A Multiprocessor Framework for Rapid-Prototyping and Evaluation of Soft Transceivers

Muhammad Salman Asif[1], Mohammad Omer[2], Dr. Amjad Luna[3], Dr. Noor. M. Sheikh[4]

Al-Khwarizmi Institute of Computer Sciences, UET Lahore, [1,4]Department of Electrical Engineering, UET Lahore, [3]Department of Computer Science, LUMS

E-mails: salman@uet.edu.pk, omer@kics.edu.pk, amjad@lums.edu.pk, deanee@uet.edu.pk

***Abstract* — Recent years have seen rapid evolution in the architectures being explored for realizing high-speed software-defined radios. There is, however, a distinct need for a low-cost programmable platform where algorithms for base-band transceivers can be rapidly prototyped and tested with real-world data, streaming in from diverse sources of telecommunication traffic. This paper explores an analytical method for laying out such a generic platform. It investigates the constraints involved in realizing such a platform, and the minimum functionality needed within the solution so as to provide adequate scalability to allow the implementation of a wide variety of communication algorithms. The paper concludes with a case study of a multi-channel communication system that has been successfully implemented on the proposed platform, highlighting the performance benchmarks it had to meet in order to prove suitable for the task of communication system evaluation.**

***Index Terms* — MIPS, DSP, Analog Front End (AFE), Buffers, Data Converters, PAM (pulse amplitude modulation)**

I. Introduction

With the low-cost availability of increasingly higher performance programmable devices, the choice of an implementation platform for communication systems is gradually shifting towards DSPs and FPGAs. This trend is also being fuelled by the crucial need to accommodate, on a single device, a host of competing, divergent standards that mark the evolutionary path along 3G and beyond. Soft transceiver architectures have, therefore, evolved significantly over the past few years. However, almost all of the activity in this area is exclusively focused on applications related to 3rd and 4th generation cellular systems. There is a sizable segment among the rest of the communication applications that is not as demanding as the 3G/4G systems. Such applications include TDMA satellite modems, mobile and portable radios, telemetry systems, remote data acquisition, and so on. Employing 3G/4G architectures for such applications would certainly be an over-kill.

When a soft transceiver makes a leap from its simulation environment to the real world, several pressing issues come into play immediately. These include finite word-length effects and interfacing with the real-time data sources and sinks. The former can be addressed through the wide array of configurable and programmable logic available to the system designer. There are, however, two main issues associated with the latter: the type of interface that will feed data to the transceiver, and the analog interface that will carry the up sampled signal from the transceiver onto a real channel. There is, thus, a distinct need for low-cost generic programmable platforms where algorithms for base-band processing of software radios could be cost-effectively prototyped and tested in an end-to-end system environment.

The rapid prototyping architecture proposed in the rest of this paper attempts to address these problems. The proposed system lends itself easily to rapid prototyping of a large class of contemporary transceivers. We have realized the proposed system and have used it for evaluating several communication systems, the results of which find a mention at the end of this work

II. System Components

In order to identify the requirements for prototyping system, we first need to look into the internal details of what forms a base-band transceiver.

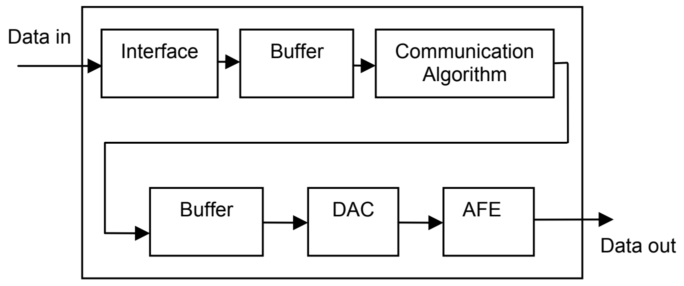

Fig. 1. A Generic System Layout

The Fig. 1 shows the skeleton transceiver that can theoretically implement all communication methods at the base-band level. All that the system needs is a digital interface from which the data pours into the system. The interface data needs to be buffered before being relayed into the main computational engine. It is not advisable to waste the execution bandwidth of the engine in fetching data sample by sample [1][2]. A cushion is therefore provided in the design, where the data can be buffered before the processor fetches the same in a long haul. It can be seen that the buffers encompass the processor at both ends. The system finally terminates in a DAC which is the last line in the digital chain and the analog data originates thereafter. Analog front end is able to provide the needed filtering and differential drive wherever required [3][10].

III. Communication System Considerations

In a typical communication system, the receiver chain is made up of a number of typical building blocks with a few features differing here and there. In order to narrow down our focus, we begin by looking at a common receiver architecture which employs pulse amplitude modulation (PAM). A base-band PAM signal would generally be represented by

$$s(t) = \sum_{n=-\infty}^{\infty} A_n g(t - nT_s) \tag{1}$$

Where A_n denotes the set of M possible amplitudes corresponding to $M=2^k$ possible k-bit blocks of symbols. The waveform *s(t)* is a real-valued signal pulse whose shape influences the spectrum of the transmitted signal, and T_s is the symbol interval.

In a PAM architecture, the modulation is simple and does not involve a complicated data processing chain. A typical M-ary PAM modulator with an uncoded data-rate of R bps takes up the input bit-stream, breaks it into k-bit blocks and then forms $M=2^k$ PAM symbols. After the formation of PAM symbols, what is left is to use interpolation to over sample the PAM symbols and use transmit filtering to shape the symbols in order to limit the bandwidth of the resulting signal [4]. The pulse shaping is most commonly done through filtering the PAM symbols with a raised cosine filter of the form

$$H_e(f) = \left\{ \begin{matrix} 1 \\ \frac{1}{2}\left\{1 + \cos\left[\frac{\pi(|f| - f_1)}{2f_\Delta}\right]\right\} \\ 0 \end{matrix} \right\}$$

$$\text{for} \quad \begin{matrix} |f| < f_1 \\ f_1 < |f| < B \\ |f| > B \end{matrix} \tag{2}$$

where B is the absolute bandwidth,

$f_1 = f_o - f_\Delta$

$f_\Delta = B - f_o$,

f_o is the *6-dB* bandwidth of the filter.

The rolloff factor is defined by $r = f_\Delta / f_o$

It can be seen that the pulse shaping filter converts the ideal pulses to finite bandwidth pulses that can easily negotiate with a band limited channel and fulfill Nyquist criterion for ISI prevention.

The receiver structure is much more complex; not only in terms of the nature of signal processing involved but more importantly because many different receiver architectures may be used depending on several factors that include; needed data throughput, level of reliability, and the nature of the application. We will however employ a commonly used receiver architecture that provides all the necessary functionality required of a typical non-fading wireless link. If a PAM modulation scheme has been employed, the most important task left for the receiver is the carrier phase recovery and the baud loop timing recovery [5]. If these are accomplished successfully in the presence of additive white Gaussian noise, the receiver can then employ a wide array of signal processing and coding techniques to recover the signal reliably, thus covering the wide spectrum of possibilities mentioned above.

IV. Hardware Map Of The Transceiver

Having discussed both the algorithms and the system hardware, we are now ready to look at the dynamics of the system in detail. Before going to look at how we may be able to develop architecture, it would certainly help if we can list the desirable features of our layout. A look at the transceiver requirements will bring out the following facts:

1. Provision for data-link layer processing.
2. Flexibility to allow changes in the main communication algorithms.
3. A flexible back-end with an interface structure that can allow data to pour in through a wide variety of sources and interfaces.
4. Configurable glue that can take care of the inter-processor communication.
5. Appropriate data buffering so as to conserve I/O bandwidth of the main computational engine.
6. Configurable clocking strategy in order to drive various processing engines as well as determine the data flow within processor.
7. A suitable analog front-end that has configurable bandwidth in order to allow a large class of signals through it.

In the light of the requirements that have been spelled above we propose a generic transceiver architecture that fulfills them.

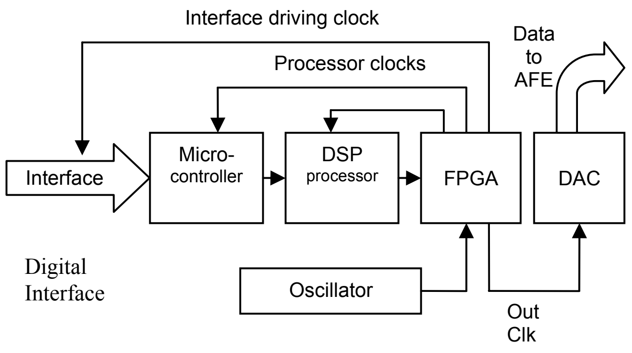

Fig. 2. Component Layout and Connectivity

The Fig. 2 shows that the scheme is a three tier approach to the system design which closely follows the architecture mentioned in the beginning. In effect, the processing horsepower resides within the DSP CPU, and the buffering operations have been accomplished by enclosing the processor within a microcontroller and an FPGA. Several virtues of the design however need a close mention. They are listed below:

1. A high end microcontroller like the ones carrying an ARM core are most suited towards the front-end data processing as they have a rich set of peripherals that can be used to tap data in from various standard interfaces.
2. FPGA structure is a convenient method for housing a FIFO type buffer because it always comes equipped with memory banks [6].
3. Driving all clocking interfaces through a single device of FPGA ensures that the system timing is maintained despite clock jitters and oscillator wanderings.

It is apparent that the proposed architecture is capable of providing the true functionality of a transceiver with an onboard solution. Such a transceiver is capable of adapting to various data rates and can allow several communication systems to be successfully implemented. It must be mentioned that a typical FPGA has the ability to take several different clocks at the input and is able to divide them up with a wide range of integer and fractional factors so that a number of high slew clocks can be delivered at general- and special- purpose I/O pins of the device. Such latitude of clock provision through the use of delay locked loops inside the FPGA is instrumental in ensuring that the transceiver can provide clocks for driving the system at arbitrary rates. Through the use of this allowance, we can drive our data converters at almost any rate dictated by differing system requirements.

The use of a programmable digital signal processor within the solution provides for the ease of communication system implementation [11]. A powerful DSP would form the heart of the system yet does not interface directly to the external environment on any of its ends. At one end, it captures data from a microcontroller which is fetching the same from real world traffic. At the other end it sinks its data into a FIFO buffer which relays it sequentially to data converters. This structure has been crafted to minimize the I/O bandwidth required for the DSP. The DSP is now free to take up the computational load of a complete communication system. The communication algorithms coupled with their implementation strategies will now determine the likelihood of successful prototyping. We therefore give an architectural highlight of these in the next section.

V. Receiver Architecture

A communication receiver would severely encumber the implementation platform due to the complexity that accompanies it. Many of the receiver architectures however, share a common set of tasks that need to be successfully accomplished.

V-a. A Typical QAM Receiver

We look at a typical QAM receiver before delving into the choice of hardware platform that it will dictate;

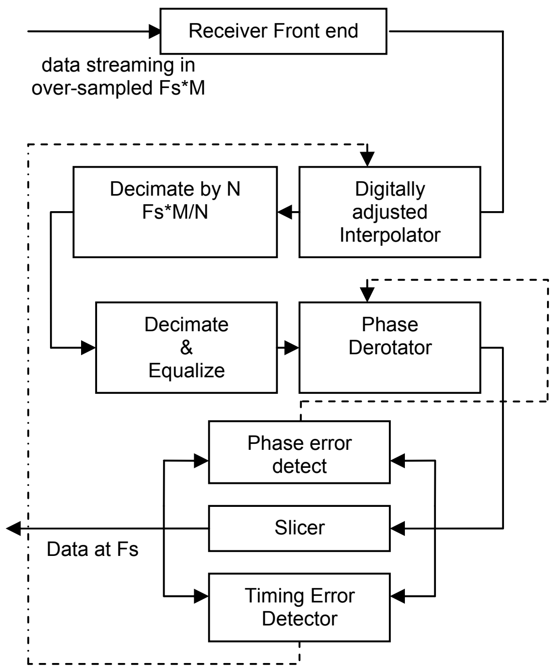

Fig. 3. Receiver Model

The receiver in the Fig. 3 employs the simplest possible architecture for data recovery. In order to keep the discussion simple, many important symbolic notations have been omitted like that of real and complex data paths. For the brevity of discussion the diagram has also omitted the modes (decision directed and training) in which the receiver will be called upon to operate. The front end of the receiver is sampling the data at a rate faster than the symbol rate. This of course is essential for the proper working of timing recovery loop of the receiver [6]. But the rate of over sampling will be dictated by the type of algorithm being employed for the detection of timing and phase errors. Generally, the sampling frequency F_{samp} is four times that of symbol frequency F_{sym}

V-b. A Generic Communication Receiver

A standard QAM receiver implementation has been detailed above. It can be easily inferred that several other receiver structures like those of QPSK, PSK, PAM, and all the M-ary versions of the above will follow the same scheme as laid out in the receiver model. As the signal constellation gets more and more dense, the change only occurs in the structure of the decision device (slicer). A slicer used for a four point QAM will have a simple single threshold decision structure, while a multipoint QAM receiver will have a greater number of slicer thresholds to compare the received signal against. Principally, the same underlying receiver architecture can suffice for all these modulation schemes. In the instance of high density constellation schemes receiver will be more sensitive to phase and timing errors. Apart from the need to have more robust timing and phase recovery algorithms in place, the receiver for a large class of modulation schemes will remain essentially unchanged [8]. We, therefore, stress the notion that evaluation and suitable performance of the receiver for a given QAM scheme will automatically render it suitable for the prototyping of a large class of communication receivers.

In the light of this architecture we can observe that a receiver needs to be implemented on a platform that allows the various filtering operations to happen quickly enough in real time simultaneously, and can provide for the complex feedback structure of the receiver. We have selected a TMS320C5510 for the task of implementing this engine. The DSP is customized for intensive signal processing operations typically found in communication systems and can provide a powerful mix of standard sum of products implementation plus a very flexible decision structure to implement feedback paths.

VI. LOAD CALCULATION

Texas Instrument's DSP has been employed as the major engine for implementing the communication algorithms. TMS320C5510 is a fixed point processor loaded with a dual MAC unit for intensive DSP applications [9]. The processor has a fragmented memory with six simultaneous access busses for quick data access. The implementation of several different receiver architectures has proven the computational superiority of the engine. Following is a brief summary of the results obtained so far with the engine at a clock frequency of 200 MHz. The table lists down operation on a chunk of data which contains 32 bits of digital data.

Algorithm	**Process Time in CPU Cycles**
4 point QAM receiver with hard decision slicer	13000
4 point QAM transmitter with a transmit filter of 41 Taps	3600
8 PSK trellis coded decoder with hard decision trace back	8000
8 PSK trellis coded encoder	200

Table 1   Performance Sheet

A simple calculation will then lead us to explore the DSP data rates which can be successful prototyped. Suppose that the digital data stream has been generated by a standard 64kbps channel. The transceiver computation time would be the sum of transmitter and receiver execution plus some looping overhead. For the present we neglect the overhead involved. Therefore the ability of DSP to handle the same would be,

Traffic generation time for 32 bits = 1/64000 *32 sec
DSP computation time = $1/200\text{x}10^6$ * 16600 *(sum of transmit and receive clock cycles without error coding)*

For successful real time operation
DSP computation time > Traffic generation time

If we repeat the above calculations for a varying number of data rates, it would be observed that our DSP is well capable of handling 4/8 voice channels of standard 64 Kbps with/without channel coding. However, the figures quoted are not the most optimized ones and further work is continuing in order to bring them to the level of being benchmarked with TI's standard implementations.

VII. HARNESS DESIGN

The harness of the system induces a point where successful design decisions will distribute the computational load on each individual component as well as minimize the problems relating to data flow hierarchy. We propose a mixed serial/parallel architecture where a DSP communicates with controller on a serial link and communicates with the FPGA on a parallel highway. The figure below illustrates the interfaces deployed

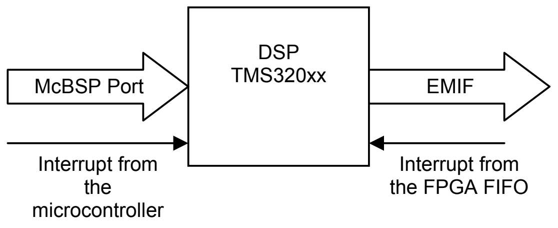
DSP
TMS320xx
McBSP Port
EMIF
Interrupt from the microcontroller
Interrupt from the FPGA FIFO

Fig. 4. Interface Specifics

As can be seen in the Fig. 4, the DSP is interfaced through a Multi channel Buffered Serial Port (McBSP) to the microcontroller interface and through an external memory interface to the FIFO residing within the FPGA [6][13][14]. The types of interfaces employed are design decisions which give us the following key advantages.

1. The raw data traffic being generated and modified by the link-layer exists in a highly packed version of raw bits, all of which has to undergo formatting and base-band processing. The magnitude of the data dictates that a high speed serial link will adequately serve the purpose
2. The data being spewed out of the DSP is a modulated one and has to go through a data converter. The data therefore exists as digital samples of an analogue waveform which will be generated upon DAC interpolation. Each sample therefore resides with a certain number of precision bits as a separate entity. The sheer volume of the data and its character dictates that the same should be relayed over a parallel channel configured to the width of each data sample

We, therefore, can have our interface judgments supported by the achievable data rates of both the interfaces. McBSP can typically handle a high rate E1 stream as well as higher multiplexed versions of the same. EMIF on the other hand can transmit as fast as the peripheral clock which has a maximum operation frequency of half the processor clock.

VIII. Conclusion

We have demonstrated the need for a rapid prototyping platform for digital transceivers and discussed the skeleton of a generic architecture. The generic architecture was then fine-tuned to absorb requirements posed by different system level constraints. We finally emerged with a home-brewn multiprocessor framework that has been implemented and has demonstrated successful prototyping realizations. We then laid out the performance of standard receiver implementations on the proposed platform and compared their feasibility by assuming a standard 64 kbps traffic generator. The paper has presented an integration scheme for successfully weaving all processors into a synergetic framework geared towards communication applications

Acknowledgement

We thank *Texas Instruments* for providing evaluation kits for DSP benchmarking as well as *ATMEL* limited for supporting the work with the grant of ARM 7 RISC machine and the evaluation boards.

References

[1] Kopetz and Hermann, "Real-Time systems, design principles for distributed embedded applications" *Springer Verlag publishing*, volume 395, issue, pp. 35-36, 2002.

[2] Jefferey H. Reed, *Software Radio: A Modern Approach to Radio Engineering.* Prentice Hall International, 2002

[3] Analog Devices technical staff, *The ADSP 21xxx applications handbook*, ADI Inc, 1998.

[4] Leon W Couch, *Digital and Analog Communication Systems*, Prentice Hall International, 2001.

[5] Edward Lee and David Messerchmit, *Digital Communications*, Kluwer Academic Press, 2000 pp 250-255.

[6] XILINX Inc, *SPARTAN ii Device Data sheet*, XLINX Inc, 2002.

[7] Dimitrios Efstathiou, Jose Fridman, and Zoran Zvonar, "Recent Developments in Enabling Technologies for Software Defined Radio" *IEEE Communications Magazine*, August 1999

[8] Proakis, John G., *Digital Communications*, 4rd Ed., McGraw-Hill 2001.

[9] Texas Instruments Inc, *C55xx programmers' reference,* TI 2003.

[10] Joseph Mitola, III, "Software Radio Architecture: A Mathematical Perspective" *IEEE Journal on Selected Areas in Communications,* vol. 17, no. 4, pp. 514-538, April 1999.

[11] Kondo, Matsuo and Suzuki. "A Software Defined Architecture Concept for Telecommunication Information Systems" *ICC94,* (NY: IEEE Press, 1994)

[12] Chris Dick, Fred Harris and Michael Rice. "FPGA Implementation of Carrier Synchronization for QAM Receiver" *Kluwer Publisher - Journal of VLSI Signal Processing* 36, 57-71, 2004.

[13] S. Srikanteswara, J. H. Reed, P. Anthanas, and R. Boyle, "A Software Radio Architecture for Reconfigurable Platform," *IEEE Communications Magazine*, vol. 38, no. 2, pp.140-147, February 2000.

[14] J. Mitola, "The Software Radio Architecture", *IEEE Communications Magazine*, vol. 33, no. 5, pp. 26-38, May 1995