[To appear in Proceedings of RAWCON'99]

Waveform Synthesis for Transmission of Complex Waveforms

Matthew L Welborn[*] and John C Ankcorn

MIT Lab for Computer Science
Software Devices and Systems Group

Abstract

A new transmit architecture for digital radios is proposed that emphasizes the use of pre-computed waveforms stored in memory instead of real-time computation. We present results that show how direct synthesis techniques can be extended to directly generate many types of digitally modulated waveforms such as PSK or QAM. These techniques are especially appropriate for software radios implemented on systems with general purpose processors and large memory systems.

1. Introduction

A conventional modulator for a QAM system is shown in Figure 1. This system produces transmit waveforms from input data bits through a modulation process which has three distinct steps: (1) bit-steam to symbol-stream mapping, (2) transmit pulse-shaping and (3) frequency translation to intermediate frequency [3, 5].

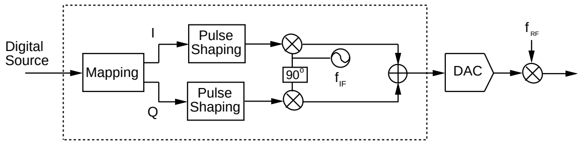

Figure 1: Conventional digital QAM modulator

This modulation process can be viewed as a mapping from input data bits to output transmit waveforms, allowing us to transform the output waveform *generation* process into one of waveform *selection*. We propose a new approach in which this mapping from bits to transmit waveforms is accomplished in a single step by using look-up tables containing pre-computed waveforms (Figure 2). This technique is related to direct digital synthesis (DDS) techniques, which are used to generate sinusoidal signals from tables of pre-computed output samples in some digital systems [4]. In this work, we have extended these ideas to the synthesis of complex digital modulation waveforms.

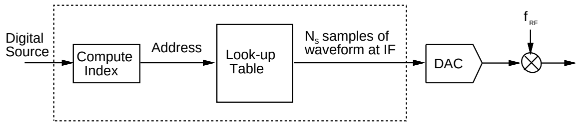

Figure 2: Proposed digital modulator

This approach is motivated by the continued rapid growth in memory capacity relative to processor speeds for some types of digital systems. These techniques are particularly well suited to software radio systems implemented on architectures which use general purpose processors (GPP) and have large amounts of main memory available for storage of pre-computed output waveforms.

2. Analysis

To produce digital waveforms, we pre-compute symbol-length sequences of output samples; the output waveform is generated by piecing together the pre-computed sequences of samples. Because bit-to-symbol mapping may not be linear and because the transmit pulse shape may extend over multiple symbol periods, the mapping is not as straightforward as just one input bit to N output samples. To understand how we can pre-compute these waveforms, we start with the discrete-time equation for a pulse-amplitude modulated (PAM) waveform with (pos-

[*]mwelborn@lcs.mit.edu, (617) 253-1000, fax: (617) 253-0147

sibly complex) symbol values a_k:

$$s[n] = \sum_{k=-\infty}^{\infty} a_k p[n - kN_s] \quad (1)$$

Here $p[n]$ is the impulse response of the transmit pulse-shaping filter (symmetric about $n = 0$) and N_s is the number of samples per symbol period. We can simplify this computation by approximating the response by $p_z[n]$, which is $p[n]$ symmetrically truncated to K symbol periods or KN_s samples:

$$p_z[n] = \begin{cases} p[n] & |n| \le \frac{KN_s}{2} \\ 0 & |n| > \frac{KN_s}{2} \end{cases} \quad (2)$$

One effect of truncating this response in the time-domain is to spread the spectrum of the output signal in the frequency domain, so K must be selected with this in mind. As the filter response is now non-zero only for a period of K symbols, the approximated output waveform, $\hat{s}[n]$, only depends on the values of K symbols (centered around $k_n = \mathrm{Round}[\frac{n}{N_s}]$) at any given time:

$$\begin{aligned} s[n] \;\approx\; \hat{s}[n] &= \sum_{k=-\infty}^{\infty} a_k p_z[n - kN_s] \\ &= \sum_{k=k_n-\frac{K}{2}}^{k_n+\frac{K}{2}} a_k p_z[n - kN_s] \quad (3) \end{aligned}$$

This baseband signal is produced at the output of the pulse shaping filter in the typical QAM modulator shown in Figure 1.

This same equation (3) allows us to compute a one-to-one mapping from a set of K input symbols (or BK bits if $|a_k| = 2^B$) to a sequence of N_s output samples. In this way, we compute all possible symbol-length sequences of output samples and store these values in a look-up table which is used to directly generate the same output waveform as in Figure 2. Accepting some limitations on the available carrier frequencies allows us to incorporate the step of translating the signal to IF in the table, thereby improving performance. When we translate the baseband signal, $\hat{s}[n]$, to IF with center frequency f_c we get the following passband signal:

$$\begin{aligned} y[n] &= 2\mathrm{Re}\left\{ e^{j2\pi f_c nT} \sum_{k=k_n-\frac{K}{2}}^{k_n+\frac{K}{2}} a_k p_z[n - kN_s] \right\} \\ &= \sum_{k=k_n-\frac{K}{2}}^{k_n+\frac{K}{2}} c_k \cos(2\pi f_c nT + \theta_k)\, p_z[n - kN_s] \quad (4) \end{aligned}$$

where the symbol values have been re-written as $a_k = c_k e^{j\theta_k}$. Restricting the choice of center frequencies to integer multiples of the symbol rate, that is $f_c = k/N_sT$ for some k, results in the $\cos(2\pi kn/N_s + \theta)$ term in the right hand side of (4) having an integer number (k) of periods in N_s samples. This ensures that the we will have continuous phase in the generated IF waveform.

3. Implementation

The size of our look-up table is $2^{BK} \times N_s \times W$ bytes where there are $|a_k| = 2^B$ symbols in the constellation and we use W-byte length samples. For example, a 4-PAM system with a 6-symbol length pulse-shaping transmit filter generating 10 samples per symbol and 2 bytes/sample (16 bits) requires a table size of 80 Kbytes, which is quite reasonable and would actually fit in the *cache* on most modern GPPs. For larger symbol constellations or longer filter responses, the table size grows rapidly (exponentially in the length of the filter response or number of bits/symbol). One way to limit this growth is to decompose the large table into several smaller tables in which we perform parallel look-ups of partial waveforms, adding the pre-computed values to produce the final waveform. To use this technique, we must find an appropriate way to decompose the required sample sequences.

One solution is to decompose the table into D separate tables (where D divides K) by having the index into each table depend only on $\frac{K}{D}$ consecutive symbols (or $\frac{BK}{D}$ consecutive bits) instead of the full K symbols (or BK bits) as shown in Figure 3. Now the total table space is $D \times 2^{\frac{BK}{D}} \times N_s \times W$ bytes and the cost of this decomposition is that we perform $(D-1)$ additions for every output sample and $(D-1)$ additional table index calculations for each sequence of N_s output samples. A system such as 16-QAM with a 6-symbol length pulse-shaping filter would require $4 \times 2^6 \times N_s \times W = 5120$ bytes (with $D = 4$) or $2 \times 2^{12} \times N_s \times W \approx 164$ Kbytes (with $D = 2$) when $N_s = 10$ and $W = 2$ bytes/sample.

Decomposing the look-up table into multiple smaller tables allows us to reduce the total table size at the expense of one or more additions per output sample. This trade-off between memory resources and computational resources enables this technique to be used on systems with different relative amounts of resources. To increase our ability to trade computation for memory, we can also modify the above technique to generate sequences of output samples that are multiple symbol-length periods in length. If we wish to generate samples for M symbols, we can compute tables which map from $(K + M - 1)$ input symbols to $(M \times N_s)$ output samples. (The initial technique above was a special case of this for $M = 1$.) We can then use the table decomposition technique with multiple-symbol look-up tables to produce a general

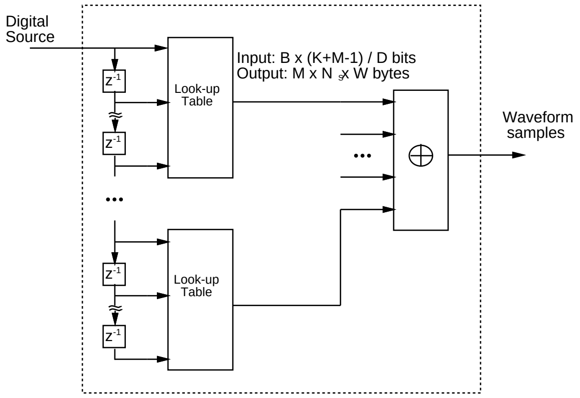

Figure 3: QAM modulator using parallel look-ups to reduce table size.

technique which give us increased flexibility to trade-off between computation (additions) and memory requirements. The space requirement for this general technique is now $D \times 2^{\frac{B(K+M-1)}{D}} \times M \times N_s \times W$ bytes of memory, requiring at most $(D-1)$ additions to produce each output sample from the results of the individual table lookups.

In Figure 4 we show the total memory requirements for both a 8-PAM system and a 16-QAM system. These plots show the trade-off between memory usage and computation using the table decomposition described above to generate sample sequences for multiple symbols.

This technique of pre-computing output sample sequences instead of direct computation is also helpful when we need an output waveform which might contain multiple signals combined using frequency multiplexing, as might be done in a wideband cellular or PCS transmitters. In such systems, we can actually generate each of the individual signals at passband with the correct center frequencies, so that we can simply add the resulting individual channel signals.

4. Performance

To provide a comparison of our proposed architecture with more conventional architectures, we have analyzed equivalent operation counts required to produce a single output sample for each scheme.

The operations required to produce a single output sample of a digital waveform at IF using the conventional architecture (as in Figure 1) are summarized in Table 1. In this table we assume that the local oscillator used to generate the IF carrier is implemented using direct digital synthesis techniques.

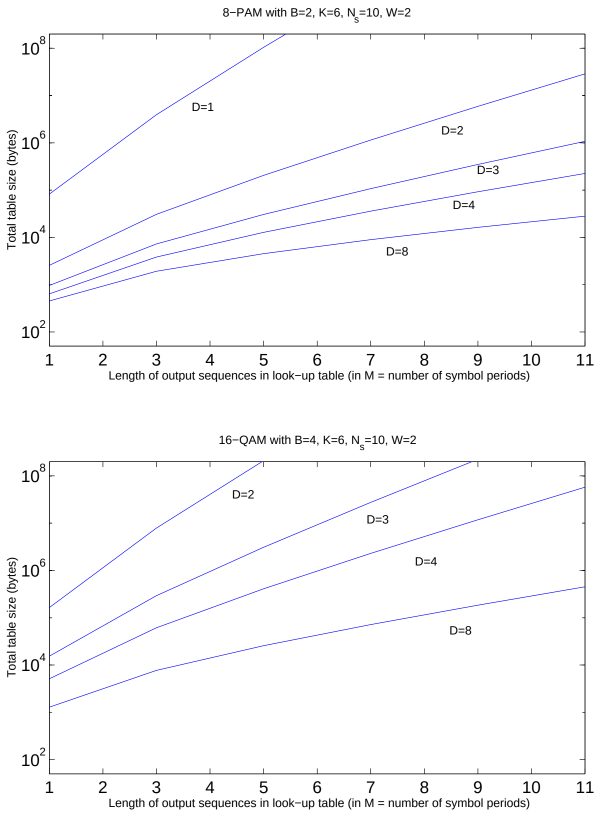

Figure 4: Plot showing memory requirements versus number of symbol periods per table entry for (a) 8-PAM and (b) 16-QAM waveform synthesis with $K = 6$, $N_s = 10$.

For the proposed architecture, Table 2 summarizes the operations required to produce a single output sample. In this case, the operations required to compute the table index are performed only once for each sequence of MN_s samples, which are stored in a contiguous region of the look-up table.

The algorithms developed in this work have been implemented in a software radio system designed and built as part of the SpectrumWare research project at the MIT Lab for Computer Science. This *Virtual Radio* system is a complete, functioning software radio that uses a wideband receiver/down-converter and a specially designed PCI-based I/O system for data transfer. The system supports a sustained data transfer rate in excess of 500 Mbps to and from software applications running in user space

Function	Required operations
Filter (FIR)	$2 \times K \times N_s$ multiply-accumulate ops
DDS for LO	$2\times$ array-fetch + increment DDS phase accumulator
Mix to IF	$2\times$ multiplies + 1 addition

Table 1: Required operations for conventional modulator

Function	Required operations
Compute Index	$\frac{1}{MN_s} \times \left\{ \begin{array}{l} \mathrm{B-bit\ shift} \\ \mathrm{modulo\ operation} \\ \mathrm{add\ symbol\ value} \end{array} \right.$
Look-up in table	array-fetch + increment sample index

Table 2: Required operations for new proposed modulator

on a GPP-based personal computer. Further information on this system can be found in [1, 2, 6, 8, 7].

Current implementations of these algorithms on a 400 MHz Pentium II-based PC can generate multiple narrowband digital signals at sample rates as high as 10 Msamples/sec. This system can generate continuous digital modulation signals with effectively any constellation or pulse shape: such as BPSK, 8-PAM, 4-QAM with any desired pulse shaping or excess bandwidth characteristics. To perform the digital combining of multiple signals at IF we have used single-instruction, multiple-data (SIMD) instructions, such as the MMX instruction set, to provide improved performance.

5. Conclusions

This work demonstrates an effective technique for trading-off memory usage for computation complexity in a digital transmitter. We anticipate that these techniques will become valuable as the memory capacities of systems continue to increase. These techniques also show promise for implementations which require low-power consumption, due to the inherent advantage of memory devices over computational elements.

6. Acknowledgements

This research was supported under the DARPA Global Mobile Communications project and also under a National Science Foundation graduate research fellowship.

7. References

[1] Vanu G. Bose and Alok B. Shah. Software Radios for Wireless Networking. In *Proceedings of Infocomm '98*.

[2] Michael Ismert. Making Commodity PCs Fit for Signal Processing. In *USENIX'98*. USENIX, June 1998.

[3] Edward A. Lee and David G. Messerschmitt. *Digital Communication*. Kluwer Academic Publishers, 2nd edition, 1994.

[4] Jeffrey H. Reed. Direct Digital Synthesis. In *Tutorial notes from PMIRC 1998*.

[5] James Gunn, Kenneth Barron, William Ruczczyk. A Low Power DSP Core-based Software Radio Architecture. *IEEE JSAC issue on Software Radios*, April 1999.

[6] David L. Tennenhouse and Vanu G. Bose. The SpectrumWare Approach to Wireless Signal Processing. *Wireless Network Journal*, 2(1), 1996.

[7] Matthew L. Welborn. Narrowband Channel Extraction for Wideband Receivers. In *Proceedings of ICASSP'99*.

[8] Vanu Bose; Mike Ismert; Matt Welborn and John Guttag. Virtual Radios. *IEEE JSAC issue on Software Radios*, April 1999.