Simulating Real-World Signal Environments for Receiver Testing

Howard Hilton, Hewlett-Packard, Lake Stevens Division
Everett, Washington

***Abstract*—One of the challenges in designing a digital communications receiver is verification of performance with "real world" signals. The generation of test signals which can be calibrated and customized often requires the development of specialized signal generation hardware. Standard transmitters may not yet exist and usually do not have the flexibility to produce signals necessary to evaluate the receiver's ability to handle interference from a crowded and highly dynamic spectral environment. A thorough test also requires combining multiple transmitter outputs into a fully occupied signal scenario which mimics the complete antenna signal in an operational situation. This paper describes a signal generation system which eliminates the need for developing custom hardware. The system is easily configurable to output most existing communication signal types as well as new signal types.**

Introduction

The recent high level of activity in wireless communications has resulted in a proliferation of digitally modulated signal types. The generation and reception of these modern signals involves complicated signal processing algorithms which are unique to each signal type. In many cases the complexity is handled in software which runs on a digital signal processor (DSP). However, some of the high speed algorithms require dedicated hardware. Typically a general DSP is combined with function specific hardware to produce a complete implementation of a modern digital communications source or receiver. It has been difficult to define a general source or receiver which can be reconfigured to work with an arbitrary digital modulation type.

Motivations for a generic source

One reason for wanting a versatile source is to generate signals which deviate in some controlled manner from an ideal signal. This capability is useful in the design or manufacturing of receivers, providing a way to examine a receiver's response to non-ideal signals. For example, the symbol rate may be adjusted to examine the range over which a receiver is able to acquire symbol-lock. Standard commercial transmitters cannot be used since they are usually not designed to produce these controlled impairments.

A second application of a generic source is to produce new signal types for which commercial transmitters are not yet available. During the development of a new wireless system, designers often run into a "chicken and egg" problem—not having a source to verify a receiver nor a receiver to verify a source. Also, designers often want to test the ability of a new modulation scheme in a real-world environment to see if their assumptions about the modulation performance are correct. With a configurable source and receiver, a new modulation scheme can be tested without having to spend the time or money to build a specialized prototype of the system.

A third use for a generic source is to produce a complete spectral environment for receiver testing. This requires multiple signal sources with independent control of frequency, amplitude and modulation. With an increasing number of wireless modulation types being used, it is important for manufacturers to know that their equipment will work in a crowded spectrum. The multiple channel source can also be used to simulate multipath effects by programming several channels to produce the same signal with varying amplitude and delay so that they interfere with each other in a random fashion.

A fourth application becomes practical if the generic source is low cost and small. The generic source may be used as an operational transmitter in wireless applications where the number of installations is small enough to not warrant the cost of developing a specialized transmitter. The generic source may also be an attractive option in cases where a transmitter must be capable of generating multiple signal types. Having a specialized transmitter for each signal type may be too bulky or costly. Also, the interoperability of individual commercial transmitters may provide a logistical problem. How does the

appropriate source get activated? How about multiplexing the antenna or the data source?

To summarize, the motivations for a generic source include:

- Generation of impaired signals
- New signal types
- Real-world spectrum simulation, including multipath
- Versatile operational transmitter

Historical approaches to a generic source

There has been a progression of sources with increasing flexibility starting with simple oscillators, function generators, and sweepers. Later came signal generators with built-in amplitude, phase or frequency modulation. More recent are signal generators with I/Q modulation inputs, some of which have built-in capability to produce selected digital modulation types. A few even have the flexibility to produce impairments. These have gone a long way toward satisfying the need to generate standard signals with potential impairments. Signal generators have also been used in a few real-world spectrum simulation systems; however, the cost of these systems has limited their use and the number of independent channels which have been simulated. One problem with current signal generators which are dedicated to standard signal types, is the delayed availability for new or emerging standards.

The most flexible signal sources have been arbitrary waveform synthesizers or arbitrary waveform generators (AWS, AWG, ARB). These are often used to drive the modulation inputs of an RF signal generator to produce an arbitrarily modulated signal. Although this type of system may appear completely flexible, the practical limitations of the AWS make it fall short for some applications of a generic source. Most AWS sources use a limited memory to save a sample sequence of a signal. The samples are pre-computed and stored. Then they are played back by continuously looping through the sample memory. Care must be taken so that when the memory loops back to the beginning, there is no undesired transient in the output. This often places a constraint on the signal which is inconsistent with the desired specification. The necessity to pre-compute the samples means that an AWS cannot be used for an operational radio.

The timing of samples in AWS can provide difficulty for digital modulation using a given symbol rate. The reason for this is that the computation of samples often requires the AWS sample rate and the symbol rate be related by a small integer ratio. This may force the AWS to have a variable synthesized output sample rate.

Another limitation of the AWS approach comes from its finite memory size. This makes it difficult to simultaneously achieve both high sample rate and long duration signal segments. Getting sufficient time duration of a wideband signal with the necessary signal content is a challenge with the AWS approach.

Real-time Signal Computations

The limitations of the AWS approach has led to a new type of generic source—one which digitally computes the signal samples in real-time. Clearly, a general purpose DSP and a digital-to-analog converter can be used for real-time computation up to a limited sample rate. However, the signal bandwidth has been severely limited because of the amount of computation required by the DSP to produce each output sample. In order to achieve bandwidths required by many common digitally modulated signals, some of the computations must be done in dedicated high-speed hardware. The trick to maintaining generality is carefully choosing a flexible hardware architecture. Most digital modulation signals can be generated using the block diagram summarized in figure-1.

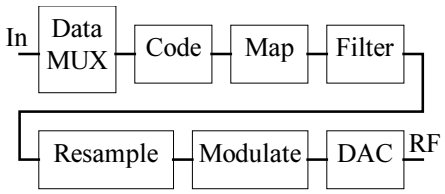

Figure-1 Generic Communication Source

The data multiplexer (MUX) is used to select the source of data for various fields within the transmitted data sequence. Most digital communications protocols define fields within a transmission. One field might contain a fixed data pattern used for synchronization. The next might contain a certain number of traffic data words. There may be guard fields with a fixed data pattern at the beginning and end of a transmission. In time domain multiplexed (TDMA) systems, the gap between bursts can be considered as another field consisting of special

data words which produce a NULL signal output. A generic communications source needs to include appropriate data source options, multiplexers, and counters to provide sufficient field generation flexibility.

The coder takes a sequence of data words and produces a sequence of symbol codes. One common example is differential coding, where the output symbol codes depend on the current and previous input data words. Another use of the coder is for error correction. For example, a 3-bit symbol sequence may be generated from 2-bit input data sequence where the additional bit of information contains redundancy used by the receiver for correcting errors. The coder should be general enough for implementing such diverse examples as Morse coding, for which the input data is a sequence of ASCII bytes and the output is a sequence of one-bit symbols with the appropriate timing to create the Morse code pattern for each ASCII character. In this last example, each input code produces a variable number of output symbols.

The map function converts a symbol code sequence to a sequence of signal samples. Mapping modes which are useful include: symbol codes to real samples, symbol codes to complex samples, dual channel (I/Q) symbols to I/Q samples. When mapping to complex sample values it is also useful to be able to specify an additional fixed angle of rotation between symbols. This is useful for implementing such modulation types as pi/4 DQPSK and VSB.

The filter function applies an arbitrary filter to the sample sequence. It needs to handle real signals, complex signals, or interleaved I/Q signals. For complex signal types the filter coefficients themselves should be allowed to be complex. This provides the capability of creating asymmetric frequency responses which are useful in simulating certain signal transmission impairments. For offset modulation it is necessary to create filters with a different amount of delay in the I and Q (real and imaginary) outputs.

The resampler block is necessary because the previous blocks all operate at some small integer multiple of the symbol rate. In order to convert the sampled sequence to a true analog signal, it is necessary to interpolate or resample the signal at some high sample rate which is matched to the analog reconstruction filter cutoff.

The modulator function takes the complex sample sequence (I/Q samples are treated as real & imaginary) and shifts the frequency by mixing it with a complex local oscillator. Alternatively, the I and Q inputs can be used to control the magnitude and frequency of the carrier. These two modulation modes allow for all combinations of AM, QAM, FM, and PM.

The DAC performs the digital-to-analog conversion of the complex sample sequence to provide I/Q analog outputs. To remove the imaging or aliasing effects due to sampling, the outputs must be filtered to select a single Nyquist interval of the output spectrum.

Provided that each of the blocks in the above block diagram are sufficiently flexible, a source with this architecture can be used to produce virtually any digitally modulated signal. In fact, it can even produce high quality analog-like modulation by treating the input data words as quantized analog samples and mapping them to a large number of finely spaced symbols. If the blocks are implemented with hardware rather than software, the source can operate at high symbol rates. Since the function of each block is specified with parameters, such as filter tap values, a user knowledgeable in communication theory can set up the parameters to produce a wide range of signal types without having to resort to programming. In contrast, to use an AWS source, the user has to actually compute the sample values. Probably the most striking advantage of the new approach is that the data sequence can be sent to the source in real-time, making it capable of transmitting an actual message with useful information as well as protocol.

A Commercial Real-Time Source Product

A product which implements the flexible block diagram of figure-1 is the HP E2748A. It is a 2.5”x4.2” board level module which conforms to the TIM-40[1] specification. One or more of these modules can be plugged into a carrier board. The analog outputs can be combined to produce a

[1] The TIM-40 specification is a widely used standard originated by Texas Instruments. It defines the physical parameters and connector pins.

multi-channel source in a small physical package. Software is included to provide an interactive user front panel so the configuration parameters can be set up without programming. All the features of the module are also accessible through a programmer's interface library which can be called from "C" or Microsoft Visual Basic user programs. The software supports both a PCI based carrier board for direct plug-in to a PC, or a VXI carrier. The PC carrier can hold up to three E2748A modules, while the VXI carrier can hold up to six. Multiple carrier cards can be used to provide higher channel count. In the VXI carrier, provisions are made to allow modules to be synchronized. This simplifies the simulation of multipath by using an E2748A for each of the reflection paths.

The configuration settings for several standard signal types have been generated and saved in files shipped with the E2748A. The files can be easily loaded using the front panel interface. These setups can be used directly for simple test signal generation, or they can be used as starting points for creating similar modulation types.

The following sections give specifics on the E2748A implementation of each block in the generic communication signal generator.

Multiplexer

The data multiplexer, shown in figure-2, consists of a field counter which controls the selection of the data source from among four available sources. The sources are: an on-board pseudo-random noise source with a repetition period of greater than 140 trillion input data words, a user loadable register, data memory with up to 128k words, and a real-time data flow via the serial COM port of the TIM-40 interface. Up to 16 fields may be specified with finite lengths ranging from 1 to 2,097,151. Any field may also be specified with infinite length, meaning that once entered, the field will never complete.

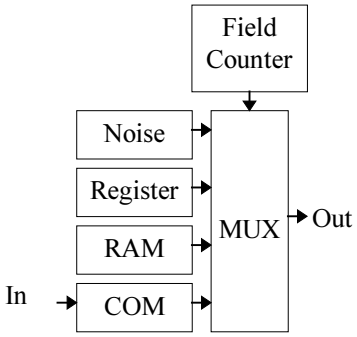

Figure-2 Data Multiplexer

Other attributes may be associated with each field. Specifically, each field may individually bypass the coder, reset the coder feedback register, reset to the beginning of the RAM data sequence, reset the noise generator, or jump back to field-0 on completion. The starting field may be specified to be other than field-0.

Coder

The coder, shown in figure-3, consists of a very general RAM based state machine. The user may configure how many of the RAM address bits are taken from the input data register and how many are taken from the feedback register. The maximum number of total address bits is 17. The RAM output is 16-bits. The most significant bit of the output may optionally be defined as a flag to request a new data word into the input register. Otherwise, a new data word will automatically be requested for each output symbol.

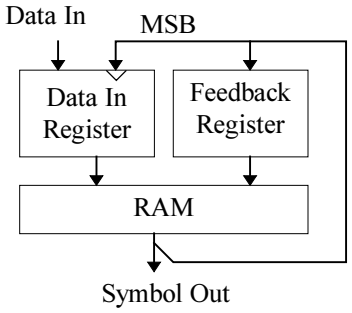

Figure-3 Coder

The simplicity of the coder structure belies its inherent power. The required flexibility stated earlier in this paper is achieved and more. The main disadvantage is the tedium of creating a coder RAM table which implements a desired function. The E2748A software addresses this for commonly used coding functions.

Map & Filter

Figure-4 shows a block diagram of the mapping and filter blocks from a functional perspective. The map merely converts symbol codes to high resolution (16-bit) complex symbol values. The complex L.O. multiplication provides for an optional symbol rotation before filtering. The multiplexer and counter insert a user selectable number of zero samples between each symbol. This increases the filter sample rate and available Nyquist bandwidth over which the filter response may be specified. The samples then enter a direct-form FIR filter consisting of an array of input data registers, each with a tap multiplier and accumulation to generate the filter output.

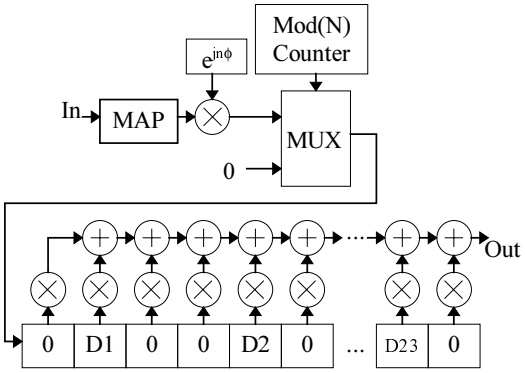

Figure-4 Map/Filter Functional Diagram

Figure-5 shows another block diagram of the mapping and filter, this time from an implementation perspective. Clearly some economies may be achieved relative to the functional diagram. First, we do not need to store, multiply or add the zero input samples. Note that these have been eliminated from the input data register array.

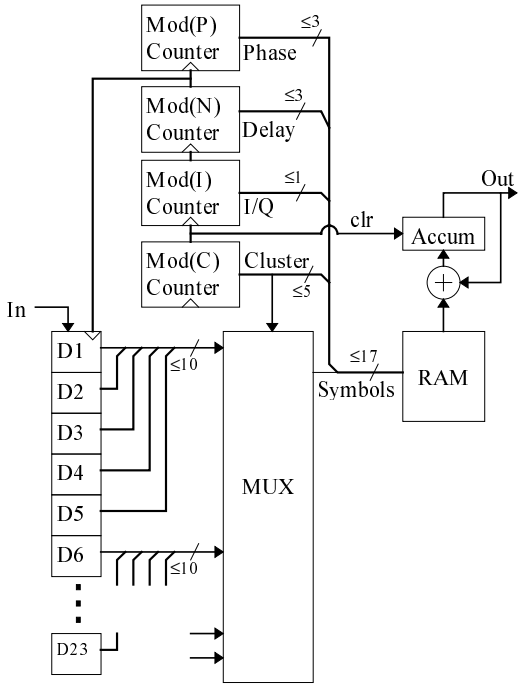

Figure-5 Map/Filter Implementation

Second, rather than map to 16-bit symbol values and then perform 16x16-bit multiplies, we can store the shorter symbol codes in the data register array rather than the mapped values associated with the codes. Also, rather than perform the multiplies at run-time, we can pre-compute the product of all possible symbols with each tap and store the result in memory. Then at run time the result can be recalled merely by looking up the product using the tap number and symbol code to form the address of the result. For example, a 32-tap filter with four possible symbols would only require 5+2 or 7-bits address space to look up all the required multiplier outputs. Since more address bits are available, it is possible to look up more that one tap product at a time. Figure-5 shows an example where the sum of five tap products are looked up with a single RAM access. Thus, we need only five RAM accesses to compute the output of a 69-tap filter (counting the interleaved zeros shown in the original filter).

The effects of the complex local oscillator can also be included in the lookup merely by expanding the memory address to include all possible rotations of the most recent symbol. One additional address bit required for filtering complex symbols is an I/Q flag which indicates whether the real or imaginary output is to be generated.

The resulting RAM address consists of the following fields: up to 10-bits representing as many symbol codes as can be packed into a single cluster of taps, up to 5-bits for the current tap cluster index, up to 1-bit to indicate I or Q output, up to 3-bits to indicate how many zero inputs have been inserted since the most recent new symbol code, and up to 3-bits to include the effects of the complex local oscillator. Since the maximum number of address bits is 17, not all fields can be configured to their maximum size at the same time.

Figure-5 shows when each counter is updated to a new value. During the time when the filter outputs are being computed, the cluster counter is incremented at the full 30MHz memory access rate. Once the cluster counter "wraps around" at the maximum cluster count, the result of the output accumulator is clocked out as a completed filter output, the accumulator is cleared, and the I/Q counter is incremented. After both the I and Q outputs have been generated the I/Q counter wraps around and the delay counter is incremented to indicate another zero input has been inserted. After the desired number of zeros have been inserted, with outputs computed for each, the delay counter wraps around. When this occurs a new input symbol is shifted into the input symbol register array and the L.O. phase counter is incremented.

Not shown in either version of the map/filter block diagram is a delay block in series with the output which allows the Q-channel to be delayed a specified number of samples relative to the I-channel. This is useful for modulations such as Offset-QPSK.

Resampler

The resampler shown in Figure-6 is implemented with a single HP proprietary application specific integrated circuit (ASIC). It consists of a filter which, in effect, convolves the sampled input sequence with a continuous impulse response. The near-Gaussian impulse response is finite in length and lasts for 7 input sample periods. The output of this continuous convolution may be computed at arbitrary instants in time by using the seven input samples which fall within the convolution interval and using the fractional time offset of the last sample to compute the appropriate multiplier for each of the samples. The coefficients must be recalculated for each new output to reflect the new position of the samples within the continuous convolution interval. Outputs equally spaced in time are computed by using the time accumulator shown in the block diagram. Whenever the time register exceeds one input sample period, a new input sample is clocked into the shift register containing the most recent seven samples. The E2748A runs the resampler output sample rate at a constant 15MHz. Thus, the input sample rate is 15MHz times the fractional time increment, ΔT. The resolution of the ΔT register is $1/(4096*10^{10})$. This results in available input sample frequency steps of 0.366uHz.

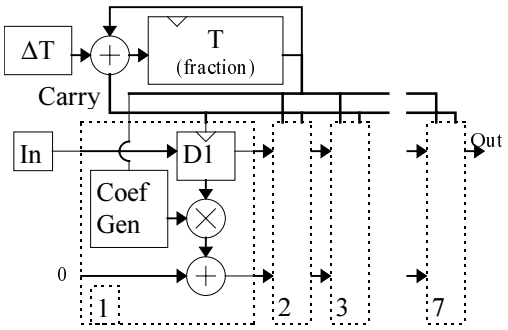

Figure-6 Resampler

A very important observation to make at this point is the fact that the computations prior to the resampler are done on an as-needed basis. The resampler requests or "pulls" data when it is needed. The actual time that the computations are performed is unimportant as long as the result is available when the resampler needs it. This "pull" timing approach distinguishes this approach from other real-time computed sources. The benefits ripple all the way back to the ultimate data source, which no longer has to worry about precise timing. The data source may be a host based program with interrupt driven requests for additional data. Coupled with the data FIFO manager built into the E2748A software, the host software virtually does not need to concern itself with timing. The timing of the entire system is controlled by the 15MHz resampler clock. This critical benefit will be recognized by any reader who has developed a real-time computation system with precision timing control.

Modulator

The modulator in figure-7 is implemented with a single ASIC. It treats the I/Q input as a complex number which it multiplies by the output of a

complex local oscillator. The local oscillator is implemented with a phase accumulator which, for each sample, increments by an amount determined by the contents of the frequency register. An alternate FM mode switches the Q input into the frequency adder, in effect making the Q port be an instantaneous increment to the center frequency. The I port controls the magnitude in FM mode.

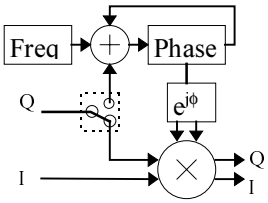

Figure-7 Modulator

Hardware implementation

The three main RAM based functions described above: data RAM, coder, and map/filter all share the use of the same physical memory. The user can select how the available memory address space is allocated among those three functions. Because of this sharing, the existing ASICs, and the simplicity of the remaining hardware, the entire physical implementation of the E2748A can be accomplished with the hardware block diagram in figure-8. FPGA-1 implements most of the features of the product including the COM port interface, while FPGA-2 provides high speed multiplexing between the resampler and modulator ASICs. Note the conspicuous absence of any programmable DSP chip. All control and data are sent to the board via the serial COM port.

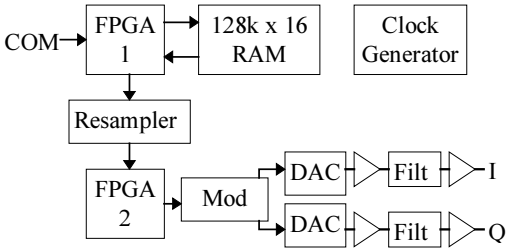

Figure-8 Hardware Block Diagram

The clock generator uses a 30MHz voltage controlled crystal oscillator which may free-run or, on a VXI carrier, may be locked to a shared 10MHz external reference. When multiple E2748As are used on a VXI carrier, a shared ECL line may be used to synchronize the start-up of the modules.

In order to save space the resampler and modulator chips are mounted as bare die. They are directly attached and wire-bonded to the printed circuit board and encapsulated within a single protective cover.

Summary

The E2748A represents a new level of communications signal generation capability. Some of its key contributions include:

- Extreme flexibility to produce a large variety of existing and new signal types.
- Real-time computations allow live transmission of non-repetitive protocol and information.
- Multiple modules may be used together to generate a complete spectral simulation, including multipath.
- High level software front panel allows setup and use without programming.
- Small size and low cost.