The T9000 Transputer

David May, Roger Shepherd and Peter Thompson
Inmos Limited
1000 Aztec West
Almondsbury
Bristol BS12 4SQ, UK

Abstract

The T9000 transputer integrates a complete computer in a single VLSI chip of over 2 million transistors. It contains 3 major subsysytems: a pipelined superscalar processor, a communications processor and a 16kbyte fully associative cache. In this paper we discuss some of the issues arising in the design of the T9000.

1 Introduction

A transputer is a VLSI component, consisting of a processor, a memory and a communication system, designed to be used in the construction of parallel systems. INMOS introduced the first transputer, the T414, in 1985. Since then a range of compatible 32-bit and 16-bit transputers have been introduced and these have been used in a wide range of applications.

The design of a transputer is undertaken on the assumption that it is a component of a multiprocessor system[4]. Many of the design tradeoffs, therefore, differ from those made by the designers of conventional microprocessors. These include devoting a significant proportion of the silicon area to of a communication system[1] and choosing an instruction set architecture tuned to the execution of concurrent programs, rather than sequential programs.

The T9000 is a new transputer[1]. It represents an improvement on the existing generation of transputer products in both capability and performance. The T9000 extends the transputer architecture in a number of ways. The most important of these is that the T9000 transputer decouples the physical connectivity of a system from its logical connectivity. Between any two directly connected T9000 transputers there may be established an almost unlimited number of *virtual channels*. The T9000 link system also enables transputers to be connected via a network of C104 packet routers which allows virtual channels to be established from any transputer to any number of other transputers. Other extensions of the architecture include the enhancement of the process model to provide per-process error handling facilities and the ability to run programs under memory management.

The T9000 has about ten times the performance of a T805. This improvement derives from a variety of sources including the use of caching, improvements in semiconductor technology, and a highly pipelined, superscalar processor.

2 Overview

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

Figure 1 shows the structure of the T9000.

3 Interconnection and memory structure

The key to the T9000 transputer's performance is in the memory architecture and internal interconnection structure of the device. This is important because of the enormous memory bandwidth that the

[1] 25% of silicon area for the T414, 16% for the T9000

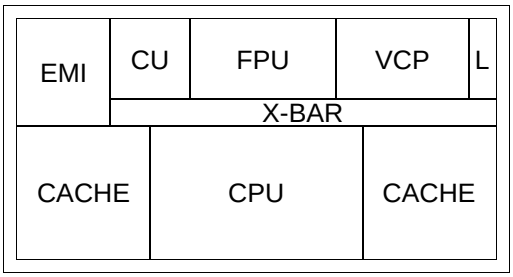

Figure 1: Block structure of the T9000

T9000's subsystems demand. The majority of the bandwidth is demanded by the processor and the communication system. The T9000 processor, running at 50 Mhz, executes at about 10 times the speed of a T805 processor running at 20 Mhz. This means that the T9000 processor makes 4 times as many memory accesses per cycle as the T805 processor. A similar situation exists with the communication system. The four links of the T805 provide a maximum bi-directional data bandwidth of 11.2 Mbyte/s, whereas the four links of the T9000 provide a maximum bi-directional data bandwidth of 70 Mbyte/s. This means that the T9000 communciation system makes 2.5 times as many data accesses per cycle. In fact, even more bandwidth than this is required as the memory holds data structures which support the virtual channel system.

Fortunately, a large proportion of the memory accesses made within a transputer are very structured and thus caching can be used to reduce the demand on the external memory system. The T9000 uses two levels of caching. The first level exists within the processor and provides caching of data in the workspace of the currently executing process. This has a significant effect as typically over 1/3rd of all access made by the processor are to the workspace. The second level of caching sits between the major functional blocks and the external memory system. Ultimately, however, accesses must be made to external memory and the memory interface of the T9000 provides 64-bit wide access, giving a maximum data rate from external memory of 200 Mbyte/s.

Even though caching reduces the requirement for external memory bandwith there remains the matter of providing the processor and VCP with sufficient bandwidth to the cache. The T9000 adopts the classic solution of multibanking the memory system. The internal memory is divided into four banks, each bank caching one quarter of the address space. When an address is presented to the memory system the access request is routed through the cross-bar to the appropriate bank of memory. There are 9 ports onto the cross-bar, 1 for the PMI, 4 for the processor, 3 for the VCP and 1 shared between the scheduler and 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/s.

4 Cache

The T9000 has 16k bytes of on-chip cache. The cache is write-back and allocates when writes miss the cache. One advantage of this structure is that all accesses made to external memory are of complete cache lines. This removes the need to provide mechanisms to access individual bytes of external memory, even though the machine is byte addressed. (In fact, this is only partly true as it is necessary to provide for access to uncached regions of memory).

The cache is organised as 4 banks, each containing 4k bytes of fully associative memory arranged as 256 lines of 4 words. The banks are interleaved on address bits 4 and 5. This may seem strange as the obvious choice would be to interleave on bits 2 and 3. However, that choice is incompatible with having multi-word cache lines containing contiguous addresses. The choice of four word cache lines represents a tradeoff between minimising the external bandwidth generated by cache misses[2], maximising the hit-rate[3], and choosing an implementation architecture which is consistant with the chip area available. The practical choices for the T9000 were 2 words-per-line or 4 words-per-line. With a 64-bit wide memory interface the extra external bandwith of a 4 words-per-line cache, as compared with a 2 words-per-line cache is acceptable. The lower overhead (tag store) of the 4 words-per-line cache is attractive; if is probable that if a 2 word-per-line cache had been used then only 8k bytes of cache could have been implemented and this would have nullified any performance benefits that might otherwise have derived from the smaller line size.

Each bank of the cache is fully associative. Every 4 word line has associated with it a CAM which

[2] a smaller line typically generates lower external bandwidth

[3] both a larger line and a larger cache typically generate a higher hit rate

contains the 26 undetermined bits of address (2 bits are determined by byte within word, 2 bits by word within line and 2 bits by bank within cache). There are a number of benefits from using a fully associative cache as compared with a direct mapped or set associative cache. The first is that any address may be mapped into any line of the cache and so the cache hit-rate does not collapse when operating on data structures which "beat" with the cache size. A second advantage of using a CAM is that it is more power-efficient than a set associative cache. When a read is made from an N-way set associative cache, N lines are accessed, and for each line one word and the tag are read; the word from the line with the matching tag is then chosen. When a read is made from a fully associative cache only the single matching word need be read; this represents approximately a factor of N saving in power consumption. A further advantage of using a CAM is that a write can be made to the cache in a single cycle, rather than requiring first an access to the tag store and then a write to the data store. A novel feature of the the T9000 cache, which is made possible by the use of a CAM, is that the one line is always kept empty, to be allocated when the next miss occurs. In this way the latency caused by a miss can be reduced as it is not necessary to first write back a (dirty) line before the cache line can be filled. The holding of this spare line is possible because any line can be used to cache any address. In an N-way set associative cache a similar mechanism would require keeping 1/N'th of the cache empty.

Thus far the internal memory has been described as a cache, however, it is possible to configure the memory so that it operates as pure RAM or as a mixture of 50% RAM, 50% cache. This is a useful facility for a number of purposes. One is that the T9000 can be used as a stand-alone computer with no external memory attached, a second is that it is possible to lock key areas of code and data into memory, either with a view to increasing performance, or with a view to improving predicability of execution time.

5 Processor

The T9000 was designed to provide a very fast implementation of an existing instruction set. Although the T9000 extends this instruction set, both to provide enhancements to the process model and to improve performance, to a first approximation the instruction set of the T9000 is the same as the instruction set of the T805. Each instruction consists of a single byte arranged as a 4-bit function and a four-bit operand. The instructions are chosen to achieve compact program representation, although a relatively large number of instructions are used.

The most important part of a fast implementation of the transputer instruction set is the provision of very fast access to local variables. In the transputer these reside in memory, whereas in a register machine a compiler attempts to keep them in registers. This puts the transputer at a potential disadvantage in two ways. The first is that access time to memory is slower than to registers, the second is that the transputer may require more memory bandwidth than a register machine.

These problems are overcome in the T9000 processor by using a workspace cache which can cache the first 32 locations of the workspace. This cache can be accessed essentially as fast as registers in a register machine, and can be used to supply data for all instructions which access the first 32 words of the workspace. The cache is operated as a write-through cache, that is, whenever a write is made to the workspace the write is made both to the workspace cache and to memory. In this way memory always contains a correct image of the workspace.

Another issue in implementing a fast transputer is achieving a fast execution rate. The conventional approach to speeding the execution of instructions is to use pipelining. However, the semantic content of many transputer instructions is low. This makes pipelining difficult as there is not enough work to do in these transputer instructions to warrent execution on a pipeline. To overcome this the T9000 *groups* several dependant instructions together and then executes the resulting group on a pipeline. In this way the T9000 can execute up to 8 instructions per cycle on a pipeline.

The T9000 pipeline has seven readily identifiable stages:

- The *fetch* and *decode/group* stages of the pipeline prefetch and then group instructions. The groups are then executed on the remaining 5 stages of the pipeline.
- The *local* stage is capable of performing two *load local* or *load constant* instructions

each cycle. As the workspace cache is triple ported, with two read and one write port, it is able to support two *load local* instructions executing at the same time. Note that a *load local* instruction which references one of the first 32 locations of the workspace can be executed without having to compute the address of the workspace location; the operand of the instruction can be used to directly address the workspace cache.

- The *address* stage can perform two three-operand address calculations per cycle. These addresses may be used either in the *non-local* stage to read from memory, or in the *write* stage to write to memory. Note that for *store local* (unlike for *load local*) the address of the workspace location referenced has to be computed since a store will be made to memory as well as to the workspace cache.

- The *non-local* stage allows up to two reads at a time to be made from memory. Provided that the data reside in different banks of the cache the two reads can be made in the same cycle. No attempt is made to service these reads from the workspace cache; there is no need as the memory is kept consistent with the cache.

- The *ALU/FPU* stage is where arithmetic and floating point operations are performed. This stage contains hardware to perform shifts in 1 cycle, multiplication in 2 to 5 cycles, and normalisation in 2 or 3 cycles. The FPU is capable of performing single precision addition and multiplication in 2 cycles, and double precision multiplication in 3 cycles[3].

- Finally, writes are made in the *write* stage of the pipeline. Typically, the address will have been computed earlier, in the address stage, and the value to be stored will have been computed in the ALU/FPU stage. All addresses written to are checked to see whether they lie within the workspace cache, and if they do the workspace cache is updated, ensuring that it remains consistant with the memory.

The T9000 processor uses pipeline parallelism in a fat (multiple execution unit) pipeline to achieve fast execution of the transputer instruction set. The maximum size of group the pipeline can accept is determined by the instruction fetcher and grouper examining upto 8 bytes at once. The pipeline can accept 1 group per cycle (peak), which gives a maximum execution rate of 8 instructions per cycle. However, as the T9000 can fetch only 4 bytes per cycle, the peak sustainable instruction execution rate of the T9000 is only 4 instructions per cycle (200 MIPS).

6 Communication System

The external communications of the T9000 are managed by a communications processor called the VCP. The main function of this subsystem is to accept high-level communications commands from the processor and translate them into sequences of packet exchanges on the serial links obeying a strict protocol.

The VCP multiplexes simultaneous communications on an arbitrary number of virtual channels. It does this by keeping information relating to each virtual channel in a data structure in memory called a VLCB. Each time a packet relating to a particular channel is to be sent, the VCP adds the corresponding VLCB to a linked list. As each VLCB is taken from the head of a list, the VCP sets up a DMA transfer directly from the workspace of the communicating process to one of the four (100 Mbit/s, full-duplex) serial links. Every packet starts with a header taken from the VLCB and contains at most 32 bytes of data, giving a effective bi-directional data rate of upto 17.6 Mbytes/s per serial link. Accesses to the VLCBs require a further memory bandwidth of around 20 Mbytes/s, giving a total bandwidth requirement of the order of 100 Mbytes/s which is met by three ports on to the cross-bar.

The VCP consists mainly of two DMA controllers and three large state machines, implemented as microcoded datapaths. The first state machine accepts high-level commands from the CPU and performs corresponding operations on the VLCBs, adding them to linked lists as required. The second state machine removes VLCBs from the linked lists and programs the output DMA controller to transmit the required packets. The third state machine decodes incoming packets, programs the input DMA controller to store data directly into the workspaces of the communicating processes, updates the corresponding

VLCBs and puts them back on linked lists if more packets need to be sent. The behaviour of each microcoded machine is highly conditional, and so the microcode ROMs were designed so that conditions stable as little as 6 ns before the end of a cycle determine the state on the next cycle.

The three main state machines operate concurrently and asynchronously, to deliver high sustained performance when many channels are active at once. This necessitates several interlocks to ensure consistent behaviour; for example, the first packet of a message may arrive either before, after, or simultaneously with the command from the CPU which determines the correct destination for the message. To prevent unnecessary performance degradation, most interlocks operate on a per-channel basis, using the virtual channel number as input to a comparator.

The input DMA controller includes a cache-line buffer. Whenever this buffer is completely filled by data in an incoming packet, the DMA controller makes a special access to the cache, so that if a miss occurs the line is allocated in the cache but not filled from external memory, thus saving external memory bandwidth. The output DMA controller is pipelined in three stages, which are parameter set-up, packet transmission and possible rescheduling of the process sending the message. This pipelining ensures that packets are sent continuously as long as there is a non-empty queue.

The four serial links, called DS-Links, transmit at a finely-programmable speed upto 100 Mbits/s using a novel two-wire encoding which requires a maximum frequency of only 50 Mhz on each wire. One wire carries the digital signal and the other wire (the 'strobe') changes state only when the data does not. In this way only one wire is changing state at a given moment. This encoding is decoded with self-timed logic which has only to discriminate the order in which edges occur on the two wires, giving a whole bit-time of skew tolerance. The self-timed decoding circuitry enables a DS-Link to receive data at any rate, regardless of its transmission speed. DS-Links transmit each byte of data as a 10-bit sequence, which includes a parity check bit and a flag to distinguish control sequences. These control sequences are used to deliniate the ends of packets, and to implement a strict byte-level flow-control mechanism which ensures that data is never overwritten.

7 Summary

The T9000 transputer has high performance processor. This has been achieved by extensive use of caching and a novel processor implementation. The processor uses a fat pipeline and dispatches several dependent instructions into the pipeline each cycle. The resulting processor is able to saturate a 25 MFLOP floating point unit. The processor is supported by a communications system which supports communications at high speed and low latency. The close integration of the processor, communications system and cache takes full advantage of the capabilities of VLSI technology, enabling a high degree of concurrent operation.

8 Acknowledgements

The development of the T9000 transputer was supported by the CEC within the ESPRIT programme.

References

[1] INMOS Limited, *The T9000 transputer products overview manual*, INMOS Limited, 1991

[2] INMOS Limited, *Transputer Instruction Set - a compiler writer's guide*, Prentice Hall, 1988

[3] Simon Knowles, *Arithmetic Processor Design for the T9000 Transputer*, ASPAAI-2, SPIE San Diego, July 1991

[4] D May, R Shepherd: *The Transputer Implementation of occam*, proceedings of the Second International Conference on Fifth Generation Computers, Tokyo 1985