

## **Prototyping wireless base stations or edge devices on a DSP/FPGA architecture using high-level tools**

Louis Belanger, LYR Signal Processing, Quebec City, Quebec, Canada

John Ahern, Comlab, Quebec City, Quebec, Canada

Paul Fortier, Electrical and Computer Engineering Department, Laval University, Quebec City, Quebec, Canada

Abstract :

This paper presents the development of a GSM cellular radio General Packet Radio Service (GPRS) wireless application on a combined FPGA/VLIW DSP architecture. A general introduction of the application and its processing requirements are first given. An implementation strategy, based on the use of high-level development tools, is proposed. Finally, implementation results and benchmarks are presented.

==== Abstract =====

With the multiplicity of standards in cellular and wireless telephony, ranging from 'plain' AMPS (analog cellular in the 800 MHz range), GSM (with 3 bands : 900, 1800 and 1900 MHz) to GPRS (General Packet Radio Service), and upcoming 3G (third generation), Bluetooth in the 2,4 GHz range (not to mention the Wireless LAN IEEE 802.1x standards), developers are faced with many challenges. One of these challenges is to develop devices that can operate in multi-standard modes that address not only interoperability aspects, but interference problems as well. To this must be added the soon to be implemented higher bandwidth systems, the increased complexity of protocols and the frantic pace of innovation (recycling of obsolete cellular devices might become big business if the present pace of innovation continues !). On a general basis, with the convergence of telecommunications and network devices, communication developers need to know such wide-ranging disciplines as data communications, RF, DSP, embedded programming, voice and data processing requirements and emerging standards in the next-generation (broadband) communication devices.

To do so, the developers need tools that complement their expertise and address all these difficult interoperability and high performance issues at the simulation and verification level prior to and/or during implementation and testing phases. Design verification can be expanded and synthesised for specific target hardware, using heterogeneous simulation and co-verification approaches. With an ever shortening time-to-market, development, integration and field verification phases have to be minimised as much as possible in order to reduce time consuming redesign steps.

A new signal processing architecture paradigm in the wireless arena is to partition the processing in the following approach: analog front end for RF; FPGA-based for IF translation; DSP-based signal (voice, video, data) for encoding; RISC-based for protocol

processing. Most of this combination is well known and in fact almost a de-facto industry standard (looking at the design of most cellular phones, for example), but a new element in the equation is the FPGA (instead of a fixed-function ASIC). Reconfigurability benefits of FPGA's are quite well accepted but a new paradigm is emerging with the lowering of costs and increase of capacities, making them contenders to replace ASIC's or fixed-function chips. Another interesting aspect in using a low-cost FPGA is the possibility of embedding an organisation's IP in hardware, which is a nice fringe benefit.

There is a design challenge however in exploiting this combined architecture. Some complexity is added, and more specifically, developers might have to master new concepts such as VHDL and some advanced features of simulation tools. Hence appears the need for very good system level tools that harvests the benefits of the combined heterogeneous architecture, compensates one developer's (or its team) weak spots in expertise, and facilitates healthy system level testing prior to fielding.

How this combination of high-level tools and closely coupled heterogeneous platforms can work is shown in a design example: a GSM and GPRS (General Packet Radio Service) system [3-7]. GPRS is part of the interim 2.5 G effort to provide intermediate higher bandwidth capabilities to actual GSM phones, while waiting for the 3G devices and platforms to be introduced in the field. Typically, GPRS (and also competing approaches like EDGE (Enhanced Data rates for Global Evolution, where in fact GPRS will be part of it at the physical layer, but with higher speed modulation ) will bring to GSM portable devices enhanced data rates in the 100 Kbps-1 Mbps range. It should also be noted that GPRS is a packet service in complement to the switched voice channels, that is, GPRS allows simultaneous data and voice connections. A sub-usage of GPRS, which might be its Holy Grail, will be a GPRS only connection for roaming data/Internet-only devices, such as wireless PDA's.

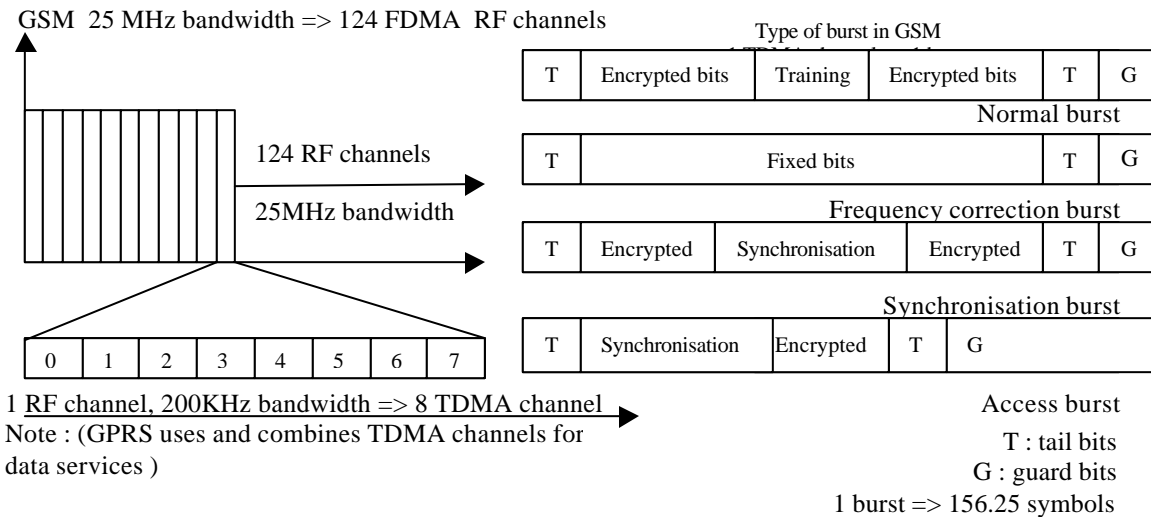


Figure 1: GSM FDM, TDM and 'bursts' structure

In the figure above, a GSM physical channel is presented. There are 124 200 kHz channels that are frequency-multiplexed in a 25 MHz wide RF spectrum, one for each

down-link and up-link paths. The figure also shows in more detail how the ‘bursts’ of each GSM channel is constructed. Basically, each burst is part of an 8 slot TDM frame, forming a 200 kHz wide spectrum. Each one has tail bits and an extended guard interval to avoid interference as long as the MS ( Mobile Station) is within 35 Km of the BS (Base Station ). Some fixed training-bit sequences allows synchronisation between the MS and the BS. It is the BS that commands the MS to advance its transmission time. GPRS is supported by dedicating TDMA channels to data transmission; doing so, these TDMA channels are shared between GSM users within a cell.

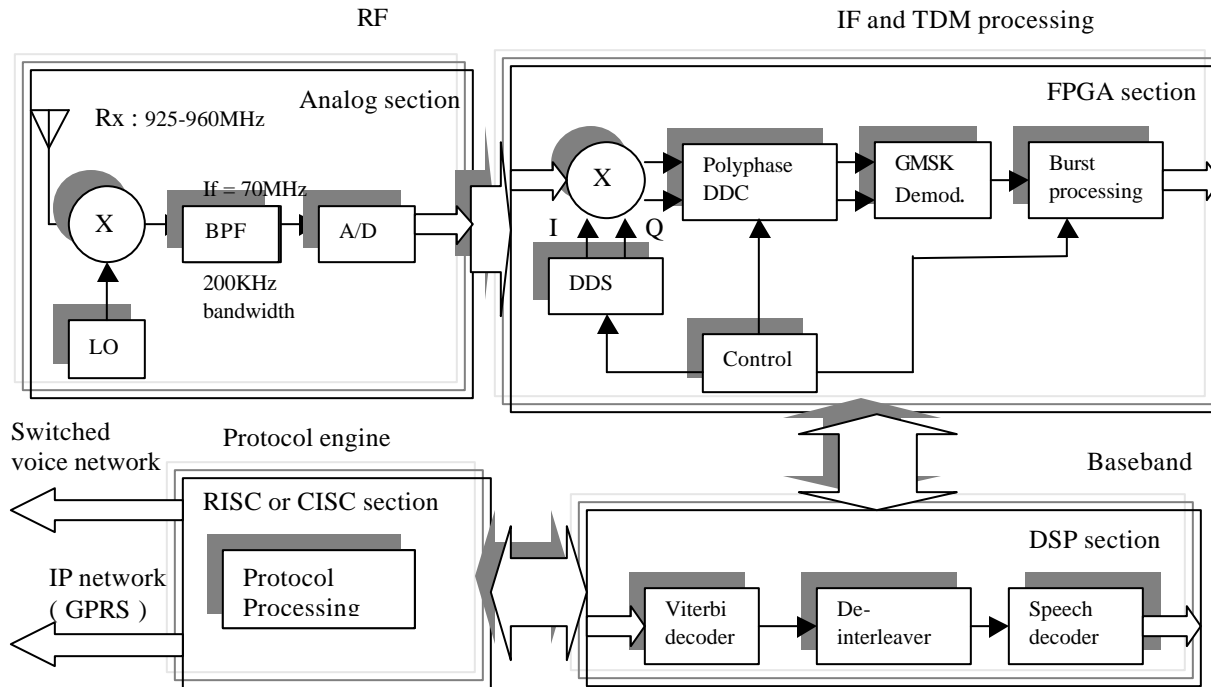


Figure 2: Partitioning of GSM front-end processing between FPGA, DSP and RISC.

Using GPRS and EDGE as a design example is quite pertinent to benchmark the platform architecture and benefits, since it broadly covers many implementation aspects. In the figure above, the main uplink layer 1 elements of a GSM speech and data transmission chain are presented. The figure shows how the processing between different architecture is partitioned. Agile FPGA-based IF translation is needed, and will put to task concepts such as polyphase DDC (Direct Down Converter) and DDS (Direct Digital Synthesis). In addition, FPGA circuitry can be used effectively for burst-processing to implement the very precise timings inherent in the time-slotted TDMA (Time-Division Multiple-Access) architecture of GSM and its frequency hopping and synchronisation schemes. DSP-based baseband processing can tackle the task of encoding, encrypting, interleaving, packetizing and transmitting as well as the reciprocal receiving function. Finally, communication protocol handling is done at the RISC processor level.

The next step is to map this architecture to the system tool level, allowing co-simulation and target co-verification at the application level (including RF channels). There are a great variety of tools on the market. Most of them are in a relatively high price range category: COSSAP from Synopsis, DSP Station, SPW from Cadence and EESof's ADS. There are also more affordable tools such as Matlab/Simulink that can do much of what the higher level tools achieve. Moreover, such a tool, when combined to new VHDL-related tools such as Xilinx System Generator (allowing system level simulation of VHDL/FPGA signal processing functions in Simulink) [1] and Frontier Design's ART Builder (allowing C-to-VHDL code generation, easing C-code-based simulation and VHDL synthesis and verification) provide very advanced capabilities while conserving a familiar interface such as Matlab/Simulink.

### Simulink model

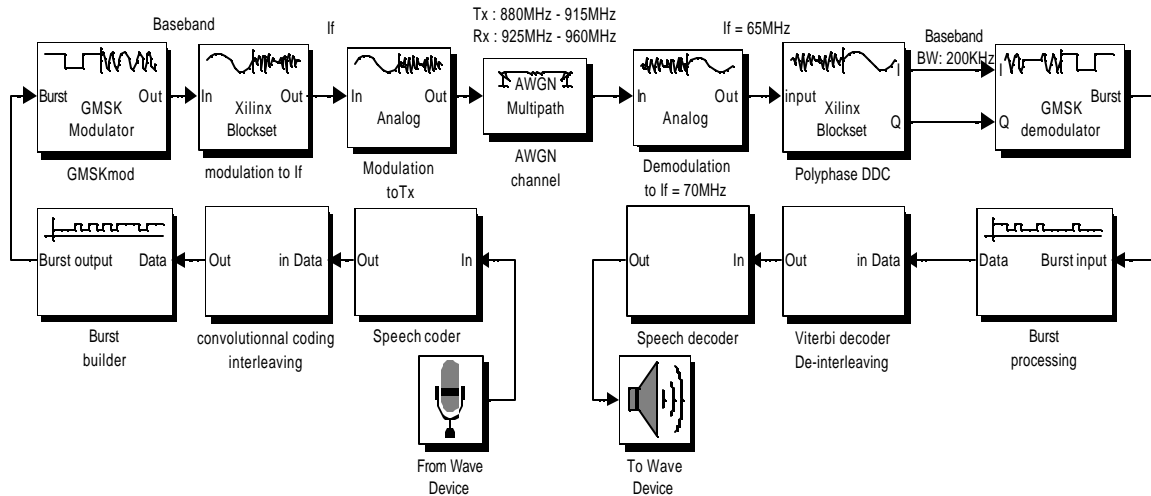


Figure 3: Simplified Simulink Model of GSM Processing

The figure above shows a simplified Simulink Model of the main uplink and downlink GSM layer 1 processing elements. Note that the simplified representation models in detail the baseband and IF processing components. RF modulation is achieved using a simple oscillator. The RF channel is represented using standard Simulink RF channel models.

### Tools mapping

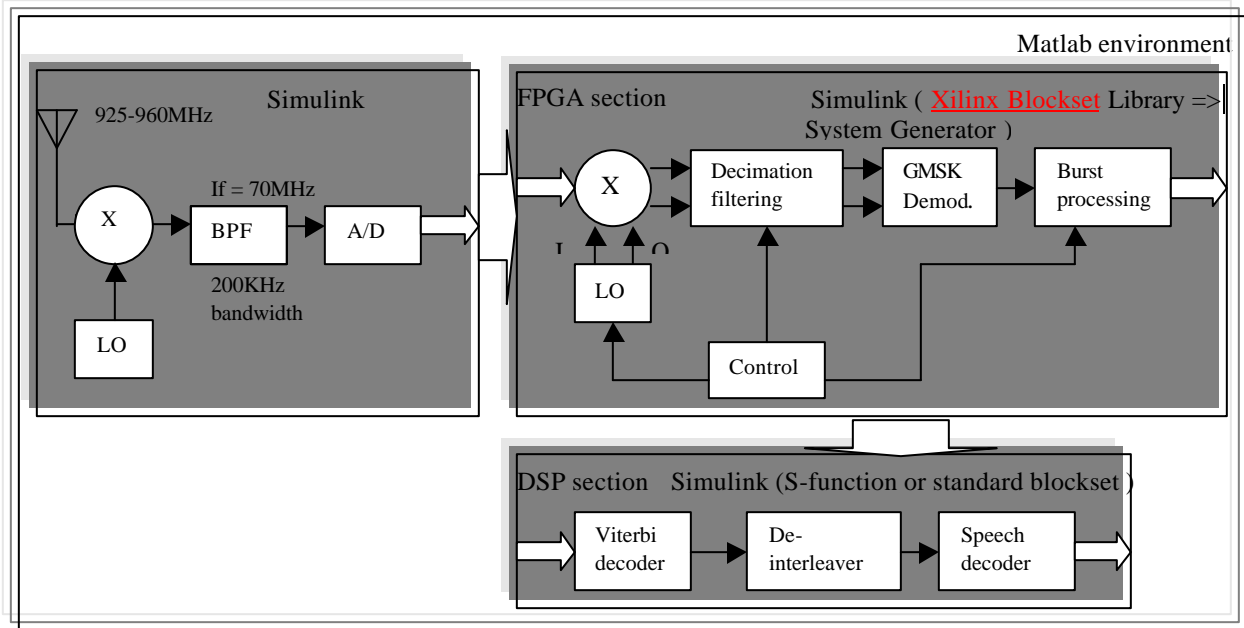


Figure 4: Tools mapping in the Matlab/Simulink environment

The figure above shows in more detail the different tools and approaches which are involved when simulating such a model and implementing part of it on an external target hardware while keeping the overall Simulink environment as the main interface. By default, most of the simulation is initially done using existing Simulink blocks. In some cases, such as the interleaver and block encoder, for example, standard C-code is used. The Burst processor is a simplified version using C-Code, but a more complete VHDL version can be designed using tools such as Mistral [2]. The IF processing part is done using the new System Generator tool from Xilinx. This tool allows the simulation of FPGA-implemented processes within Simulink, and to target a Xilinx FPGA using Xilinx cores.

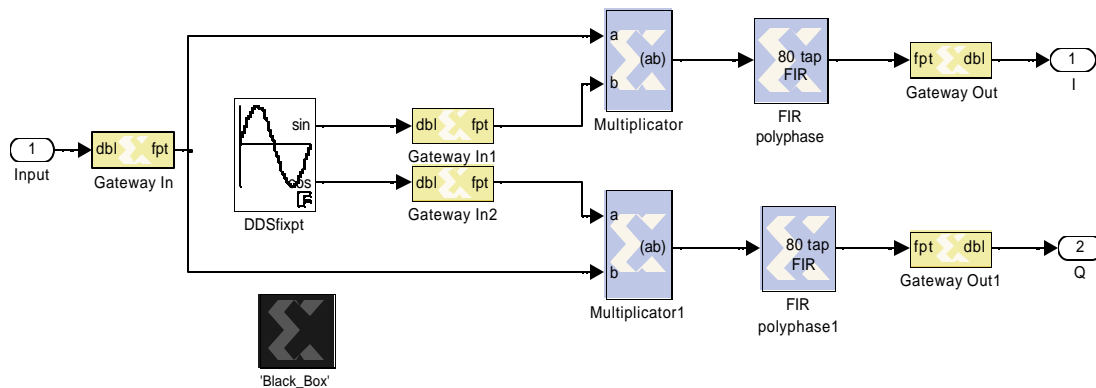


Figure 5: Xilinx Blockset libraries for Polyphase DDC block

In the figure above, use of the [Xilinx Blockset](#) Library to design polyphase DDC for Xilinx FPGA is shown. System Generator uses Xilinx IP Cores for multiplier and FIR polyphase to optimise resources in the Xilinx FPGA. When there is no optimised cores for a specific function, the Xilinx Blockset Black\_Box allows VHDL code to be incorporated in the synthesis of the system. In this model, DDSfixpt is an IQ oscillator designed with fixed point blockset, with VHDL code generated with LSP's Basic VHDL tool.

### Co-simulation and co-verification at the hardware level

In tool terminology, simulation of data communications and signal processing requires different approaches : SDF (Synchronous Data Flow) simulation for continuous signal (e.g. voice, video) processing, DE (Discrete Events) simulation, FSM (Finite State Machine) for protocol states. Traditionally, these approaches are segmented into different models of simulation, and interfacing approaches have been developed to allow parameter passing and code wrapping in the context of validation and verification. These interfaces are referred to as FMI (Foreign MoC (Models of Computation) Interfaces). Once all sub-systems and sub-models have been tested, it is thus possible to move to the system level for simulation and target verification.

At the verification level, more common tools such as standard C compilers and DSP development environments, simulators, etc., can be used to optimise the code generated by the high-level tools. It is important to note, and this is the beauty of the approach, that optimised target implementations can be verified at the system level simulation, thus ensuring that the overall design and its optimised version (with necessary trade-offs) maintains a correct functionality.

The use of high-level tools for target implementation needs a way to implement, optimise and verify block-by-block the target implementation, while keeping the simulation environment as a verifying environment. To do so, the concept of gateway is used. With this approach, part of the simulation can run on the host, and other parts on the target hardware. In the GSM model, C-code interleaving and block coding components, expected to run on the DSP, are 'gateway'ed to a C6x DSP. Doing so, this code can be profiled and eventually optimised up to the point of using pre-optimised libraries, with the advantage of retaining the overall model as a verification tool. Also, VHDL code and Xilinx optimised blocks are simulated for correct FPGA-based implementation.

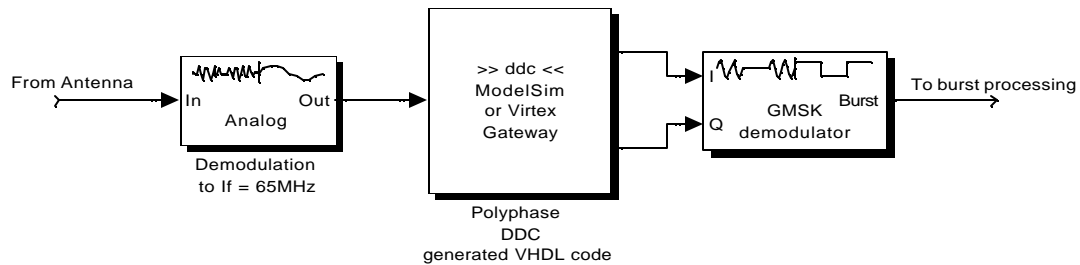


Figure 6: Gateway block to co-simulate or co-verify VHDL code into Simulink simulation

The figure above shows how VHDL code can be co-simulated or co-verified in a standard Simulink simulation with a gateway block. The polyphase DDC model has been replaced by the code generated by System Generator. The validity and effectiveness of the whole model can be verified whether some sections are designed for FPGA or DSP implementation.

In the case of target co-verification, special blocksets are used to allow passage of arguments and timing control information to the physical hardware. FPGA co-verification is done using FMI interface blocksets, allowing injection and read-back of signals, and control of the synchronisation, including single-stepping through the FPGA implementation stages.

#### Some results and benchmarks

Some of the results and benchmarks to compare the high-level approach to the 'manually assembled' building block approach, with emphasis on the benefits and drawbacks of using a high-level approach, are presented. Obviously, the high-level approach provides a fast way to implement a solution, and test it for correct behaviour. However, direct implementation from a non-optimised high-level model is definitely non-efficient. For examples, synthesising the DDS from the floating-point blockset gave some tremendously heavy computational code (the DDS needed many floating-point DSPs). The fixed-point version of the same model which led to direct VHDL synthesis generated large VHDL files. Optimised blocks and FPGA cores had to be used to implement the final result in the FPGA. The benefit, though, of using the high-level approach is the gradual verification of the model and final target.

IP Core Block ( Core Generator )	Number of blocks used	Resources per block (# slices)	Total resources ( # slices)	Total resources (%) Virtex XCV300
DDS (12 bits amplitude, 10 bits LUT)	1	300 slices	300 slices	10 %
Multiplier ( 12 bits * 12 bits)	2	85 slices	170 slices	5.5 %
FIR (12 bits Taps, 24 bits input, 20 Taps)	4	250 slices	1000 slices	30 %
Adder (24bit, carry out)	3	13 slices	39 slices	1.2 %

Figure 7: implementation results in the Virtex FPGA

The figure above presents some of the implementation results in terms of FPGA gate count implementation efficiency of the IF processing. The DA FIR exploits distributed arithmetic, processes 3 input bit per clock cycle, and thus it will take 8 clock cycles per valid output. With the highest speed grade FPGA and 12 bits sample and 24 bits calculation, this gives a processing throughput of up to 80 Msample/sec, taking into account that the DA FIR processes pre-decimated samples with a 4-to-1 decimation ratio. Further optimisation with a fully parallelized implementation can lead to a higher throuput in the 150 MHz range, at the expense of a higher gate count. Also, in this implementation, the filter uses multi-channel capabilities, so we can process 2 channels with some simple additional logic, harnessing the true power of the inherently parallel architecture of FPGA's.

## Conclusion

This paper has presented an approach for using high-level tools to simulate and implement a state-of-the-art wireless application. It also presented additional and more recent tools that eases the development of an optimised FPGA target from these high-level tools. Some benchmarks are also presented, giving indications on the validity and effectiveness of the overall approach.

## References

- [1] Per Holmberg , Anne Mascarin , 'FPGA Reach Into Mainstream DSP Applications' , RTC June 2000.
- [2] Coons, David: 'FPGA Implementation of a GSM Baseband Processor', Proceedings DSP World Fall '99.

[3] GSM 05.01: 'Digital Cellular telecommunications system (Phase 2+); Physical layer on the radio path; General description', ETSI, <http://www.etsi.org/>

[4] GSM 05.02: 'Digital Cellular telecommunications system (Phase 2+); Multiplexing and multiple access on the radio path'.

[5] GSM 05.03: 'Digital Cellular telecommunications system (Phase 2+); Channel coding'.

[6] GSM 05.04: 'Digital Cellular telecommunications system (Phase 2+); Modulation'.

[7] GSM 05.05: 'Digital Cellular telecommunications system (Phase 2+); Radio transmission and reception'.

[8] Chris Dick, Fred Harris , 'FPGA Multirate Filters: A Case Study Using Virtex', 1999 ICSPAT Proceedings.

[9] Chris Dick, Fred Harris, 'Direct Digital Synthesis – Some Options for FPGA Implementation', SPIE Conference on Reconfigurable Technology, Boston Massachusetts, September 1999'.

About the authors :

Louis Bélanger is the general manager of Lyre Technologies and its Lyre Signal Processing division, which has introduced the SignalMaster DSP board line of products. He graduated in 1979 from Laval University's Electrical Engineering Department, and earned his MSEE in 1985.

John Ahern is president of Comlab, an engineering firm specialized in telecom and wireless device development. He obtained his BSEE degree in 1980 followed by an MSEE degree in 1984 from the Electrical Engineering Department of Laval University in Quebec City.

Paul Fortier is a full-time professor at the Electrical and Computer Engineering Department, Laval University, Quebec City, Canada. He graduated in 1982 from Laval University's Electrical Engineering Department, and obtained his MSEE in 1984. He then obtained his PhD from Stanford University in 1989.