T9000 - A Preliminary Evaluation of Arithmetic Performance

G. Bader* and B. Przywara

Interdisziplinäres Zentrum für Wissenschaftliches Rechnen
Universität Heidelberg
Im Neuenheimer Feld 368
D-6900 Heidelberg
t9@iwr1.iwr.Uni-Heidelberg.de

May 5, 1993

Abstract

This brief note presents a first assessment of the arithmetic performance of the new transputer processors T9000. The presented results are obtained on a T9000β version of the processor running with reduced clock speed of 10 MHz. Due to the still reduced functionality of the available processor we consider only computational kernels of BLAS type. Nevertheless, we hope that these preliminary tests provide some idea of the computing power of the T9000 processor for real applications.

Keywords: T9000β-Transputer, BLAS Routines.

*present address: Institut für Mathematik, Technische Universität Cottbus, Karl-Marx-Str. 17, O-7500 Cottbus

1 Introduction

Distributed memory parallel computers based on transputers of the T800 family have been available for 8 years now. They have found - mostly in the academic community - high acceptance. The success of this technology can be contributed to a large extend to the favorable balance between computational performance and communication speed. This allows in particular the efficient use of such systems with large numbers of processors and the implementation of algorithms with fine granularity.

Meanwhile T800 systems do no longer satisfy the need for continuously growing computing power. The new processor T9000 has been announced to provide an approximately 10 fold increase in computing power and, what is equally important, communication bandwidth. This has let, long before the silicon was available, to expectations about the actual performance of the T9000 processor for practical applications. With this note we hope to provide an early assessment of the potential performance of this new transputer processor.

The now available version T9000β of the processor has still limited functionality and is running with a reduced clock speed of only 10 MHz. At present it is able to reach a peak performance of 40 Mips and 3.33 MFlops. The final version of the T9000 transputer was announced to run with a clock speed of 50 MHz. It should provide a peak performance of 200 Mips and 25 MFlops. The crucial question for advanced processor technologies is, to what extend this peak performance can be put to work for practical applications.

The present paper documents some performance tests done for the the β version of the T9000 processor. We hope that these results are a contribution towards the assessment of this processor. We also try to provide, on the basis of measured results, an estimate for the performance of the final 50 MHz version of the processor.

2 Microprocessor architecture

The main emphasis of this section is to provide a brief overview of the architecture of the T9000 processor as far as they are relevant for understanding the performance tests carried out. A presentation of the details can be found in a number of technical publications, see for example [3, 4, 5]. We follow closely the presentations [6, 10].

The T9000 comprises a superscalar 32-bit processor (CPU) with a 64-bit floating point unit (FPU), a communication processor (VCP) together with 4 communication links, a control unit and its associated links, an external memory interface (EMI) and 16 KBytes of on-chip memory (Cache). The final component of the T9000 is a crossbar switch (X-Bar) which connects the other components.

The T9000 operates at 50 MHz approximately 10 times as fast as a T805 processor with 30 MHz. Thus, it issues approximately 4 times as many memory accesses per cycle as the T805. The 4 communication links provide a maximum bandwidth of 70 MBytes/sec as compared to 11.2 MBytes/sec for the T805. This implies that the T9000 communication system makes 2.5 times as many data accesses per cycle.

In order to reduce the requirements in terms of bandwidth on the memory system the T9000 uses two levels of caching. The first level is within the processor. It provides caching of data in the workspace of the currently executed process with a capacity of 32 words. As a high percentage of access made by the processor is for data in the workspace this has a significant effect. The second level of caching is located between the major functional units and the external memory. Ultimately, however, access must be made to external memory. The memory interface of the T9000 provides a 64-bit wide memory access, giving a maximum of 200 MByte/sec.

The T9000 has 16 KBytes of on-chip memory. This RAM can be configured as 16 or 8 KBytes of cache. The latter allows to lock code and data into fast on-chip memory for time critical processes. The cache allows that external memory accesses are mostly made for complete cache lines. They are chosen to be 4 words long, e.g. 4 single or 2 double precision data items. This reduces effectively the requirements for external memory bandwidth. Note, however, that the processor can also be used without external memory. It operates then directly on the fast on-chip memory without cache.

Even if caching reduces the requirements for external memory bandwidth there remains the necessity to provide processor and VCP with sufficient bandwidth to the cache. The internal memory is divided into 4 banks, each caching a quarter of the address space. When an address is presented to the memory system the request is routed through the crossbar to the appropriate memory bank. There exist 9 ports onto the crossbar, 1 for the EMI, 4 for the processor, 3 for the VCP and 1 shared between the scheduler and the control unit. Each bank of the cache can provide access to one word every cycle and thus the total on-chip memory bandwidth is 800 MByte/sec. Figure 1 provides an overview of the main building blocks of the processor and their interconnection.

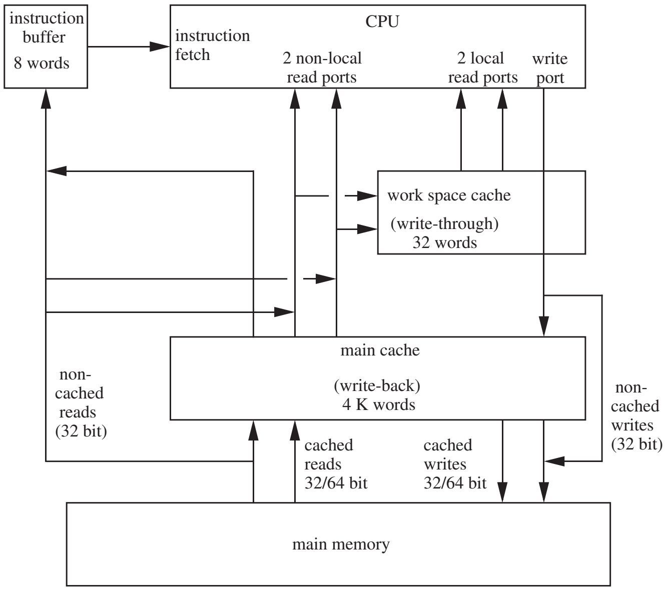

Figure 1: T9000 processor architecture

The T9000 executes most of the instructions in a single cycle. This is in contrast to T800 transputers. Especially most of the slower instructions of the T800 have been significantly improved:

cycles	T800	T9000	
fpadd	7	2	(double precision, 64 bit)
fpmul	20	3	(double precision, 64 bit)
mul	36	$2-5$	
shl n	n	1	

Already the T800 processor allows for the integer and floating point units to operate in parallel. This is known as a superscalar architecture. The T9000 processor takes this kind of concurrency much further. Important classes of instructions, such as address calculations, the loading of integer or floating point numbers, the operations on these and the final storing are done in separate processor units. Actually, some of the most frequent operations are done on multiple functional units:

stages	instruction class	units	instructions
1	local memory read	2	e.g. **ldl**, **ldlp**, **ldc**, ...
2	address calculations	2	e.g. **ldnlp**, **wsub**, ...
3	non-local memory read	2	e.g. **ldnlp**, **wsub**, ...
4	int or fpu operations	1	e.g. **add**, **fpadd**, ...
5	memory write	1	e.g. **stl**, **stnl**, **fpstnl**, ...
6	jump	1	e.g. **eqc 0**, **cj**, **j**, ...

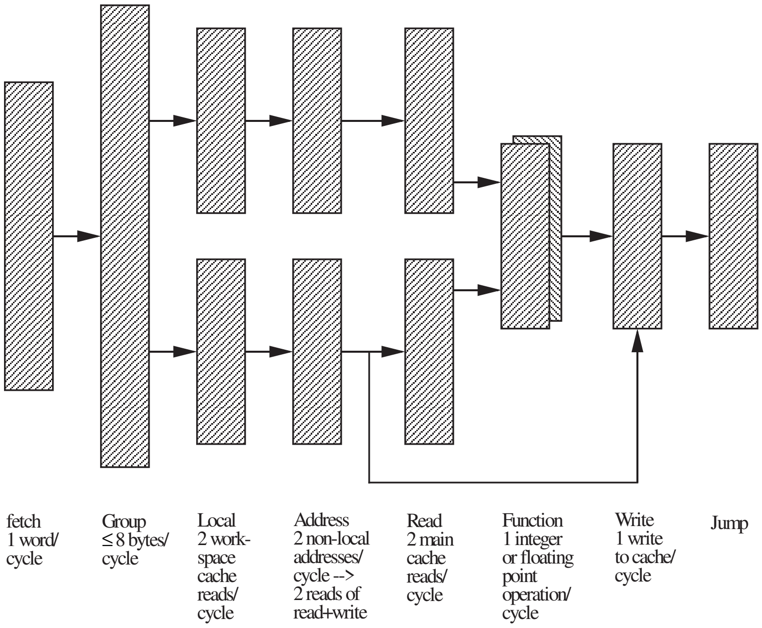

Figure 2: T9000 processor pipeline

The processor units are organized as a pipeline, e.g. figure 2. In order to maximize the throughput of the processor pipeline, the processor is equipped with a hardware implemented grouping mechanism. It takes the incoming stream of instructions, builds groups of instructions, and feeds those into the pipeline with a rate of one group per cycle.

3 Evaluation of computational kernels

In order to make the grouping mechanism and the implications for arithmetic kernels more transparent we consider two examples of BLAS Level 1 routines in detail. See [8] for the definition of these **B**asic **L**inear **A**lgebra **S**ubroutines. The first example is the calculation of scalar products. We limit our considerations to the single and double precision versions of sdot and ddot respectively. The actual coding of these routines is done with loop unrolling to reduce the loop overhead. Table 1 presents a typical assembly language segment for sdot. Note that the rules in table 1 indicate the groups formed in the processor pipeline. For the corresponding double precision version ddot the single precision assembler instructions fpldnlsn and fpldnlmulsn have to be replaced by its double precision instructions fpldnldb and fpldnlmuldb, respectively.

1	ldl	x	6 instructions
2	ldnlp	2	8 bytes
3	fpldnlsn		
4	ldl	y	
5	ldnlp	2	
6	fpldnlmulsn		
1	fpadd		1 instruction
			2 bytes

Table 1: assembler sequence sdot

Based on this assembler sequence we find that 6 processor cycles are necessary within the pipeline to evaluate the two groups on the T9000. On the available 10 MHz version of the processor we can reach 3.33 MFlops. This is actually the peak performance of this processor, c.f. [6]. Note that this estimate is based on the evaluation of many sequences of the form given in table 1. Hence, it is assumed that the pipelining effect leads to a partial overlapping of the sequences. For this reason we call such estimates theoretical performance. It does not contain any cycles necessary for loop overhead and for potential bank conflicts which are likely to occur when loading the data. Thus, the measured performance results are bound to be slightly lower than these numbers suggest. For the corresponding double precision routine ddot we find 7 cycles for the evaluation of the two groups and 2.86 MFlops of theoretical performance.

The second example is the BLAS routine saxpy. A typical assembler language sequence is provided in table 2. For the single precision routine saxpy we need 6 cycles for the evaluation of the two groups of table 2 giving a theoretical performance of 3.33 MFlops. Note, that saxpy needs 2 load and 1 store operations more than sdot. This leads inevitably to memory bank conflicts. Consequently, the measured results, c.f. table 3, are bound to be lower than for sdot.

1	ldlp	a	5 instructions
2	fpldnlsn		7 bytes
3	ldl	x	
4	ldnlp	2	
5	fpldnlmulsn		
1	ldl	y	6 instruction
2	ldnlp	2	8 bytes
3	fpldnladdsn		
4	ldl	y	
5	ldnlp	2	
6	fpstnlsn		

Table 2: assembler sequence saxpy

The double precision version needs 8 cycles to evaluate and gives a theoretical performance of 2.5 MFlops. For this routine we have to load 3 and store 1 double precision data. This takes 8 processor cycles for data movement within the processor. The actual computation, 1 multiply and 1 add, takes only 5 cycles for completion. Thus, in this case a saturation of the internal memory bandwidth takes place. The limiting factor for the performance is data movement and not computational speed. Similar effects show for a number of double precision BLAS level 1 routines. Note, however, that this is also known for other superscalar architectures and in particular for vector processors. The situation improves for BLAS level 2 and 3 routines.

4 Measured performance results

Before presenting the results obtained we outline briefly the available test environment. The hardware consists of a Parsytec GC-cube with a test board and a single T9000β processor. The test board also carries a T805 processor. It is used to load the T9000 processor with executable code and data via its dual ported RAM. The T9000β processor is running under a reduced clock speed of 10 MHz. At present the processor has a peak performance of 40 Mips and 3.33 MFlops. Finally, the board is equipped with 4 MByte of external memory.

The software environment at the present state is the INMOS C-compiler, [2], and a preliminary PARIX-T9000 version from Parsytec. The PARIX environment is called PHOENIX, see [9]. It allows to work with FORTRAN and C. Both systems are so called cross systems, e.g. the source code is compiled on the host computer. The executable code is loaded onto the transputer and executed by a run time system.

4.1 On-chip results

As pointed out in section 2, the T9000 can run without external memory. In this mode of operation the cache is not needed. This implies that the 16 KBytes of on-chip RAM are accessed without using the cache. As not all functions of the cache are yet fully operational we have chosen to evaluate the processor performance in this mode. The presented results should give an good idea of what percentage of the 3.33 MFlops of peak performance show up in real computations.

We have tested a selected number of BLAS level 1 routines with the INMOS C-compiler. Both source code (src) and optimized assembler (asm) coding are considered. The obtained results are compiled in table 3. Depending on the various routines, the results show good performance. This is in particular true for almost all single precision routines. For some double precision routines the limitation by the internal memory bandwidth can be observed.

function	MFlops	
	src	asm
daxpy	1.633	1.896
saxpy	2.254	3.030
ddot	2.353	2.694
sdot	2.623	3.101
dscal	1.127	1.481
sscal	2.116	2.367
idamax	0.279	1.227
isamax	0.366	1.365

Table 3: Performance results T9000

Now consider the Gauss-Seidel iteration for a standard five point discretization of the Laplace equation in 2 dimensions. The algorithm may serves as a simple model for computational fluid dynamic problems. Recall that for a N by N regular grid this algorithm solves iteratively a linear system of dimension $N * N$ with a very sparse coefficient matrix. In terms of processing this implies that large arrays of data are recursively touched while only a moderate amount of computations is done. Thus, similar to BLAS level 1 routines, the effective bandwidth for data movement within the processor plays an important role for the observed performance. A typical result for the T9000 is shown in table 4. Note, for comparison, the T800 processor running at 25 MHz performs for this algorithm with 0.556 MFlops.

Gauss-Seidel	MFlops	
	src	asm
$N = 16$	1.150	1.586

Table 4: Performance for Gauss-Seidel iteration

4.2 Results with cache and external memory

In any large size real computations the data have to be stored in external memory. Thus, it is of prime importance to test the behavior of the processor under such real life assumptions. Any external memory system has typically less bandwidth than internal memory. Hence, we have to expect a certain reduction of the processor performance when used with external memory. The question is to what extend this reduction shows up. The answer is obviously problem dependent. In this respect BLAS level 1 routines are a difficult test for any processor with an internal cache.

function	MFlops	
	no Cache	Cache
dsum	0.900	1.424
ssum	1.804	2.378
dasum	0.726	1.421
sasum	1.329	2.273
daxpy	0.601	0.770
saxpy	1.629	2.045
ddot	0.824	1.225
sdot	1.521	2.040

Table 5: Performance results with caching

As the cache on the present version of the T9000 is not fully functional we have done only a preliminary testing so far. The results have to be interpreted with caution as it is not always clear to what extend the limited performance is due to the reduced functionality of the cache. In particular, the port between cache and external memory is only 32-bit wide. In the final version of the processor this will be extended to a 64-bit wide port. This yields a doubling of the corresponding bandwidth.

In order to be able to compare the results with and without cache we have selected similar routines as in subsection 3.1. The obtained results are presented in table 5. When using external memory together with the cache the performance is considerably lower than the on-chip results, c.f. table 3.

This is, to a large extend, due do the presently limited external memory bandwidth. In general, however, it is difficult to give a reliable judgment of these results at present. This has to wait till all components of the processor operate as they should.

Finally, we list the results for the Gauss-Seidel iteration. Iterative techniques are typically applied for large systems. Hence, the effectiveness of data movement between processor and external memory is of very high importance. The corresponding results are shown in table 6.

Gauss-Seidel	MFlops	
	no Cache	Cache
$N = 16$	0.544	1.396

Table 6: Performance for Gauss-Seidel iteration

Summary

Here we would like to give a brief summary and judgment of our, still limited, experiences with the new Transputer processor T9000β. Many more tests have to be carried through before the performance of this processor as a building block for massively parallel systems can be reliably evaluated. However, before this can be done the processor has yet to reach its full functionality. In particular, very little can be said at present on the processors balance between communication speed and computational performance. Nevertheless, some conclusions can be drawn from our testing.

Lets first discuss the performance when the processor is running without external memory, i.e. without cache. The presented results prove that especially for single precision calculations the achieved performance is very close to the peak performance of the processor. For double precision data the results are, depending on the algorithm, to some extend limited by the internal memory bandwidth. Still, the observed performance is reasonable close to the peak performance of the processor. On the basis of the measured performance results a linear speedup can be expected when the processor is running with higher clock speed. Thus, we guess a five fold increase of the presented on-chip numbers for the 50 MHz version of the T9000. In comparison to the T805 this would be a factor 8 to 10 of performance gain.

When considering the results with cache and data stored in external memory the present situation is not quite as satisfactory. Especially, the presently limited cache to external memory bandwidth reduces the observed performance considerably. This should, however, improve substantially as soon as the memory interface becomes 64-bit wide. This and the not yet full functionality of the cache makes it more difficult to extrapolate the observed results for the 50 MHz version of the processor.

However, applying a rather conservative estimate, it is reasonable to expect a performance gain of a factor 5 to 8 for the 50 MHz T9000 when compared to the T805 processor. On the same basis one might safely guess the absolute performance in the range between 6 and 10 MFlops for double precision computations.

Finally, it should be mentioned that the superscalar architecture of the T9000 processor is less sensitive to the optimization quality of code produced by the compiler than the T800 family of processors. The performance reduction involved for less careful coded algorithms is rather moderate. On the other hand, writing an optimizing compiler might not be an easy task.

References

[1] G. Bader and B. Przywara, *Arbeitsumgebungen und Leistung von Transputersystemen*, in: Bader/Rannacher/Wittum eds., *Numerische Algorithmen auf Transputer-Systemen*, Teubner Skripten zur Numerik, 1992.

[2] INMOS, *ANSI C Toolset User Manual*, Inmos 1990.

[3] INMOS: *The T9000 Transputer Instruction Set Manual*, Preliminary Information, Inmos 1992

[4] INMOS: *The T9000 Transputer Hardware Reference Manual*, Preliminary Information, Inmos 1992

[5] INMOS: *The T9000 Transputer Products Overview Manual*, INMOS Databook Series, Inmos 1991.

[6] F. Lücking, *T9000: Microprocessor Architecture*, Parallel Link, Special Issue Launch T9/CeBIT, Parsytec, 1993.

[7] F. Lücking: *Efficient BLAS level 1 Library for Transputers*, Parsytec GmbH, Aachen 1990.

[8] C. Lawson, R. Hansen, D. Kincaid and F. Krough : *Basic linear algebra subprograms for FORTRAN usage*, ACM Trans. Math. Software 5, 308-1371, 1979.

[9] Parsytec Eastern Europe, *PHOENIX PARIX, Release 0.9*, Chemnitz 1993.

[10] R. Shepherd: *Next Generation Transputers and Beyond - 2, T9000: superscalar transputer*, Aug. 1991.