Final Report

on

Dual-Tone Multiple Frequency (DTMF) Detector Implementation

Güner Arslan

for

EE382C

Embedded Software Systems

May 1998

Abstract

Dual-tone Multi-frequency (DTMF) signals are used in touch-tone telephones as well as many other areas such as interactive control, telephone banking, and pager system. As analog telephone lines are converted to digital, researchers became interested in digital DTMF detectors.

There are many digital DTMF detecting algorithms, but most of them do not comply with the related International Telecommunications Union (ITU) and Bellcore recommendations or are not suitable for real-time implementation.

In this project, my primary aim was to implement the new DTMF algorithm proposed by M. D. Felder, J. M. Mason, and B. L. Evans on a TMS320C50 digital signal processor for 24 channels. My design tool in this task was Ptolemy, to be more specific the C50 domain, which caused several problems during the project. In order to overcome these problems, I started to write a new target for the C50 domain which is named as SimC50 due to its ability to run assembly codes on a simulator and display the results. Despite the fact that current C50 targets generate code for TI TMS320C50 DSP Starter Kit (DSK), the new target generates code for TI TMS320C50 Evaluation Module (EVM).

At the time this report was written all the tasks of the new target except getting the results from the simulator part, were finished. Additional work was needed to modify some C50 stars because they were written for DSK only.

1 Introduction

Dual-tone multi-frequency (DTMF) is an international signaling standard for telephone digits. These signals are used in touch-tone telephone call signaling as well as many other areas such as interactive control applications, telephone banking, and pager systems.

A DTMF signal consists of two superimposed sinusoidal waveforms whose frequencies are chosen from a set of eight standardized frequencies. These frequencies were chosen in Bell Laboratories, where DTMF signaling system were originally proposed as an alternative to pulse dialing system in telephony.

The detector part of early DTMF systems consisted of analog implemented bandpass filterbanks, which were tuned to the eight standard frequencies. As analog lines as well as many other analog systems were converted to digital, researchers became interested in digital DTMF detectors. Digital implementation has many advantages over analog implementation such as accuracy, stability, re-programmability, and chip count; that is, instead of using several analog chips for detecting multi-channel DTMF tones, only a digital signal processor (DSP) chip is used for all channels.

Many digital DTMF detecting algorithms have been proposed [1], [2], [3], [4] but they have several drawbacks as follows:

- Most of them do not comply with the related International Telecommunications Union (ITU) and Bellcore recommendations.
- 2. Some of them have too heavy computational load.
- 3. Some of them need too much memory to implement on a low cost DSP.

The algorithm proposed by M. D. Felder, J. M. Mason, and B. L. Evans [5],[6] is an efficient DTMF detection algorithm which complies with all ITU and Bellcore specifications and does not need any buffering. The inventors, simulated their algorithm on Ptolemy's synchronous data flow (SDF) domain and concluded that this algorithm could be implemented for 24 channels, on a low cost fixed point DSP such as the TMS320C50. In this project, my aim was to implement the mentioned algorithm on a TMS320C50 for 24 channels.

This report is organized as follows: Section 2 gives background information about DTMF signals, standards, and detectors. Section 3 is focused on implementation topics such as modeling and code generation in C50 domain, the new target for C50 domain and the motivation to write it. The last section concludes this report.

2 Background

A DTMF signal consists of two superimposed sinusoidal waveforms whose frequencies are chosen from a set of eight standardized frequencies. For example, by pressing the "1" button on the touch-tone telephone key pad in Figure 1, a signal consisting of a 697 Hz sinusoid and a 1209 Hz sinusoid is generated. A DTMF detector attempts to detect these frequencies in the presence of noise, and determines which button is pressed.

	Coloumn frequencies (Hz)					
Row frequencies (Hz)		1209	1336	1477	1633	
	697	1	2	3	А	
	770	4	5	6	В	
	852	7	8	9	С	
	941	*	0	#	D	

Figure 1: Touch pad in DMTF systems

Detection of frequencies in noisy environment is a well studied area in digital signal processing. The difficulty of DTMF tone detection is due to the standards which must be satisfied when these signals are detected. A summary of ITU and Bellcore's DTMF standards and recommendations are given below:

1) Signal frequencies:

Low group (Hz): 697, 770, 852, 941

High group (Hz): 1209, 1336, 1477, 1633

2) Frequency tolerances:

- a. Frequencies with an offset less than $\pm 1.5\%$ must be accepted.
- b. Frequencies with an offset more than $\pm 3.5\%$ must be rejected.

3) Signal Reception Timing:

a. Tones with a duration less than 23 ms must be rejected.

b. Tones with a duration more than 40 ms must be accepted.

c. An interruption of more than 40 ms must be accepted as a pause (One tone has finished, a new one has started).

d. An interruption of less than 10 ms must be tolerated (The tone continuous).

4) Twist (power difference between frequencies):

a. The low frequency may have 8 dB higher power

b. The high frequency may have 4 dB higher power

Bellcore has some recommendations which are in fact a subset of ITU specifications. However, Bellcore does not only give some standards, but also some standardized performance tests as well.

Some Bellcore tests and recommendations which are different from ITU specifications are as follows:

1) Bellcore guard time test:

Guard time is the minimal length of a tone that can be reliably detected. The guard time is determined by decreasing tone lengths at the input of the detector and counting the number of detects. The receiver guard time is calculated in ms as (500- total detects)/10 and must be under 40 ms.

2) Bellcore power level test:

A minimum detection of 25 out of 35 at a tone power of -25 dBm is required.

3) Bellcore talk-off Tests:

The most important test might be the talk-off test which determines how often a detector detects a speech as a valid tone. Bellcore provides three one-hour audio tapes which include over 50,000 speech samples, to test a detector. Testing a DTMF detector with all these speech samples is equivalent to testing the detector on one million calls if it was be used in a local central office. Bellcore specified the following allowed false detection numbers: For digits 0-9; 333 for digits 0-9, *, #; 500 and for digits 0-9, *, #, A-D; 600.

Since these standards were determined when DTMF detectors were analog their aim was to specify optimum analog detectors. For example, the standard frequencies are determined so that they have no common multipliers. This guarantees that none of the frequencies have common harmonics and thereby improves the performance of analog detectors. However, the most commonly used digital frequency analysis technique, the Discrete Fourier Transform (DFT), samples the frequency domain with equally spaced samples, and therefore, it is not possible to have a sample exactly at each standard frequency.

This fact guided researchers to use a Non-uniform Discrete Fourier Transform (NDFT) in

DTMF detection algorithms. Although it was possible to have arbitrary located samples in frequency domain, more computation was required when using the NDFT. The tradeoff between high frequency resolution and less computational load is the main problem in designing DTMF detectors.

The new algorithm [5],[6] complies with all frequency specification of ITU, pasts all Bellcore test, and its computationally efficiency is such that it can be implemented on a DSP for 24 channels. The algorithm is based on NDFT but uses about 30 different modifications. These modifications result in an algorithm which requires no buffering and according to the inventors simulations, requires approximately 1000 words of data memory, 2000 words of program memory and 24 million instructions per second (MIPS) when implementing on a digital signal processor for 24 channels.

3 Modeling and Implementation

In order to implement the algorithm on a TMS320C50, my plan was to go through these steps:

- Model the algorithm in Ptolemy's C50 domain,
- Generate code for TMS320C50 within the C50 domain,
- Optimizing the generated code by hand such that it can be implemented on a DSP for 24 channels.

Since the algorithm was already modeled in Ptolemy's SDF domain, I wanted to re-target this model to C50 domain. Before re-targeting the SDF universe to C50, I checked whether in the computations any number exceeds the range of -1 to 1. This is important since I would implement the algorithm on a fixed point processor. Although the creators of the SDF domain model scaled the numbers in the algorithm in several places, I found 3 points where the numbers exceeds this range. By re-scaling, I fixed this future problem. Then I created a new universe which included only the DTMF detector galaxy, a dummy constant input and two black-holes to connect the outputs of the detector galaxy. The new universe is shown in Figure 2.

3.1 New Stars

All stars which are used in a source domain should be present in the target domain to be able to re-target a universe. Therefore I have written the required TMS320C50 assembly codes to create the C50 versions of these three new stars which are used in the SDF domain model. These stars were:



Figure 2: The *DTMFCode* universe which includes only the detector galaxy and some dummy input and outputs

- 1. *Power*: This star computes the power of an input stream. The length of the stream can be modified by the user.
- 2. DtmfNdftCoeff: This stars returns the NDFT coefficients for one of the 8 standard DTMF frequencies. These coefficients will be used as an input to the NdftVarD star and are defined as $2\cos(\frac{2\pi f_i}{f_s})$ where f_i is a DTMF frequency and f_s is the sampling rate which can be defined by the user.
- 3. NdftVarD: This star accepts a NDFT coefficient and a stream as input and computes the NDFT over some number of samples which can be defined by editing the numSamples parameter. A second parameter sequentialExecs determines how many times the algorithm will be executed without reseting the internal states. For example, if you specify numSamples=106, sequentialExecs=2 and a coefficient of f_i =697 is fed to the coefficient input, this star will compute the 2 × 106 = 312 point DFT at the frequency point f_i =697 Hz.

3.2 Problems in C50 Code Generation

The next step was to generate the TMS320C50 assembly code of the DTMF detector within Ptolemy. At this step I encountered some problems:

- 1. The generated code was 105 standard letter pages long,
- Ptolemy generates only code for Texas Instrument (TI)'s TMS320C50 DSP Starter Kit (DSK).
- 3. In the generated code, some memory locations are allocated several times.
- 4. The C50 domain did not have a simulation target.

Why the first item was a problem is obvious. Managing 105 pages of code is not easy. A large portion of the generated code, consisted of memory declaration directives. Since the C50 domain uses a SDF model it allocates memory for every arch in the graph. In other words the SDF over-models our application.

The second item was a problem because I wanted to generate code for the TI TMS320C50 Evaluation Module (EVM). Although both boards have the same processor they are quite different. The differences between EVM and DSK is summarized in Table 1

DSK	EVM	
• No emulation hardware	On-board emulation	
• No external memory	64K single access RAM (SRAM)	
• Simplified assembler	Full feature assembler	
• Generated code is not in common object	COFF format code generation	
file format (COFF)		
• Does not have a linker phase, assembler	Memory is allocated by	
does the memory allocation	the linker	
• Different assembler directives: .ds .ps .entry	.bss .usect .sect .asect	

Table 1: Differences between DSK and EVM

The main difference is that the DSK has very limited features compared with the EVM. The most important difference in my case is the fifth item in Table 1; since the DSK does not have a linking phase, memory allocation is done by the assembler. This means that for every code and data block, memory is allocated immediately after the first line is encountered. Since the assembler does not know what kind of code or data is in the following part of the code it cannot allocate the current block such that the memory usage is optimum. On the other hand, in the EVM, an assembler is used to generate object files for every code and data section, and then a linker is used to allocate memory for each object by using their size. This results in an optimum memory allocation. When working with small projects, it might be possible to manually specify starting addresses for each code and data block for optimum memory allocation, but if the assembly code is as long as 105 pages, this is not an easy job.

3.3 Conversion from DSK compatible to EVM compatible code

At this point, I had the desired assembly code in DSK compatible format and wanted to convert it to EVM compatible code. At the beginning, it seemed to be a substitution task only. I thought substitution of all DSK assembler directives to EVM assembler directives should solve my problem. Therefore I wrote a PERL script which did this job. But after running the script on the code I realized that the conversion is not as easy as I thought.

The problem was that some numbers which I supposed to be addresses were in fact not

addresses. For example, one snapshot of the C50Test star:

LAR AR2,#0001	; load auxiliary register #2 with 1
XC 1,LT	; execute the next code if acc < 0 $$
SBRK #01h	; subtract 1 from auxiliary register

Although the AR registers are generally used as index registers, in this piece of code the AR2 register is used like an accumulator. In this case, it is not possible to discriminate whether the number in the LAR command is an address or an immediate value by looking only the LAR instruction line.

3.4 New Target: SimC50

Since, a code as long as mine cannot be converted from DSK compatible to EVM compatible manually, writing a new target which generates assembly code for EVM was the best solution. Therefore I started to write a new target in C50 domain which is called SimC50. This domain will be able to generate code for EVM and, if required, run it on the EVM simulator to obtain memory and run time statistics.

One important point is that the C50 stars are written with only DSK in mind; as a result some of these stars include assembler directives which work only with DSK assembler and are not compatible with EVM assembler. Therefore all these non-compatible stars have to be fixed such that both it is possible to use them with both targets. By the time this report was written the new target was generating code and was compiling code with the EVM assembler. Although the linked code could be loaded into the EVM simulator there were still some problems with running and getting the required information back to Ptolmey.

4 Conclusion and Future Work

In this project, my aim was to implement the new ITU and Bellcore compliant, zero buffering, NDFT based DTMF detection algorithm proposed by M.D. Felder, J.M. Mason, and B.L. Evans on a TMS320C50 for 24 channels. Since the inventors of this algorithm had already simulated the new algorithm in Ptolemy SDF domain, I planed to use the C50 domain to generate C50 assembly code for the detector and optimize it by hand.

Since, in order to re-target a universe from one domain to another, all stars have to be present in the target domain, I have created the C50 versions of the three new SDF stars.

The generated code was not compatible with the TI TMS320C50 EVM board and an automatic conversion from DSK to EVM compatible code was not easy. This motivate me to write a new target for the C50 domain which can generate code for the EVM and can simulate it to obtain memory requirements and computation time information. However, all stars in C50 domain were written for the DSK target and many of them were not compatible for use with the new target. Therefore I started to modify all C50 domain stars such that they were compatible with both DSK and EVM targets. By now, the new target is able to generate EVM compatible code and use the assembler and linker to generate executable binaries. These binaries can be loaded to the simulator, but running and passing required information like memory usage and run time back to Ptolemy is not finished yet.

References

- S. L. Gay, J. Hartung, and G. L. Smith, "Algorithms for multi-channel DTMF detection for the WE DSP32 family," Proc. IEEE Int. Conf. Acoust. Speech Signal Processing, pp. 1134–1137, May 1989.
- [2] P. C. Mock, "Add DTMF generation and decoding to DSP-μ P design," *Electronic Design Engineers Magazine*, pp. 205-220, March 1985.
- [3] S. Bagchi and S. K. Mitra, "An efficient algorithm for DTMF decoding using the subband NDFT," Proc. IEEE Int. Symp. Circ. Syst., pp. 1936–1939, April 1995.
- [4] G. Arslan, B. L. Evans, F. A. Sakarya, and J. L. Pino, "Performance evaluation and real-time implementation of subspace, adaptive, and DFT algorithms for multi-tone detection," *IEEE Int. Conf. on Telecommun.*, pp. 884–887, April 1996.
- [5] M. D. Felder, J. C. Mason, and B. L. Evans, "Efficient ITU-compliant dual-tone multiple-frequency detection using the non-uniform discrete Fourier transform." *IEEE Signal Processing Letters*, To Appear.
- [6] M. D. Felder, J. C. Mason, and B. L. Evans, "Efficient digital ITU-compliant, zero-buffering, DTMF detection algorithm using the non-uniform discrete fourier transform." Filed March 10, 1998. A Non-Confidential Specification Sheet is available at http://www.utexas.edu/academic/otl/software.html#DTMF.