Proceedings., ICIP 2002