Rapid Prototyping for a High Data Rate Wireless Local Loop

Markus Rupp, Eric Beck, Rajeev Krishnamoorthy
Bell-Labs, Lucent Technologies, Wireless Research Laboratory
791 Holmdel-Keyport Rd., Holmdel, NJ 07733-0400,
Fax: (732) 888 7074, Tel: (732) 888 7104
e-mail:rupp@lucent.com

Abstract

The development of a prototype for a fixed wireless loop system is reported. The TDMA based communication system allows flexible modulation schemes between QPSK and 64QAM in order to allow for dynamic bandwidth allocation. It is estimated that the overall complexity of the prototype is of the order of 10-100GOps. Since such a large complexity cannot be provided by existing DSPs, our prototype consists of a fast DSP, accompanied by several Altera Flex10K FPGAs. Complexity intensive algorithms are first evaluated on the DSP then moved onto the FPGAs in order to lower the load on the DSP. Such a structure allows very flexible reprogramming in C of initial algorithmic ideas and can be used to compare performance as well as validate the implemented algorithms.

1 Introduction

Wireless solutions offer more flexible connections than the now existing telephone cables. Higher demand on bandwidth in particular for short bursts in packet mode transmissions require complicated Medium Access Controller (MAC) functions to efficiently allocate the required bandwidth. Here wireless solutions offer many new ways to provide a large and flexible bandwidth for many users while at the same time consuming only moderate bandwidth for the whole communication system. The Fixed Wireless Loop (FWL) project at the Bell-Labs Wireless Research Lab yields to the development of such a system with high flexibility by applying steered beam antenna systems together with adaptive modulations schemes. A TDMA based concept allows changing the modulation scheme on a slot basis and thus transmitting higher bandwidth on demand. The system is designed to carry 64 slots in a 2.5ms frame with 156 data symbols in each slot. A beacon signal at the beginning of each frame allows for channel allocation as well as frame timing. Adaptive modulation from QPSK up to 64QAM allows data rates from 64Kbs to 1.544Mbs per user. Trellis coded modulation ensures such a high data throughput.

This paper reports the experience gained by building a prototype for such a system; in particular the modem functions. Since many transmission characteristics over wireless channels are not well understood, the development of such a prototype is considered crucial before seriously deploying wireless communication systems. The prototype aims to allow running real-time experiments for all modem functions, i.e., matched filter, symbol-timing recovery, frequency offset compensation, equalization and coding, to name the most important ones. The prototype is expected to provide a test-bed for many different algorithms, pre-selected by means of intensive simulations. Thus, the prototype is required to offer a lot of flexibility in reprogramming as well as high complexity to try even exotic functions with unusual complexity.

2 Complexity Considerations

Estimating complexity is a difficult task since the complexity count depends strongly on the hardware. For example, some DSPs require additional operations for shifting while others provide this operation in parallel during the same cycle than an addition or multiplication is performed. Thus, a general complexity count can only be a rough estimate aiming to the correct magnitude of operation numbers. For the FWL project, the transmitter is considered the easier task. Essentially a pulse-shaping filter is required with some additional hardware for packing, adding CRC and coding. A hundred operations were estimated for this operation for each symbol leading to $100 \times 400 \times 156 \times 64 \approx 400 MOps$.

The receiver on the other hand is a much more complex device and the complexity depends very much on the chosen algorithms. Beacon timing recovery, demodulation and frequency offset estimation lies in the order of a complex pulse-shaping filter thus 15,000Ops per slot. The data slots however are more complex. An additional (fine-tuning) symbol timing recovery requires about 6,000Ops, a small amount compared to the fractionally spaced equalizer that requires about 20,000Ops. The Viterbi algorithm on the other hand can exceed 20,000Ops depending on the modulation scheme used. Thus, approximately 76,000Ops per slot

0-7803-5700-0/99/$10.00©1999 IEEE

will be required for the receiver adding up to 2GOps for all 64 slots.

Thus, transmitter and receiver add up to approximately 2.4GOps. Depending on the selected hardware a factor of 10 may need to be applied leading to a final estimate of 25GOps.

Running the functions and protocols of the MAC in DSP processors is considered inefficient since it relies on many bit oriented operations and conditions. Such operations are not well suited for DSPs in general. It was therefore decided to run the MAC function on a Pentium class processor for the first phase of the prototype. The computational power of this processor was deemed sufficient to support the MAC layer functions written in ANSI C and operating under a VXWorks real time operating system.

3 Hardware Solution

Such a large amount of complexity cannot be covered by standard DSP solutions. A ©TMS320C6201 (e.g. C62) from Texas Instruments is capable of 1.6GOps under ideal situations. Programming typical algorithms showed results between 200-400MOps. This would require up to one hundred of such processors in order to deal with the required complexity. On the other hand, such a DSP is fully programmable in ANSI C, allowing a very high flexibility and even DSP programmers unskilled in the art of Assembler programming can quickly learn how to program a C62 efficiently.

Complexity is not the only concern when designing a prototype. Another very important aspect is the bandwidth of the data streams. Assuming a T/4 sampling rate and 10 bits precision, base-band signal requires $2 \times 10 \times 4 \times 4MSps = 320Mbps$. In practice the serial port of the C62 is capable of running up to 70Mbps, and since two of them are available, 140Mbps is possible at maximum speed, still not even half the bandwidth required. At the MAC/Modem interface a data-stream of about 40bytes per slot is expected leading to 8Mbps, a value that can be dealt with by the host port of the C62. However, the data rate alone is not the only important factor. Since the standard PC does not allow DMA over the PCI bus, the processor is active during data transmission and cannot be used for other operations. Thus, a rapid data transmission frees the Pentium from unnecessary wait-cycles.

It was decided to use a combination of FPGAs (Flex 10K-family from Altera) together with a C62 DSP, a concept that has been used sucessfully in the past[1]. Standard operations like the pulse shaping filter were implemented immediately on the FPGA since they require a large amount of complexity and are fairly straightforward to implement. The concept is that complex functions with uncertainties are first developed on the C62 DSP in order to run real-time experiments with them and decide final parameter values. Those algorithms that are cleared from investigation are then migrated from the DSP world into the FPGA world, reducing the computational load of the DSP.

4 Design Flow

Starting with simulations on a high level language to programming a DSP or FPGA chip, there are many possibilities for the design flow once a flexible hardware test-bed is provided. Our simulations were programmed with ©COSSAP, a communication tool set from ©SYNOSYS. ©COSSAP (like ©SPW from CADENCE) provides a rich library of DSP functions that can be connected on a graphical basis. This allows a fast and flexible design together with an easy-to-read description so that other members of the development team can quickly understand all the modem functionality. Functions that are not provided by the existing libraries can be added in ANSI C.

However, the tool's capabilities do not end at the simulation level; so-called bit-true blocks can be used to describe integer function with exact mantissa lengths of their internal variables and operations. These blocks can be combined with existing C-code to run evaluations as well as allow for automatic ©VHDL (or ©Verilog) generation.

Since the DSP from TI can completely be programmed in ANSI C, its procedures could also be developed under the ©COSSAP environment, co-simulating floating-point code with fixed-point code in TI's C. This allowed a quick evaluation and verification of the DSP code. The identical code can now run in real-time on the C62. Such a design step thus allows for rapid development without transition bugs. However, such a description usually does not include interface parts as they occur when hardware blocks are connected.

On the other hand, the bit-true blocks also provide the possibility of automatically generating ©VHDL code for chip design. The matched filters were designed with these tools and the so obtained ©VHDL code synthesized with the ©PSYS FPGA compiler. Comparisons with direct-written ©VHDL designs did not

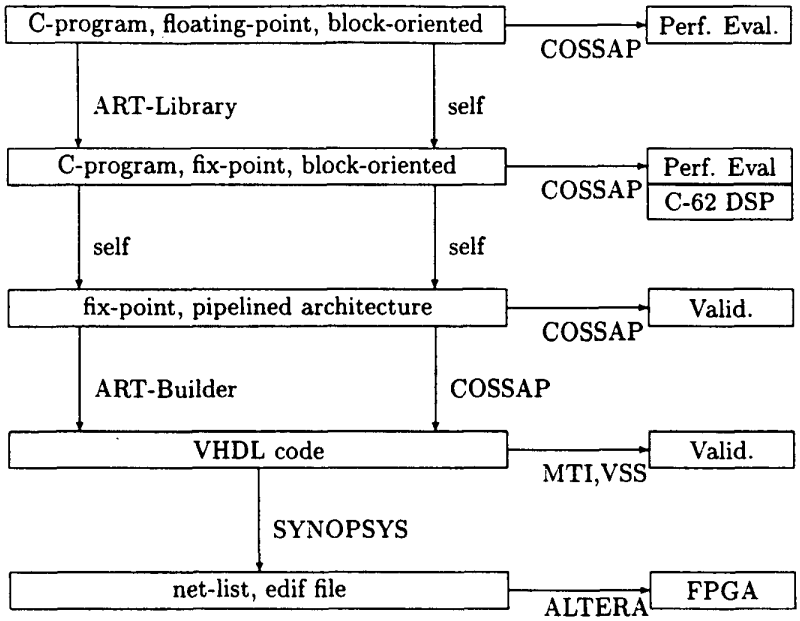

Figure 1: *Flexible Design Flow.*

show any remarkable differences in the results. However, other, more complex functions that require additional control structures are not easily described in the existing bit-true blocks. Here writing ©VHDL by hand is much more efficient and flexible.

Software tools from Frontierd corporation offer an interesting alternative. The ©*ART-Library* allows refining an ANSI C-program into fixed-point arithmetic. To this end, a class of fixed-point data types and variables is defined. A statistic tool set allows for finding the required mantissa length of each variable. A second tool, the ©*ART-Builder*, maps the C-code into ©VHDL. The so obtained C-code is quite readable. The tools allow a C-programmer to design hardware architecture on a C-level. In particular, designing a multi-rate system requires to define a (simple) finite state machine in C. The complete transmitter functions for the FWL system have successfully been programmed with these tools.

The final step for the FPGA design is to re-compile the edif-files obtained by the design compiler. This recompilation is done by Altera's ©MaxPlusII tool in order to obtain a place-and-route information for the selected FPGA chip. Now the exact logical cell amount and timing information is obtained while the estimated numbers, given by ©SYNOPSYS' design compiler, are often far of the true values.

5 Implementation Details

Sections 2 and 3 established the algorithmic complexity for the fixed wireless loop system modem, and the need to support both hardware (FPGA) and firmware (DSP) algorithms was shown. In order to realize a real time system from these algorithmic components it is necessary to provide efficient interfaces with the appropriate timing and synchronization. As fast as it is, the C62 processor core is ill suited to handle events that occur at the sample rate (16 Ms/s) or even the symbol rate (4 Ms/s). Fortunately the C62 contains peripheral units (e.g. the multi-channel serial port and DMA controller) that relieve the core processor from data transfer tasks. As such the computation can be organized so that the core processor handles events at the more reasonable TDMA slot rate (25 KHz). Organized in this manner, the prototype for this system derives the timing signals (e.g. bit, sample, slot, and frame) in hardware with the C62 slaving to these timing signals. The bit timing is passed to the C62 via the serial port controller, while the slot and frame timing is conveyed using the processor's external interrupts.

The coupling of these hardware events to the pro-

cessor firmware is illustrated in Figure 2. As shown two receive serial ports (MCBSP) are dedicated to the receive data path, and one transmit serial port to the transmit data path. Three of the five available DMA units are dedicated to servicing these serial ports. Note that the C62 DMA units are auto-initializing: the parameters for the next slot transfers are established during the current slot interval. Additionally, each DMA unit has a pool of sample buffers available to it. The status of each buffer in the pool is recorded in the buffer maintenance tables. During each slot interval the external interrupt service routine updates the DMA auto-initialization registers along with the buffer maintenance tables. In this manner a consistent flow of sample data is maintained between the DSP and the gate arrays irrespective of the processor load. Furthermore, certain (transmit) synchronization slots and idle transmit slots consist of deterministic data patterns. The interrupt service routine identifies which slots are eligible for these deterministic data patterns and automatically substitutes them, relieving the DSP core from unnecessary computations. It is noted that the overhead of this interrupt service routine is about five percent of the processor bandwidth.

The remaining processor bandwidth is dedicated to scheduling which slots require computation and actually executing the modem algorithms. The scheduling algorithm scans the modulator and demodulator slot tables for active slots. The modem algorithms/buffers are denoted as MOD and DMD for modulator and demodulator, respectively, in the figure. These slot tables are pre-established by the MAC layer to denote which slots of the frame are currently active. Groups of active slots establish virtual channels of variable bandwidth, and each virtual channel is associated with a data buffer in the DSP memory space. Again the C62 architecture facilitates the MAC interface as the host port interface (HPI) and its associated auxiliary DMA channel maps the DSP memory into the address space of the host processor. For inactive slots, any sample buffers are quickly freed and returned to the free buffer pool, whereas for active slots the corresponding sample buffer and data buffer pointers are passed to the modem signal processing algorithms. While not currently used, the final DMA unit is available to transfer virtual channel data between the MAC data buffers and a temporary (internal memory) data buffer. This avoids degrading the C62 core processor performance when the data buffers are located in external memory.

The physical embodiment of this prototyping system is the Texas Instruments EVM card. This standard PCI card includes a C62 processor, a PCI interface, significant external memory, and a mezzanine card interface. A custom mezzanine card contains the Altera FPGAs, digital to analog conversion, and the intermediate frequency electronics for the radio transmitter and receiver.

Note the flexibility of this architecture for different firmware/hardware configurations. A typical test configuration places the pulse shaping functions in the gate arrays, with the remaining modem functions residing in firmware. In this case the serial interfaces are operating at their maximum data rate. Alternative configurations place more of the modem functionality in the gate arrays. These configurations require much less bandwidth between the C62 and the gate arrays, and the DSP programs are easily modified to redefine the bits as required. The MAC interface remains unchanged.

6 Conclusion

We have presented a complete hardware and software prototyping methodology for a high data rate modem. The important feature of this prototyping method is its flexibility: modules can be written in ANSI C and tested in simulations, then ported to a fast DSP and finally to a chip level description. When implemented with fixed point C tools such as those offered by Frontier Corporation, the complete design is accomplished as a C language description, reducing the need for specialized hardware languages such as ©VHDL or ©Verilog. This methodology allows researchers to implement their algorithmic ideas in hardware with real-time capabilities and considerably speeds up the concept-to-prototype turn around time.

References

[1] G.E.Prescott,S.Tyler, "Prototyping of military radio systems using filed programmable gate arrays and DSP microprocessors," Proc. ICSPAT 1997.

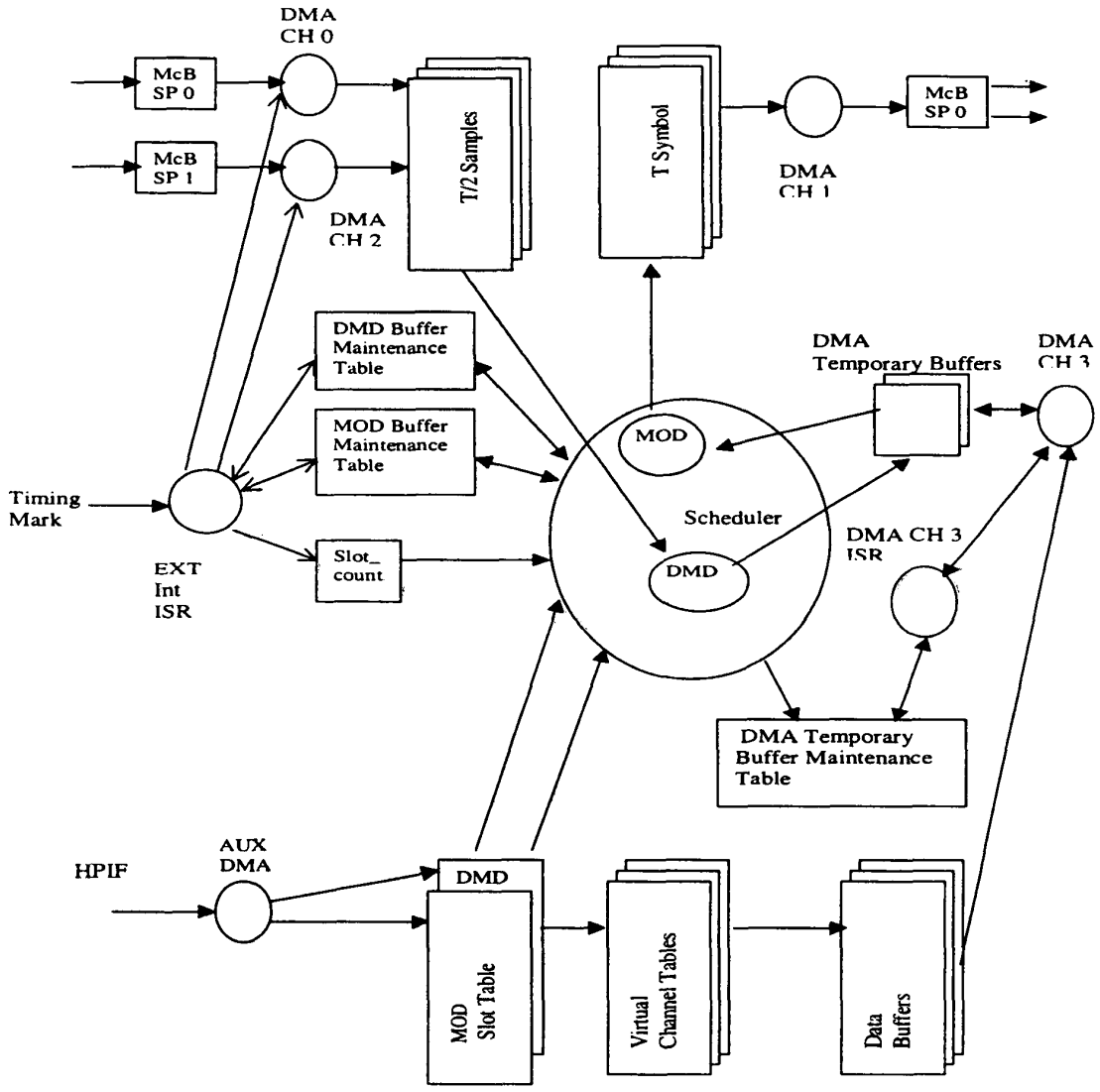

Figure 2: *The scheduling in the DSP chip.*