NEW ARCHITECTURE FOR DEVELOPMENT PLATFORM TARGETED TO PORTABLE APPLICATIONS

3.1 DSPs, Heterogeneous Architectures of DSPs, GPPs, FPGAs, and Configurable Signal Processing

Ram Sathappan, Texas Instruments Inc
Phone: (214) 567-3100
Email: rams@ti.com
Address:
12500 TI Boulevard
Dallas, TX 75243-4136
MS: 8677

Maxime Dumas, Lyrtech
Louis Belanger, Lyrtech
Phone: (418) 877-4644
Email:
maxime.dumas@lyrtech.com
louis.belanger@lyrtech.com
Address:
4495 Wilfrid Hamel Blvd.,
Suite 100
Quebec City, Qc Canada
G1P 2J7

Manuel Uhm, Xilinx
Phone: (408) 626-6325
Email:
manuel.uhm@xilinx.com
Address :
2100 Logic Drive
San Jose, CA, 95124

Performance, power consumption and cost are three critical factors developers face when designing software defined radios (SDR) for portable applications. In an attempt to satisfy these requirements, SDR developers consistently struggle to choose from the array of hardware and software products currently on the market. As a result, they often end up with components from multiple, independent vendors which greatly complicates waveform porting and system integration.

Using a generic development system for the design of portable radios has some drawbacks. For instance, the processors are not tailored for low-power consumption, the system is too cumbersome for field-testing, etc.

In an effort to deliver a development suite that is specifically designed for portable applications, a new architecture for SDR Development Platforms, is presented. Unlike existing solutions, the platform desired would have a small self-contained form factor, has a hybrid GPP/DSP/FPGA architecture composed of power-efficient processors, and would seamlessly integrate development tools to RF components.

This paper will present the SDR architecture needed in order to facilitate rapid and efficient development of radios mainly targeted to portable applications. It describes how this kind of platform can help radio developers in overcoming the challenges of designing low-power, small form factor devices. It covers topics like design partitioning via power consumption analysis, design flows (C/VHDL coding vs. model-based design) and tradeoff between performance, power consumption and cost. A brief discussion on the SCA-compliance of the platform will also be presented.

Need for Software Defined Radios

The Joint Tactical Radio System (JTRS) program sponsored by the US Department of Defense to develop the next generation of military communication devices envisions the use of Software Defined Radio (SDR) technology with standardized hardware capable of handling multiple protocols used by the military today. These equipments need to support a varied range of applications like secure voice, data, video and wideband networking waveforms (protocols). The goal is to achieve the perfect balance of power consumption, radio system size and overall cost while handling multi-protocol, multi-band, multi-function communications.

Software Defined Radio is a design philosophy that has been in existence for a long time now and is going through a re-birth as a result of advanced semiconductor components available today including high-performance digital signal processors and gate arrays, wide band data converters as well as advanced radio technologies. Designers of SDR also need development environments in order to help them take advantage of advanced hardware components, but those environments must also be compliant with the Software Communication Architecture (SCA) specifications to meet the portability as well as re-use requirements set forth in the JTRS program.

Combining of GPP, DSP and FPGA in SDR system architectures

When selecting architectures for military SDR applications, developers must use a combination of processing elements to achieve a balance in cost, power, performance, flexibility and reliability. Embedded signal processing systems generally use four types of advance processing devices to execute digital signal processing: application specific integrated circuits (ASICs), field programmable gate arrays (FPGAs), general purpose processors (GPP) and digital signal processors (DSP).

For modem processing in SDR systems, a combination of digital signal processing techniques and high performance logic is needed to support the different methods of modulation, demodulation, digital up/down conversion and error correction techniques.

The received RF data is either down converted to the intermediate frequency band using super heterodyne RF components or directly converted to baseband using direct conversion radios. The pre-processing in converting IF to baseband in a super heterodyne architecture generally involves serialization of data from the data converters along with down conversion in the frequency domain. This can be done either in DSP or in dedicated ASIC in the case of single waveform based radios for best in class cost and power consumption. But in SDR systems supporting multiple waveforms, this requires flexible logic and involves the use of FPGAs. The dynamic real time processing of DSP then takes over from the FPGA in the demodulation of the signal. The process is repeated in reverse during the transmission stage.

The system of error control for data transmission in wireless modems, called forward error correction, can be implemented either in DSP or logic gates depending on the type of encoding/decoding algorithms used. For example the Reed-Solomon encoding and decoding algorithms along with encoding for others like Convolutional and turbo codes is easy and better to implement on DSP for best cost/power benefits. However, the more complex cycle intensive technique of decoding of the Convolutional or Turbo error correction algorithms are best implemented using hard logic gates integrated in a processor or with the use of an FPGA. The FPGA plays a very strong co-processing role in SDR systems allowing for flexibility in the support of multiple protocols on the same radio.

The extracted data packets from the physical layer (layer 1) of the modem are passed on to the media access control layer (layer 2 or MAC) for management of the physical connection to the

network (also referred to as network processing). MAC layer processing involves encoding and decoding of packets into bits, transmission to and from network interface as well flow and conflict management of data packets within a channel. This networking processing requires the efficiency of Real Time Operating Systems (RTOS) and involves a number of control processing functions. The best processing elements to implement MAC functions as well as the memory management needed in RTOS require micro-processors or micro-controllers generally referred to as general purpose processors.

Thus the combination of GPP, DSP and FPGA components are a necessity in SDR systems to implement multi-band multi-protocol radios for the military.

Development Platforms for rapid proof-of-concept designs

There are many different SDR development environments available today. Most of the third party SDR development environments have a form factor, performance capability, power consumption and cost structure that is tuned and targeted for high performance infrastructure type of applications like multi-channel radio transceivers for aircrafts, ships and central communication centers. The development environments available today are generally of the PCI, VME or PMC form factors and incorporate multiple DSPs and FPGAs on the hardware to support multiple channels of processing. This has a direct impact on power consumption. In addition, very few if any of the development hardware incorporate RF components for rapid proof of concept development. What is missing is a development environment that has high levels of integration in hardware, software and tool flow to support development of small form factor radios for handheld, Manpack and vehicle based radios.

Given that every project has its own specificities, the development platform should be as modular as possible. Normally, radios are divided into three sections: processing stage, conversion stage, RF stage. This allows replacement of a module if the default setup cannot meet certain requirements of an application.

Choice of the conversion board depends directly on the up/down conversion topology of the RF front-end. While super heterodyne with digitization in baseband and zero-IF require DC-coupled converters, a radio using super heterodyne with digitization in IF or direct digitization (RF sampling) would have better performances if converters were AC-coupled. [# of bits, sampling rate etc.]

On the RF side, considerations are mainly application specific, such as which frequency bands need to be covered, and the bandwidth of the channels to be processed. In some cases, developers might want to digitize multiple channels. Motivations for that are numerous: digital channelization (block radio), implementation of frequency hopping in digital domain, scan to identify used/unused portions of the spectrum, etc.

For the baseband portion, the key aspect to be addressed is sufficient processing headroom to implement most complex waveforms. In addition to the processing, a good amount of I/Os should be available to the developers, such as audio input/output interface, video or display output, buttons or keyboard interface, switches, LEDs, and GPIO lines so as to interact directly with the platform as a real radio. Finally, the right connectivity and interface options (Ethernet, RS-232, GPIO) should be present, such that the baseband module can be used as the black-side radio and interface with the red-side equipment.

Because cost and power consumption are two critical factors when designing for portable applications, developers need to have easy access to power consumption and processor utilization. While getting an averaged power measurement gives a broad idea of the global consumption, it does not provide any information on pieces of a waveform that are active only part of the time. For instance, measuring a radio that runs a TDMA waveform will only provide an averaged consumption of the transmit, receive and idle (or other) modes. More useful to the

developer is to embed power monitoring capability separately for all processors on the development platform. This way, it is possible to associate power measurements with specific portions of the waveform. Processor utilization is valuable information for developing end products. First, it provides the knowledge required to scale down to a smaller, cheaper, less power consuming processor, and secondly to select the minimum clock frequency needed which also impacts power consumption. Additionally, it helps in deciding how a waveform should be partitioned, or repartitioned amongst the processing choices on the system.

Having a development platform with a form factor such as compactPCI is impractical for field testing. Its size and weight makes it cumbersome to deploy in the field. Moreover, it often requires mains voltage and a significant source of power, as it typically needs more than 60W to operate. Therefore, for field testing, small form factor and low power consumption are required. In addition to these specificities, deploying a system should be possible with minimal effort and equipment. To handle this, a bootloader is automatically executed at startup and searches the permanent memory to find a profile. If one is found, it is loaded to the processor. Otherwise, it seeks for a host to connect to. This functionality allows the radio to be operational right after power-on without any further action.

Software Tools and Model based design Flow

Recent surveys showed that the main selection criterion of processors is not the processor itself, but the tools for developing on them. The same should hold true for platforms. Consequently, platforms need to come with a complete board support package, so that developers only care about waveform development, rather than device drivers or interconnections. This board support package commonly comes as libraries to include in the IDE, ready to integrate with the C or VHDL code of the waveforms.

At a higher level of abstraction, the board support package integrates to a model-based development environment, such as Simulink. The design flow inherent to this board support package starts with a model that can be simulated. Once the simulation results are satisfying, the developer gradually migrates to a hardware specific model by replacing the generic blocks in the model with target specific blocks. Then come the platform specific blocks, interfacing with all I/Os of the board. The development environment then generates code from the model created and sends that code to the processors IDE, which will compile the generated code and load the executables to the processors.

In-between those two solutions lie multiple options that developers can take advantage of. As a first example, part of a team might prefer to hand-code DSP algorithms while the other part of the team wants to take advantage of model-based design and code generation for the FPGA. The board support package shall allow for such situations. Model-based design shall also support IP reuse by being able to include legacy code among the other blocks. This is done in Simulink by using S-functions for the DSP, and Black Boxes for the FPGA. At the opposite, a developer can integrate his model-based algorithm to the low-level coded design of the rest of the team with the use of Embedded Coder.

Software Communications Architecture

Next generation radio systems in order to have portability of code and standardization of systems across vendors and users will require waveform development and maintenance under the Software communications architecture framework. The SCA specifications as defined by the SDR Forum has been adopted as the standard for all future communication systems of the US military. Development platforms that would aid in the development of these next generation military radios need to be SCA compatible and support development of waveforms using the SCA framework and Object request broker (ORB) middleware to allow for waveform portability and code abstraction from hardware. The software tools to create SCA core framework

components for waveforms as well as availability of ORB middleware for each of the processing elements in the system as part of the development platform will be a major value adder to the developers of radios.