A Multi-DSP Laboratory Course

Bernhard Rinner, Reinhard Schneider, Christian Steger and Reinhold Weiss
Institute for Technical Informatics
Technical University of Graz, AUSTRIA
{*Rinner, Schneider, Steger, RWeiss*}*@iti.tu-graz.ac.at*

ABSTRACT:

This paper describes a laboratory course at our institute using TMS320C3x and TMS320C4x digital signal processors (DSPs). This laboratory course is intended for graduate students in Electrical Engineering or Telematics. It aims at introducing DSP processors and their special features as well as at demonstrating the advantages of these features, based on different experiments. These experiments range from simple audio processing by means of a DSP Starter Kit to parallel image processing using a multi-DSP system.

Keywords*: TMS320C3x, TMS320C4x, PPDS, laboratory course, parallel processing*

1. INTRODUCTION

The education of digital signal processing (DSP) in theory as well as in practice has always been a challenging task [1]. The laboratory course described in this paper was completely revised for the fall semester 1997 and is indented for graduate students in Electrical Engineering and Telematics. (The curriculum of Telematics combines Electronics, Communications Engineering and Computer Science.) These students have a basic knowledge in signal processing and programming skills in coding assembler and 'C'. Some of them have already experiences in parallel processing and in distributed systems.

This laboratory course aims at introducing DSP processors in general, their special features. The advantages resulting from the use of these features are pointed out in different experiments in the area of digital signal processing. Further goals of this laboratory course are that students get an understanding of simple DSP algorithms and an introduction to parallel processing.

The remainder of this paper is organized as follows: Section 2 presents a short overview of the laboratory course and a preceding mandatory lecture for this course. Section 3 introduces the required laboratory equipment. Section 4 describes the four experiments in more detail. Section 5 presents a course feedback. Section 6 concludes the paper with a short summary.

2. COURSE OVERVIEW

The laboratory course is preceded by the lecture "*VLSI-Prozessoren*". The attendance of the lecture is mandatory for the laboratory course. An overview of the lecture and the laboratory course is depicted in figure 1. The lecture deals with technological and performance aspects of modern VLSI processors. It is organized in three blocks:

- The first chapter deals with performance criteria such as pipelining and superscalar technologies, memory hierarchies or special instruction sets, covering high end RISC processors such as DEC Alpha and microcontrollers of the Mc638xx family.
- The objective of the second chapter is to teach principles of digital signal processors and basic digital signal processing algorithms. Special features of DSP architectures are described (HW loop support, addressing modes, etc.) and illustrated by examples. An overview of the TMS320 DSP processor family from Texas Instruments (TI) is given at the end.
- The third part introduces the architecture (data path, memory architecture, instruction set, etc.) of the TM320C3x and TMS320C4x processors. The design of the boards used in the laboratory are described at the end of the lecture.

The laboratory course, immediately following the lecture, is organized in four experiments:

- Experiment #1 introduces DSP processors

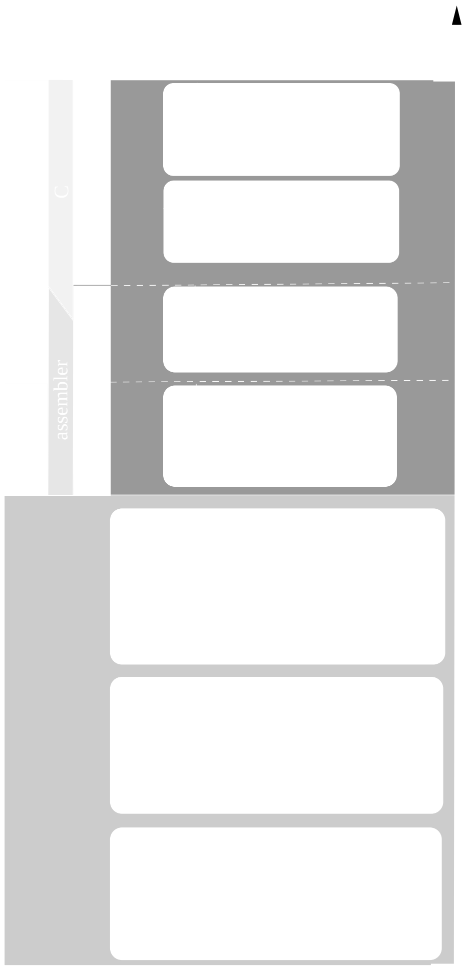

Figure 1: *Overview of the lecture and laboratory course "VLSI Prozessoren"*

and is based on the TMS320C3x DSP Starter Kit (C30 DSK) and a simple ‘audio box' with a function generator and a loud-speaker. The equipment is made available to enable home training for the students.

- Experiment #2 is based on the TMS320C30 Evaluation Module (C30 EVM) and the Code Composer software as programming environment. This experiment shows how to take advantage of the DSP processor features by programming in assembler. For that, the students optimize typical DSP algorithms like FIR and IIR filters.
- Experiment #3 deals with more complex DSP tasks in the area of image processing. The image processing algorithms are implemented in 'C' using the Code Composer and the Parallel Programming Development System (PPDS) based on four TMS320C40 processors.
- Experiment #4 introduces the parallel processing capabilities of the TMS320C40 DSP. In this experiment the image processing algorithms of experiment #3 are parallelized using up to four processors of the PPDS.

2.1 RELATED DSP-COURSES

DSP courses are prepared by educators in electrical and computer engineering departments all over the world. Most of them [3],[4],[5] are using boards employing TMS320C3x and TMS320C5x processors. TMS320C4x processors are rarely used in DSP related education of students. For example, in [2] an educational development environment for parallel image processing is presented.

3. LABORATORY EQUIPMENT

Since the objective of this laboratory is to impressively demonstrate the advantage the special features of DSPs, appropriate equipment is required to be able to explore these features. Support for parallel processing can only be demonstrated on a parallel platform. Other features, like hardware loop control etc., can easily be shown by means of a single processor platform like the DSK. Thus, the equipment changes according to the requirements of the respective goals of experiments.

3. 1 HARDWARE

The first experiment is intended for individual home training. A PC acting as host is available to each student. Students, who do not have a PC at home, can work in public laboratories at our university. The special equipment for experiment #1 consists of a complete C30 DSK package, a simple ‘audio box’ with a function generator and a loudspeaker, a power supply for both devices and cables (see figure 2, lower left corner). All the equipment is made available by the institute for every student. The other experiments are made in the laboratory in groups. For experiment #2 a host PC with a C30 EVM board inside is connected to a function generator and an oscilloscope to test the implemented filters (see figure 2, right side). Experiment #3 and #4 require a PPDS connected via a XDS510 without any further external devices (see figure 2, top). The development software is responsible for data input and output for these experiments.

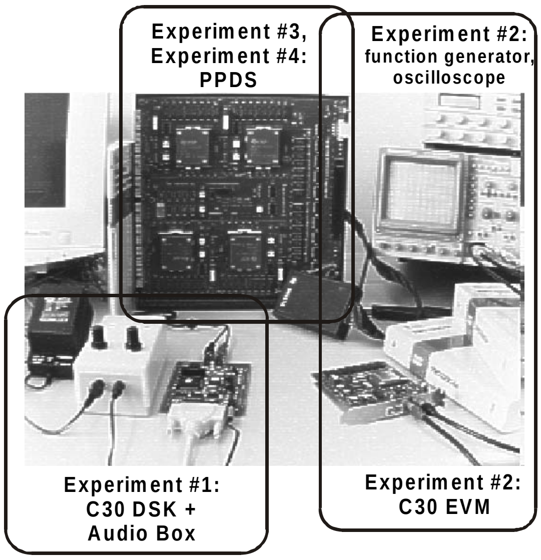

Figure 2: *Hardware equipment used for the laboratory course*

3. 2 SOFTWARE

During the first experiment, the students should become acquainted with the assembly instructions of the TMS320 floating point DSP family of TI. The software tools shipped with the C30 DSK (assembler, debugger) are perfectly suited for this purpose. From the second experiment on, the Code Composer development environment from Go-DSP and the high level language tools (HLL tools) of TI are used. This easily allows to compile and map the programs onto different memory hierarchies. Experiments #3 and #4 make strongly use of the I/O features of Code Composer (e.g., for image view).

4. EXPERIMENTS

4. 1 EXPERIMENT #1

The goal of experiment #1 is to introduce DSP processors and to become familiar with the C30/C40 assembly language. The students have to do this at home individually. This is a challenging fact, as for most of them it is the first contact with DSP processors. The objective is reached in four steps, starting from scratch, with increasing difficulty. The time available for this experiment is two weeks. All problems deal with audio processing. As an analog front-end for the DSK, the students get an ‘audio box’ with a function generator (sinus, triangle and square signals) and a loudspeaker with an amplifier.

Step 1:
The goal of this step is to become acquainted with the DSK hard- and software and the audio box. The first program (loopaic.dsk) has to be loaded and started with the debugger. The students act as measuring instrument by hearing the different sounds of the wave forms. Intentionally, the volume of the amplifier can not be changed manually and it is set up to be very loud. The first problem is to change the volume of the audio signal by changing the amplifying factor in the code.

Step 2:
In this step, a short code element has to be analyzed. The goal of this step is to start to work with the TI manuals. By examining some assembler instructions, the students learn how to use the manuals.

Step 3:
In this step, programming in assembly language is learned by changing existing programs. The analog signal must be changed in different ways in the time domain. Changes to the signal like amplification, clipping, rectification or squaring are easy to implement because only some additional instructions are needed. Differentiation, integration or amplitude modulation are more difficult to implement because they need at least one storage element.

Step 4:
The last step of the homework is dedicated to FIR – filters. First, the “fir.dsk“ program shipped with the DSK has to be examined. The sampling rate has to be programmed and the effects on the filter have to be described. Then, a filter with 57 coefficients is developed and assessed. Main topics to be investigated are the characteristic of the filter, the connection between sampling rate and filter frequencies as well as an experimental analysis of the computation time by gradually increasing the

sampling rate.

As the equipment does not contain an oscilloscope, the programs have to be tested by analyzing the audible effects of the signal changes (frequency doubling, frequency dependent volume, etc.).

4. 2 EXPERIMENT #2

Experiment #2 is the first of three experiments done in the laboratory under supervision. All these experiments have to be solved within four hours. Experiment #2 deals with a fundamental assessment of a standard FIR – filter in terms of performance and hardware utilization. The C30 EVM and the Code Composer are used as programming environment. A FIR – filter is implemented in assembly language in an optimized and non-optimized way, as well as in ‘C’. The performance of these implementations running on different kinds of memory (internal / external) is assessed. The goal of this experiment is to impressively show the differences in terms of performance.

Step 1:
A FIR – filter with given coefficients is implemented and the sampling rate is set to a given value. The characteristic and cutoff frequencies of the filter are measured.

Step 2:
The execution time of the inner filter loop without I/O is calculated from the assembly code and measured with the profiler. A filter with all useable DSP features like circular addressing, hardware loop, parallel instructions or delayed branches is implemented, as well as one with none of these features. Again, execution cycles are calculated and measured. The different implementations are tested with different mappings to internal and external memory to show the effect of the memory hierarchies on execution time. The maximum possible order of the filter is calculated for each configuration.

Step 3:
The filter from step 2 is implemented in ‘C’. The assembly codes compiled with and without optimization are compared to the implementations of step 2. This step shows the difficulty of using special hardware features by high – level programming languages.

4. 3 EXPERIMENT #3

Experiment #3 deals with more complex DSP tasks in the area of image processing. The image processing algorithms are implemented in 'C' using the Code Composer and the PPDS. The effects of the implemented image processing algorithms, e.g., low-pass filter and edge detection, are easily visualized by the Code Composer. The download of the pictures (black/white, 8 bit) into the memory of the TMS320C40 take place via the function "Data-Load" of the Code Composer. For viewing the results on the screen the function "View-Graph-Image" is used.

Step1:
To get an experience with the developing environment two simple pixel manipulation function are to be implemented by the students.

One image-processing technique is thresholding, in which the points of an image at or near a particular value are highlighted. In a gray-scale image, this highlighting can be done by converting all pixels below a given value to black, and all pixels above that value to white, producing a threshold edge between black and white regions. Then a function for brightness manipulation is programmed by the students.

The execution time for these functions is determined using the profiler of the Code Composer.

Step 2:
Window-based image processing is another technique for analyzing the information of an image. Step 2 focuses on the development of a

$$s'(x,y) = \sum_{l=-\frac{m-1}{2}}^{\frac{m-1}{2}} \sum_{r=-\frac{m-1}{2}}^{\frac{m-1}{2}} s(x-l, y-r) * h(l,r)$$

box-filter which calculates each pixel based on the following equation:
The matrix (mask) size h depends upon the degree of detail in the image but is normally limited by processing speed to quite small (e.g., 3x3 to 5x5) values. The matrix h represents a box which moves over the entire image.

The filtertyp is determined by the coefficients, e.g.,:

$$h = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \qquad h = \frac{1}{8}\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

low-pass sobel

The students are using different coefficients and images to see the effects. Additionally, they have to measure the execution time of the filter-function dependent on the dimension of the matrix size.

4. 4 EXPERIMENT #4

Experiment #4 of our laboratory course deals with parallel processing based on the image processing algorithms developed during experiment #3, i.e., the box-filter algorithm. The box-filter algorithm is parallelized by exploiting its data-parallelism and is implemented on the PPDS using the Code Composer. Thus, the students use the same development environment as in the previous experiment.

The goals of this experiment are (i) to introduce parallel processing using a distributed memory architecture, (ii) to investigate the effect of parallel processing in terms of speedup and efficiency and (iii) to explore the multi-processor capabilities of the TMS320C40 processor. Please note that the PPDS offers shared as well as distributed memory. Due to the limited time for this experiment, we restrict the parallel implementation to a distributed memory architecture.

The experiment is organized into three successive steps.

Step 1:
The data transfer via the communication ports of the TMS320C40 is introduced in the first step, i.e., the image filtering is performed like a remote procedure call using two processors. First, the complete image is sent from the first processor to second processor. Then the box-filter algorithm is executed on the second processor, and finally the filtered image is sent back to the first processor. The data transfer is implemented using the functions out_msg() and in_msg() from the parallel runtime support library (PRTS).

The students have to measure the communication times and to determine the transfer rate as well as the communication to computation ratio.

Step 2:
In this step, the box-filter algorithm is parallelized using two processors. This is simply achieved by dividing the image into two partitions (including an overlap section at the dividing line) and filtering both partitions in parallel. This parallel algorithm is implemented using a master and a slave processor.

The master processor divides the image into two partitions, sends one partition to the slave processor, filters the other partition and receives the filtered partition from the slave processor. The slave processor receives the image partition, filters it and sends the result back to the sender (compare to step 1).

Data transfer is also implemented by DMA using the functions send_msg() and receive_msg() from the PRTS library. Thus, the students can investigate the effect of DMA transfer. The runtime of the parallel implementation is also compared to the single-processor runtime of experiment #3, and the

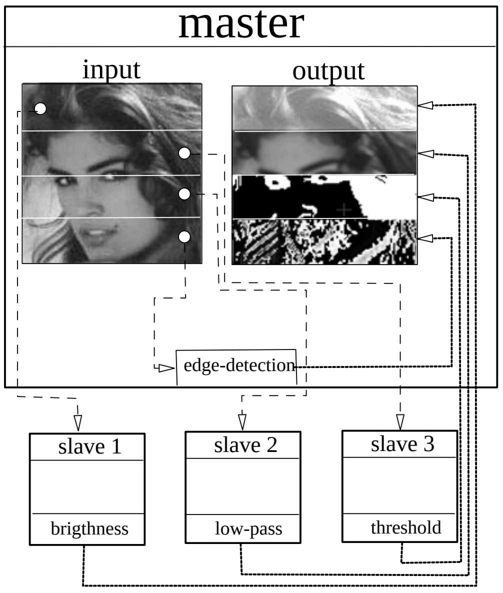

Figure 3*: Image partitioning for parallel image processing on four TMS320C40 DSPs.*

speedup and efficiency of the parallel implementation are determined.

Step 3:
In the last step of this experiment the parallelization strategy of step 2 is generalized to an arbitrary number of processors. Thus, the master processor divides the image into n partitions, sends n-1 partitions to the slave processors, filters 1 partition and receives n-1 filtered partitions from the slave processors. In figure 3 an example with four slaves is shown.

The speedups for 2 to 4 processors are determined and compared to a simple speedup model using the communication to computation ratio. The students can also experiment with the image partitioning (uniform or non-uniform) in order to achieve a high speedup.

5. COURSE FEEDBACK

At the end of the course, a feedback form was filled out by the students. The questions were divided into four sections: (i) prior knowledge of the students, (ii) detailed comments on the experiments, (iii) equipment and tools and (iv) general impression.

The knowledge of the students about DSP in general and DSP processors was low before the course. Thus, special attention will be given to promote additional courses at our university in future.

The results concerning the individual experiments reassured our concept (see figure 4). None of the students rated the demands as heavy. The modal value was “average” for each exercise. Also the time the students had to solve the problems are rated as optimal.

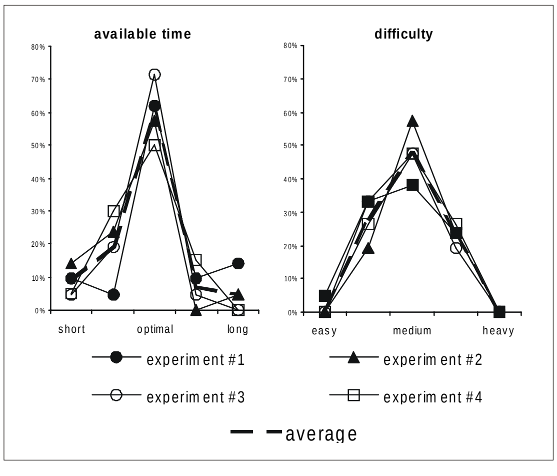

Figure 4: *Survey results concerning:*
- *available time for each experiment and*
- *difficulty of each experiment*

The direct connection between the assembly code and the human ear as a measuring instrument instead of an oscilloscope was a great challenge for the students. The average mark for lab equipment and tools was good. Only those who had difficulties to be a measuring instrument themselves (see experiment #1) wished to have an oscilloscope.

The general impression was very good, 95 % of the students are likely to recommend this course to others.

6. CONCLUSION

In this paper a laboratory course using TMS320C3x and TMX320C4x DSPs is presented. It is intended for graduate students in Electrical Engineering or Telematics. The course structure allows to get deep insight into DSP processor architecture, even starting from scratch. After a lecture part, architectural aspects and performance of the TMS320 DSP processor family are assessed by four experiments with increasing difficulty, ranging from simple audio processing to parallel image processing. The students feedback confirmed the successful implementation of this laboratory course.

ACKNOWLEDGEMENTS

The authors are grateful to Texas Instruments for the donations and, especially, to Robert C.W. Owen and Richard Oed for their support.

REFERENCES

[1] R.C.W. Owen, M.B. Akhan, and I. Munro. *Designing Teaching Material for DSPs*, Texas Instruments: Sixth Annual TM320 Educators Conference, Houston, Texas, USA, 1996

[2] J.H. Saito and M.L. Mucheroni. *ArchMDSP: using C40 for parallel image processing*, Texas Instruments: Sixth Annual TM320 Educators Conference, Houston, Texas, USA, 1996

[3] A.B. Barreto. *A TMS320C50 DSP-based Real-Time DSP Implementation Course*, Texas Instruments: Sixth Annual TM320 Educators Conference, Houston, Texas, USA, 1996

[4] K. Hoover. *TMS320C30 DSP Laboratory Course Taken Concurrently with a DSP Theory Course*, Texas Instruments: Sixth Annual TM320 Educators Conference, Houston, Texas, USA, 1996

[5] F.J. Taylor. *SPECtra: A Signal Processing Engineering Curriculum*, IEEE Transactions on Education, vol. 39, pp. 180-185, May 1996.