DSP-Based Hierarchical Neural Network Modulation Signal Classification

Namjin Kim, Nasser Kehtarnavaz, *Senior Member, IEEE*, Mark B. Yeary, *Member, IEEE*, and Steve Thornton

***Abstract*—This paper discusses a real-time digital signal processor (DSP)-based hierarchical neural network classifier capable of classifying both analog and digital modulation signals. A high-performance DSP processor, namely the TMS320C6701, is utilized to implement different kinds of classifiers including a hierarchical neural network classifier. A total of 31 statistical signal features are extracted and used to classify 11 modulation signals plus white noise. The modulation signals include CW, AM, FM, SSB, FSK2, FSK4, PSK2, PSK4, OOK, QAM16, and QAM32. A classification hierarchy is introduced and the genetic algorithm is employed to obtain the most effective set of features at each level of the hierarchy. The classification results and the number of operations on the DSP processor indicate the effectiveness of the introduced hierarchical neural network classifier in terms of both classification rate and processing time.**

***Index Terms*—Digital signal processor (DSP) implementation, hierarchical neural network classifier, modulation signal classification.**

I. Introduction

ANALOG or digital modulation is an integral part of any communication system such as television, radio broadcasting, and cellular transmission. In applications ranging from frequency surveillance or intercepting and deciphering unwanted communication, knowledge of the communication signal modulation type is needed for proper demodulation of the signal. The identification of the signal modulation type has both civilian and military applications. Knowledge by civilian authorities of the modulation type aids in monitoring unauthorized transmitters. In electronic warfare, this knowledge is a factor in threat detection, specific emitter identification, acquiring targets, and homing.

The classification of modulation signal can be grouped into two categories. One category classifies either analog modulation signals or digital modulation signals exclusively; the other category includes both analog and digital modulation signals. Initial attempts at modulation signal classification appeared in [1]–[4]. Later, Azzouz and Nandi introduced a decision theoretic approach in [5]. Six signal features were utilized in their work for both analog and digital modulation signals and a number of user-defined thresholds were employed to perform the classification. A total of 13 classes were considered with an average classification rate of 93% at 15 dB signal-to-noise ratio (SNR). Nandi and Azzouz extended their work by using artificial neural networks for both modulation signal types in [6], where it was no longer required to specify a threshold for each feature.

In [7], we used both parametric and nonparametric classifiers to classify 9 analog and digital modulation signals using 29 signal features. The parametric classifiers consisted of Bayes and Markov chain, and the nonparametric classifiers consisted of Parzen window and neural network. After examining the classifiers, the Bayes classifier with the majority vote was obtained to produce the best performance in terms of classification rate.

This paper discusses a hierarchical neural network classification scheme for 11 modulation signal classes and white noise using 31 signal features. Appropriate features are selected by the genetic algorithm (GA) at each level of the hierarchy. Most notably, it discusses the hardware implementation of the classifier on a high performance digital signal processor (DSP) processor, namely the TMS320C6701. It is shown what steps are taken in order to optimize the classifier code on this processor for real-time operation.

The paper is organized as follows. Section II presents a brief overview of modulation signals. In Section III, a review of the classical classifiers is presented. The developed hierarchical neural network classifier is also presented in this section. Section IV discusses our choice of the hardware platform. In Section V, the implementation issues related to DSP software pipelining are discussed. Finally, the conclusions are stated in Section VI.

II. Modulation Signals

The modulation is the process which changes the signal from one form into a different form to allow its transmission over a communication channel. The recovery of the signal is done by an inverse process called demodulation. More details on signal modulation can be found in many textbooks on communications, for example [8].

The modulation signals considered here consist of the following analog modulations: CW (carrier wave), AM (amplitude modulation), FM (frequency modulation), SSB (single side band); and the following digital modulations: FSK2 (frequency shift keying 2), FSK4 (frequency shift keying 4), OOK (on–off keying), QAM16 (quadrature amplitude modulation 16), and QAM32 (quadrature amplitude modulation 32).

The extraction of modulation signal features is done in two domains: instantaneous and demodulation. In the instantaneous

Manuscript received September 15, 2002; revised February 28, 2003. This work was supported by a grant from L-3 Communications Integrated Systems to the University of Texas at Dallas.

N. Kim and N. Kehtarnavaz are with the Department of Electrical Engineering. University of Texas at Dallas, Richardson, TX 75083-0688 USA (e-mail: kehtar@utdallas.edu).

M. B. Yeary is with the School of Electrical and Computer Engineering, University of Oklahoma, Norman, OK 73019 USA.

S. Thornton is with L-3 Communications Integrated Systems, Greenville, TX 75402 USA.

Digital Object Identifier 10.1109/TNN.2003.816037

TABLE I
CLASSIFICATION RATES OF THE CLASSICAL CLASSIFIERS

Classifier	Classification rate (%)
Bayes	99.02
Markov chain	98.60
Parzen window	91.54
Neural network	99.52

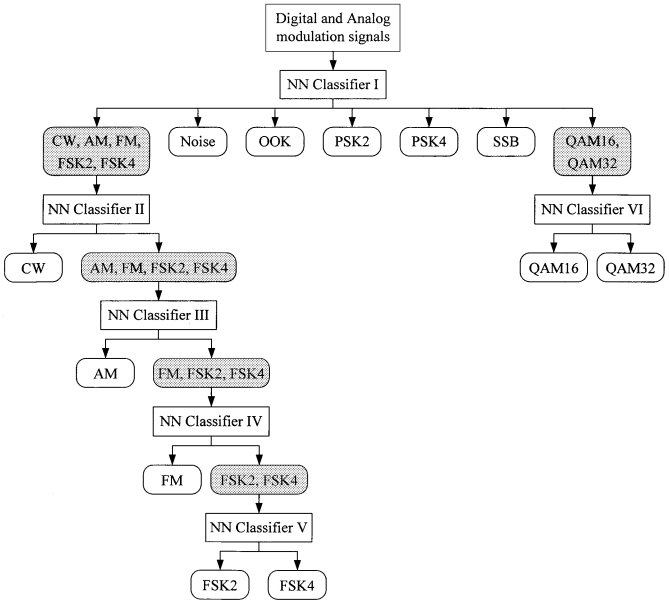


Fig. 1. Classification hierarchy.

domain, the features associated with instantaneous amplitude, frequency, and phase of the modulation signals are extracted. In the demodulation domain, it is assumed that the received signal is either FM, AM, or PM. The baud rate is first estimated by examining the transition length of the signal. Then the features associated with demodulated amplitude, frequency, and phase are extracted. A total of 31 signal features are thus extracted from both domains. The details of the signal features are reported in [7],[9] and are not repeated here. The interested reader is referred to these references for details.

For classifier development purposes, analog and digital modulation signals were generated by computer. The training dataset consisted of 1024 signal sequences per modulation class and the testing dataset consisted of 400 signal sequences per modulation class with no overlap with the training dataset. Each sequence contained 30 signal segments. This meant a total of 1024×30 training and 400×30 test samples. The SNR was varied randomly from 7 to 12 dB as part of the signal generation process.

III. MODULATION CLASSIFICATION

This section presents the classical classifiers and the developed hierarchical neural network classifier for identifying the modulation signals.

A. Classical Classification—A Prelude to Comparison

In order to make a comparative analysis between generally accepted classifiers and our developed classifier, four classical classifiers were investigated for modulation signal classification. Two of the classifiers were parametric while the other two were nonparametric. The parametric classifiers consisted of Bayes and Markov chain, and the nonparametric classifiers consisted of Parzen window and neural network. For the parametric classifiers, the density functions associated with the features were assumed to be Gaussian. For the nonparametric classifiers, the form of the density functions were considered to be unknown and were estimated from the data.

1) Parametric Classifiers: The first parametric classifier considered was the Bayes classifier having a joint Gaussian density. As it is well-known, this classifier provides the class that maximizes the *posteriori* class probability. No relationship among the classes is assumed when using this classifier. The overall classification rate of the Bayes classifier was found to be 99.02%.

The Markov chain classifier was the second parametric classifier considered. In this classifier, given a sequence of observations, the class sequence that satisfied the Bayes rule was identified. The Markov chain classifier provided an overall classification rate of 98.60%.

2) Nonparametric Classifiers: For nonparametric classification, no *a priori* information about the density function was assumed. In the first nonparametric classification approach, the density function for each class was estimated using the Parzen window [10]. The classification using the Parzen window did not have the distinct phases associated with training and testing. Probability values of a density function were directly calculated from the training data. Then, the class that produced the maximum probability was chosen. The overall classification rate using this nonparametric classifier was obtained to be 91.54%.

The second nonparametric classifier considered was the multilayer perceptron (MLP) neural network, which is widely used in classification applications. In this neural network, the nodes are arranged in three layers: input, hidden, and output. There are two sets of weights for which to train: the weights between the input and hidden layer and the weights between the hidden and the output layer. Given a set of input–output pattern pairs, the network changes its connection weights in such a way that the total squared error is minimized.

In this work, the conjugate gradient (CG) algorithm [11] was used for training. After different neural network structures were examined, the neural network with 32 hidden nodes was found to produce the best performance. For faster convergence, the initial weights were obtained by the Nguyen–Widrow algorithm [11] during the training phase.

Table I provides a comparison of the overall classification rates of the above four widely used classical classifiers. From this table, it can be seen that the best performance was achieved by the neural network classifier. However, none of these classifiers led to a real-time response when implemented on the DSP processor. The classification approach discussed next is thus developed to achieve real-time classification.

B. Hierarchical Classification of Modulation Signals

This section presents the developed hierarchical neural network classifier in order to lower the processing time and hence

TABLE II
HIERARCHICAL NEURAL NETWORK CLASSIFIER CONFUSION MATRIX WITHOUT MAJORITY VOTE (COMPUTER GENERATED SIGNALS)

-	AM	CW	FM	F2	F4	N	OOK	P2	P4	SSB	Q16	Q32
AM	11820	1	174	-	-	-	4	-	1	-	-	-
CW	-	12000	-	-	-	-	-	-	-	-	-	-
FM	186	59	11357	19	206	27	126	6	14	-	-	-
F2	-	-	10	11104	733	82	9	58	4	-	-	-
F4	-	-	83	613	11303	-	-	-	1	-	-	-
N	-	-	-	-	-	11996	-	-	-	4	-	-
OOK	19	-	-	-	-	3	11967	-	4	2	4	1
P2	-	-	4	-	-	56	10	11930	-	-	-	-
P4	-	-	9	-	-	2	4	1	11892	6	40	46
SSB	-	-	-	-	-	229	96	-	38	11530	61	46
Q16	-	-	-	-	-	-	18	-	45	-	8829	3108
Q32	-	-	-	-	-	-	12	-	58	-	3280	8650

TABLE III
HIERARCHICAL NEURAL NETWORK CLASSIFIER CONFUSION MATRIX WITH MAJORITY VOTE (COMPUTER GENERATED SIGNALS)

-	AM	CW	FM	F2	F4	N	OOK	P2	P4	SSB	Q16	Q32
AM	11999	-	1	-	-	-	-	-	-	-	-	-
CW	-	12000	-	-	-	-	-	-	-	-	-	-
FM	1	50	11949	-	-	-	-	-	-	-	-	-
F2	-	-	-	11850	40	68	-	42	-	-	-	-
F4	-	-	-	96	11904	-	-	-	-	-	-	-
N	-	-	-	-	-	12000	-	-	-	-	-	-
OOK	-	-	-	-	-	-	12000	-	-	-	-	-
P2	-	-	-	-	-	-	-	12000	-	-	-	-
P4	-	-	-	-	-	-	-	-	12000	-	-	-
SSB	-	-	-	-	-	-	-	-	-	12000	-	-
Q16	-	-	-	-	-	-	-	-	-	-	11939	61
Q32	-	-	-	-	-	-	-	-	-	-	184	11816

to achieve real-time operation. The use of a hierarchical structure allows improvements in both processing time and classification rate since such a structure narrows the classification process to fewer classes and thus fewer features at each level of the hierarchy.

1) Hierarchical Classifier Structure: The hierarchical classifier is illustrated in Fig. 1. We have previously reported this structure in [9]. The basic idea of hierarchical classification is based on the "divide and conquer" strategy. From the above classical classification analysis, it is seen that there are two class groups, referred to as superclasses here, that produce interclass errors. One group consists of the AM, CW, FM, FSK2, and FSK4 classes, the other consists of the QAM16 and QAM32 classes.

The procedure adopted for building the hierarchy is as follows. At the first level of the hierarchy, seven classes that generated no overlap were identified. They included OOK, PSK2, PSK4, SSB, white noise, and the above two superclasses. For each superclass, the signal features were examined in order to find the ones which divided that superclass into two smaller superclasses with no overlap. GA was used to select those features that were effective for classification while eliminating those features that were redundant for classification. As a result, the first superclass was broken down into the CW class and a smaller superclass consisting of the AM, FM, FSK2, and FSK4 classes.

TABLE IV
HIERARCHICAL NEURAL NETWORK CLASSIFIER CONFUSION MATRIX WITHOUT MAJORITY VOTE (REAL SIGNALS)

-	AM	CW	FM	FSK2	Noise	OOK	PSK2
AM	102	3	1	1	-	-	-
CW	1	35	1	-	-	-	-
FM	-	1	83	1	1	-	1
FSK2	-	-	1	130	1	-	-
Noise	-	-	-	-	35	-	-
OOK	-	-	-	-	-	30	-
PSK2	-	-	-	-	-	-	35

This division process was performed repeatedly until all the modulation classes were separated as indicated in Fig. 1.

GA is capable of searching a high-dimensional feature space in an efficient way. Basically, GA maximizes a fitness function, corresponding to classification rate in our case, by carrying out three stochastic types of searching or operations: population, crossover, and mutation. A chromosome set consisting of various points in the search space is initially set up. Then, the population operation is applied to make clones of the chromosomes from a previous generation. The number of the chromosomes in a new generation is determined by the fitness value of the chromosome set in the old generation. Next, the crossover operation is applied to select two chromosomes at random. Random

TABLE V
HIERARCHICAL NEURAL NETWORK CLASSIFIER CONFUSION MATRIX WITH MAJORITY VOTE (REAL SIGNALS)

-	AM	CW	FM	FSK2	Noise	OOK	PSK2
AM	107	-	-	-	-	-	-
CW	-	37	-	-	-	-	-
FM	-	-	87	-	-	-	-
FSK2	-	-	-	132	-	-	-
Noise	-	-	-	-	35	-	-
OOK	-	-	-	-	-	30	-
PSK2	-	-	-	-	-	-	35

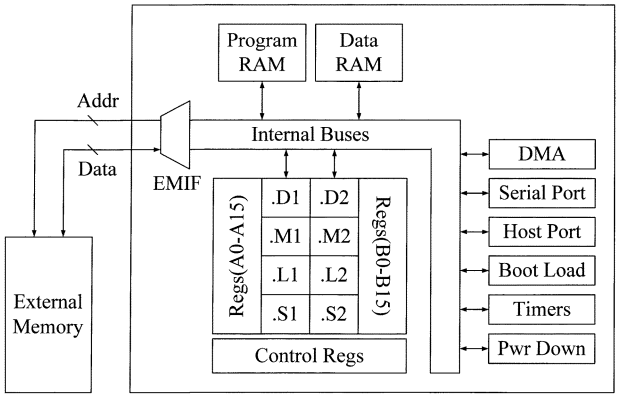


Fig. 2. Generic $C6\times$ architecture [13].

portions of these two selected chromosomes are chosen and swapped. Finally, the mutation operation is applied to invert random bits within the chromosomes. More details on GA is beyond the scope of this paper and can be found in [12].

We used each bit of the input feature vector x to GA to represent a corresponding feature. For example, $x_i = 0$ meant the ith feature was not selected by GA. GA operations continuously regenerated chromosomes to maximize classification rates. Finally, only those chromosomes which contributed positively to the classification were selected.

2) Hierarchical Neural Network Classifier (Computer Generated Signals): Based on the above hierarchical structure, an MLP neural network was designed at each level of the hierarchy. For the computer generated modulation signals, the GA outcome produced the following numbers of features corresponding to the six classification levels indicated in Fig. 1: 4, 4, 7, 2, 8, and 6. Table II shows the confusion matrix without using a majority vote and Table III shows the confusion matrix when using a majority vote over a moving window of size 30. In these tables, the labels N, P2, P4, F2, F4, Q16, and Q32 represent Noise, PSK2, PSK4, FSK2, FSK4, QAM16, and QAM32 classes, respectively. By using this classifier, the overall classification rates were obtained to be 93.32% and 99.62% without and with using the majority vote, respectively.

3) Hierarchical Neural Network Classifier (Real Signals): In addition to the computer generated modulation signals, real signals from live communication sources with known modulation types were collected and processed in real time. These sources included FM radio stations, cellular control channel, pager transmission, ham radio, air traffic control, and weather station. The collected samples were equally and randomly divided into a training and a testing set with no overlap. Table IV shows the confusion matrix without using a majority vote, producing an overall classification rate of 97.19%. Table V shows the confusion matrix when using a majority vote over a moving window of size five. It was very encouraging to see no error when using the majority vote. These high classification rates for real signals exhibited a clear evidence of the effectiveness of the hierarchical neural network classifier.

TABLE VI
NUMBER OF CLOCK CYCLES PER NODE FOR DIFFERENT CLASSIFIERS

	Bayes	NN	H-Bayes	H-NN
Min	145281632	3485710	991889	30643
Max	145281632	3485710	3240168	172261
Average cycle	145281632	3485710	1744428	66588
Average time (ms) CPU clock 167 MHz	871.69	20.91	10.47	0.40

IV. DSP HARDWARE PLATFORM

The developed hierarchical neural net classifier can be implemented on various hardware platforms. We have chosen to use a DSP hardware platform for the reasons mentioned below.

There is a fair amount of flexibility in terms of programmability as far as DSP processors are concerned. This flexibility is quite limited when using a customized very large scale integration (VLSI) chip. Since in many communication systems, it is desired to have software updating capability, a DSP platform is ideal to use since device upgrading or updating can be easily achieved by downloading a new program into the processor memory. Cost effectiveness is another benefit of using a DSP platform. Considering that communication devices are normally mass produced, DSPs provide a more cost-effective implementation platform as compared to a customized VLSI chip [13].

DSPs also offer a more attractive platform as compared to general purpose processors since their instruction sets are optimized for performing repetitive or looping type of operations [13]. Such operations are encountered in many communication applications including the one discussed in this paper.

The TMS320C6701 DSP processor used here is a 32-bit floating-point processor possessing a very long instruction word (VLIW) architecture. The central-processing-unit (CPU) of the $'C6\times$ contains 32 32-bit registers and eight functional units including two multipliers and six arithmetic units. This implies that the $'C6\times$ can execute up to eight instructions per clock cycle. The 32 registers and eight functional units are divided into two sides, called A and B. As shown in Fig. 2, each side of the processor has four functional units (.D, .L, .M, and .S) and 16 registers (A0-A15 or B0-B15). Since the architecture of this processor is symmetric, algorithm execution on one side is independent from the other side. This increases parallelism or throughput.

Software implementation on the $'C6\times$ can be done in C, linear assembly, or hand-optimized assembly. The compiler and assembly optimizer software tools provided by Texas

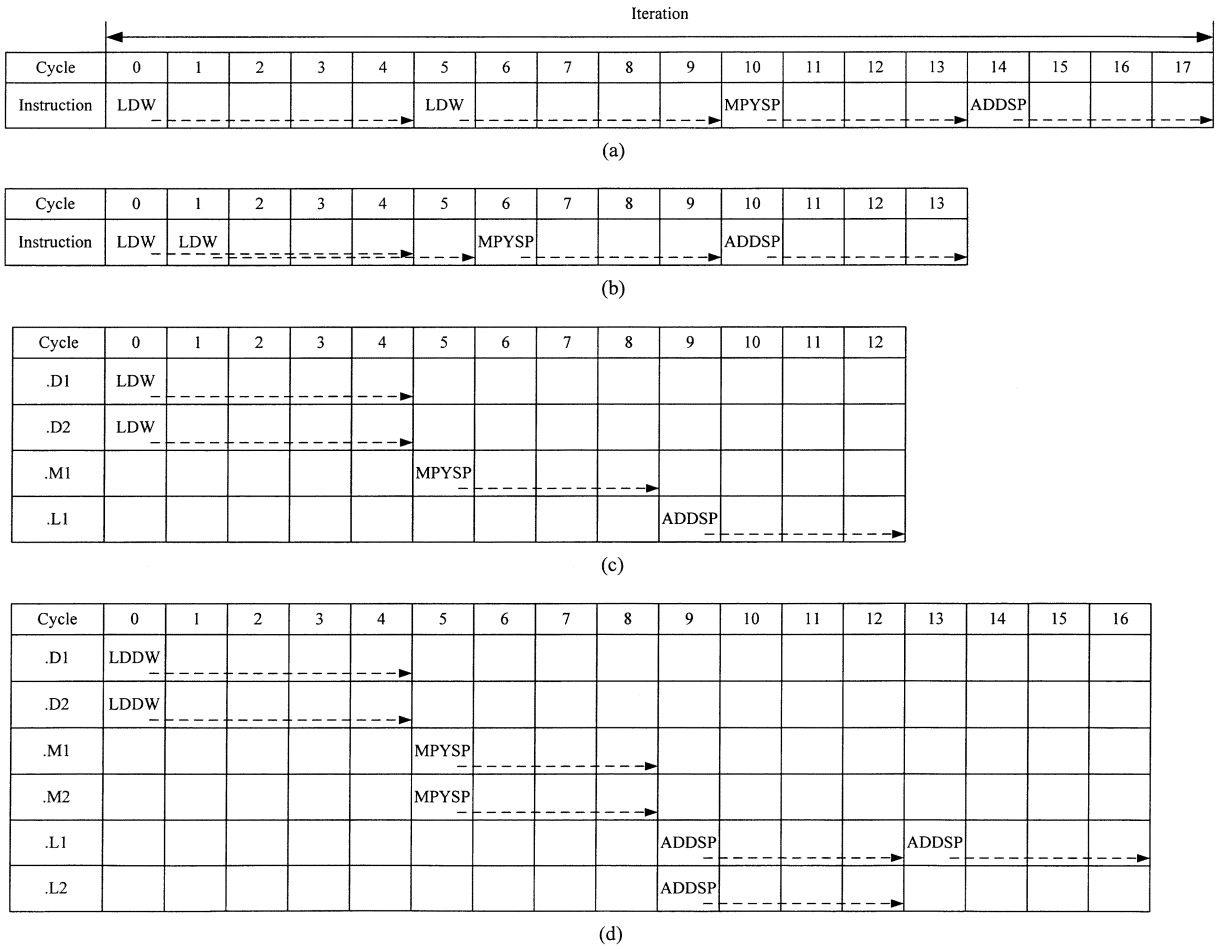


Fig. 3. Scheduling tables showing the optimization steps for an n-input node. (a) No optimization ($18n$ cycles). (b) Filled delay slots ($14n$ cycles). (c) Parallel instruction ($13n$ cycles). (d) Double word-wide instruction ($(17/2)n$ or $9n$ cycles).

Instruments are capable of optimizing the C and linear assembly codes, respectively. In many applications, acceptable performance is obtained by using the compiler or the assembly optimizer. However, in our case we found it necessary to hand-optimize the classifier code in order to achieve real-time throughput. The process of hand optimization is referred to as software pipelining. The details on the software pipelining procedure are presented in Section V.

As compared to a floating-point processor, a fixed-point processor, in general, has a less complex CPU and, thus, can operate at a higher processing speed. However, on a fixed-point processor, it is required to use so called Q-format data representation [13]. In our case, the use of Q-format data representation for the developed classifier led to some precision loss at each node of the neural nets and, thus, the cumulative effect resulted in additional misclassifications. That is why it was decided to use a floating-point processor instead of a fixed-point processor. The use of a floating-point processor allowed us to have a wide dynamic range for the weights and inputs of the designed neural nets.

V. DSP Implementation

A comparison of the number of cycles for different classification algorithms is provided in Table VI. All the programs were initially written in C and the maximum compiler optimization level was used for comparison purposes. As can be seen from this table, the processing time was greatly improved by using the hierarchical classification approach. For example, the ratio of the Bayes classifier clock cycles to the hierarchical Bayes classifier clock cycles was $145\,281\,632/1\,744\,428 = 83.28$, and for the neural network to the hierarchical neural network was $3\,485\,710/66\,588 = 52.34$.

Each node in the neural networks had a simple mathematical structure which was basically a dot product of the inputs with the weights followed by an activation function. The matrix inversion and multiplication within the Bayes classifier, however, consumed a considerable number of cycles as compared to the dot product and the activation function of the neural net nodes. As indicated in Table VI, the hierarchical neural network classifier took 66 588 cycles to classify one 31-dimensional feature vector. In other words, the hierarchical neural network classifier ran $1\,744\,428/66\,588 = 26.19$ times faster than the hierarchical Bayes classifier.

A. Code Optimization by Software Pipelining

To ensure real-time DSP throughput, the hierarchical neural network classifier code was turned into a software pipelined code. The activation function was implemented via a lookup

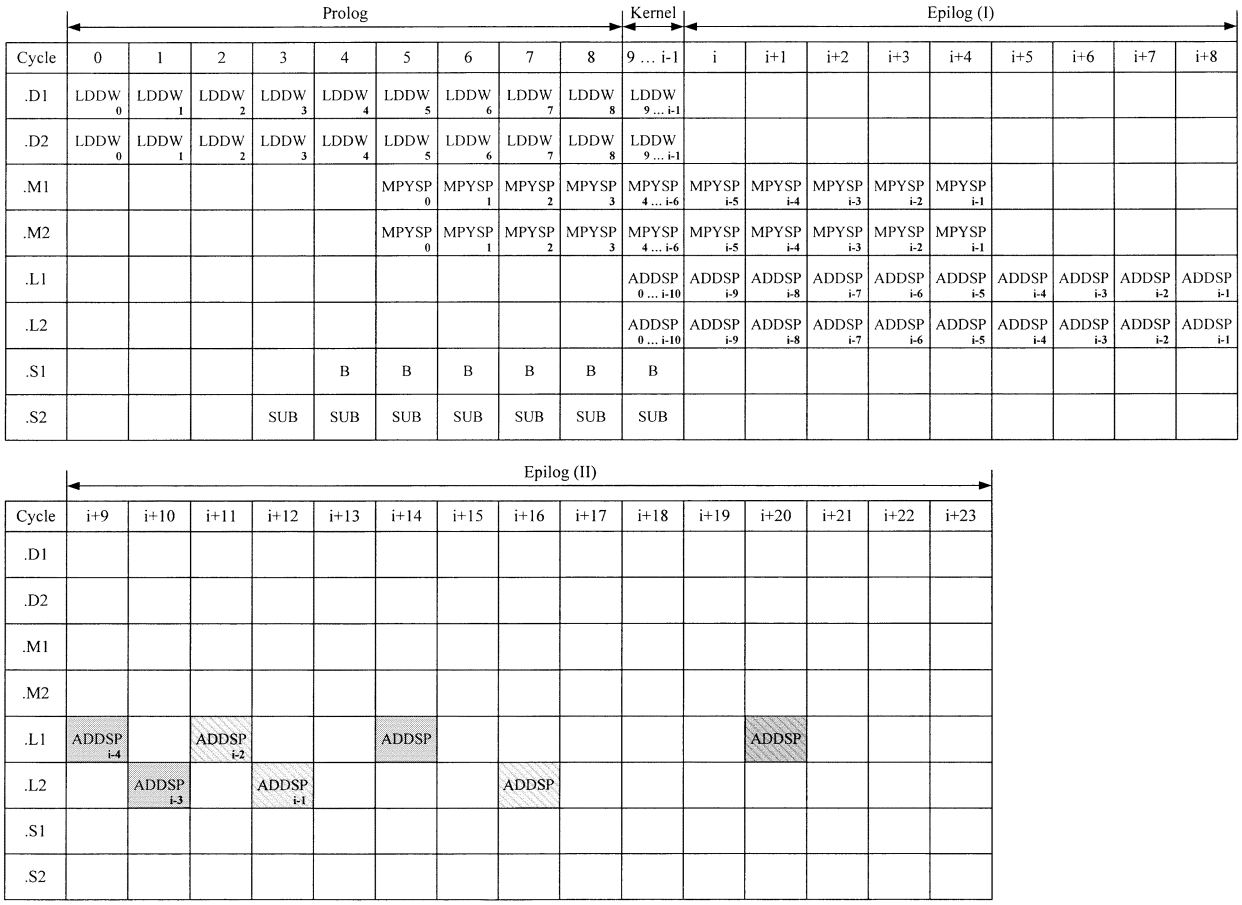


Fig. 4. Scheduling table illustrating software pipelining.

table. So the major challenge was to optimize the dot product operation associated with each node. For nonoptimized assembly, the number of cycles for one iteration was 18 cycles. For an n-input node, this led to $18n$ cycles. By filling the delay-slots, the number of cycles per iteration was reduced to 14 leading to a total of $14n$ cycles. By using parallel instructions, indicated by the double pipeline symbol "||," the number of cycles was lowered to $13n$. Attention had to be paid to avoid resource conflicts among the functional units.

By using the double word-wide capability of the processor, the functional units were fully utilized within the same cycle without any resource conflicts, and thus the number of cycles was reduced by one half since two inputs and two weights were loaded in one cycle. Then two independent multiplications were performed simultaneously. Similarly, two additions were carried out at the same time. The above optimization steps are illustrated in Fig. 3.

The scheduling table of the software pipelined code is shown in Fig. 4. This table consists of three parts: prolog, loop kernel, and epilog. Columns in this table show the instruction sets for all iterations of the code. Note that in the epilog (II) part of the scheduling table, since single-precision add instructions, ADDSP, required three delay slots, an extra 15 cycles of epilog was added to get the correct value. For instance, as shown in Fig. 4, the addition instruction at cycle $(i + 14)$ is for adding two values calculated at cycles $(i + 9)$ and $(i + 10)$. These

```
; LDDW   Load double word from memory with an offset
; MPYSP  Single-precision floating-point multiplication
; ADDSP  Single-precision floating-point addition
; B      Branch
; SUB    Signed integer subtraction without saturation

Loop:    LDDW   .D1   *A4++,A11:A10
 ||      LDDW   .D2   *B4++,B11:B10
 ||      MPYSP  .M1X  A10,B10,A6
 ||      MPYSP  .M2X  A11,B11,B6
 ||      ADDSP  .L1   A6,A7,A7
 ||      ADDSP  .L2   B6,B7,B7
 || [B1] B      .S1   Loop
 ||      SUB    .S2   B1,1,B1
```

Fig. 5. Software pipelined code. This small part of the classification code represents how pipelining is implemented in assembly language.

operations are shaded in the figure. Similarly, the addition of the values calculated at cycles $(i + 11)$ and $(i + 12)$ is performed at cycle $(i+16)$. These operations are slanted in the figure. The last sum for cycles $(i + 14)$ and $(i + 16)$ is done at cycle $(i + 20)$. The final result is therefore acquired at cycle $(i + 24)$. Thus, as seen from the scheduling table, the number of cycles for an n-input node is equal to $\lceil n/2 \rceil + 24$, where the bracket $\lceil\ \rceil$ denotes next larger integer.

The loop kernel part of the software pipelined assembly code is shown in Fig. 5 with a description of the instructions. A comparison of the code using different optimization approaches is shown in Table VII. As can be seen from this table, the least number of cycles was achieved by using the software pipelined assembly version of the code.

TABLE VII
NUMBER OF CLOCK CYCLES PER NODE FOR DIFFERENT IMPLEMENTATIONS

Implementation method	Number of cycle
C	802
C with maximum optimization option	96
Assembly	680
Assembly (software pipelined)	40

The code size was reduced by replacing the epilog (I) portion of the code with the loop kernel. To obtain the same outcome as before, the loop kernel was repeated nine more times. While the total number of cycles remained the same, the code size without the epilog (I) portion became 25 words as compared to 34 words previously. The dot product portion of the code together with the weights were also placed on the internal DSP memory. This allowed elimination of the delays associated with the external memory accesses.

VI. CONCLUSION

In this paper, a hierarchical neural network classifier has been introduced for the classification of 11 modulation signal classes plus white noise. It has been shown that the developed hierarchical neural network classifier performs better than the classical classifiers (Bayes, Markov chain, Parzen, neural network) in terms of both processing time and classification rate. This classifier was implemented on the TMS320C6701 DSP processor and live modulation signals were successfully identified. Various DSP optimization techniques were deployed to minimize the processing time and hence to achieve real-time DSP throughput.

REFERENCES

[1] T. Gallaghan, J. Pery, and J. Tjho, "Sampling and algorithms aid modulation recognition," *Microwaves RF*, vol. 24, no. 9, pp. 117–119,, 1985.

[2] F. Jondral, "Automatic classification of high frequency signals," *Signal Processing*, vol. 9, no. 3, pp. 177–190, 1985.

[3] J. Aisbett, "Automatic modulation recognition using time domain parameters," *Signal Processing*, vol. 13, no. 3, pp. 323–328, 1987.

[4] L. Dominguez, J. Borrallo, J. Garcia, and B. Mezcua, "A general approach to the automatic classification of radio communication signals," *Signal Processing*, vol. 22, no. 3, pp. 239–250, 1991.

[5] E. Azzouz and A. Nandi, "Automatic modulation recognition—Part I," *J. Franklin Inst.*, vol. 334B, no. 2, pp. 242–273, 1997.

[6] A. Nandi and E. Azzouz, "Modulation recognition using artificial neural networks," *Signal Processing*, vol. 56, pp. 165–175, 1997.

[7] T. McKinney, N. Kehtarnavaz, N. Kim, and S. Brown, "Classification of analog and digital modulation signals," in *Proc. IEEE Int. Symp. Intelligent Signal Processing and Communication Systems*, Nashville, TN, Nov. 2001, pp. 28–32.

[8] J. Proakis and M. Salehi, *Communnication Systems Engineering*. Englewood Cliffs, NJ: Prentice-Hall, 2002.

[9] N. Kim, N. Kehtarnavaz, S. Brown, and T. McKinney, "Hierarchical classification of modulation signals," in *Proc. World Automation Congress*, Orlando, FL, June 2002, pp. 243–248.

[10] R. Duda, P. Hart, and D. Stork, *Pattern Classification*. New York: Wiley, 2001.

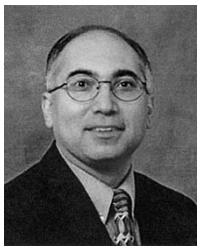
[11] C. Looney, *Pattern Recognition Using Neural Networks*. Oxford, U.K.: Oxford Univ. Press, 1997.

[12] H. Vafaie and K. De Jong, "Genetic algorithms as a tool for feature selection in machine learning," in *Proc. IEEE Int. Conf. Tools With Artificial Intelligence*, Arlington, VA, Nov. 1992, pp. 200–203.

[13] N. Kehtarnavaz and M. Keramat, *DSP System Design Using the TMS320C6000*. Englewood Cliffs, NJ: Prentice-Hall, 2001.

Namjin Kim received the B.S. and M.S. degrees in electrical engineering from Korea University, Seoul, in 1995 and 1997, respectively, and the M.S. degree from the Department of Electrical Engineering at Texas A&M University, College Station, in 2002. He is currently working toward the Ph.D. degree from the Department of Electrical Engineering, University of Texas-Dallas.

His research interests include algorithm implementation on DSP processors, neural networks, signal and image processing, and pattern recognition.



Nasser Kehtarnavaz (S'82–M'86–SM'92) received the Ph.D. degree in electrical and computer engineering from Rice University, Houston, TX, in 1987.

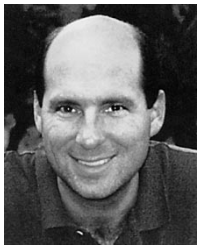
He is currently a Professor of electrical engineering at the University of Texas-Dallas. Previously, he was a Professor of electrical engineering at Texas A&M University, College Station. He has authored or coauthored two books and more than 120 journal and conference papers in the areas of signal and image processing, real-time imaging, and pattern recognition. He is currently serving as the Editor-in-Chief of the *Journal of Real-Time Imaging*. His research interests include signal and image processing, real-time imaging, and pattern recognition. More information on his research activities is available at http://www.utdallas.edu/~kehtar

Dr. Kehtarnavaz is an Associate Editor of the IEEE TRANSACTIONS ON IMAGE PROCESSING. He is a Registered Professional Engineer.

Mark B. Yeary (S'95–M'00) received the B.S. (honors), M.S., and Ph.D. degrees from the Department of Electrical Engineering at Texas A&M University, College Station, in 1992, 1994, and 1999, respectively.

He is currently an Assistant Professor with the School of Electrical and Computer Engineering, University of Oklahoma, Norman, OK. In 2002, he worked for Raytheon as a member of the Radar Signal Processing Group. His main responsibility was to design an all-digital Hilbert Transform/Low-pass filter/Decimation system-on-a-chip scheme for a KA band radar. His research interests are in the areas of digital signal processing as applied to radar signal processing, digital communications, image processing, adaptive filter design, and embedded DSP systems.

Dr. Yeary is a member of the Tau Beta Pi and Eta Kappa Nu honor societies.



Steve Thornton received the B.S. and a M.S. degrees in electrical engineering from Oklahoma State University, Oklahoma City, in 1990 and 1992, respectively.

He is a Principal Engineer at L-3 Communications, Greenville, TX. His current focus is on spatial signal processing applications implemented on embedded multiprocessor systems.