inmos

T9000 - superscalar transputer

Richard Forsyth
Bob Krysiak
Roger Shepherd

INMOS Limited - SGS-Thomson Microelectronics

SGS-THOMSON

inmos

Transputer - background

complete computer on a chip
- processor
- memory
- communication

conceived as component of a multiprocessor
- easy to connect
- support for on-chip multiprocessing/communciation

SGS-THOMSON

inmos

T9000: A better transputer

1 FASTER:
- 50 Mhz
- 10x T805 processing whilst binary compatible
- 5x T805 communication bit rate

2 MORE CAPABLE:
- multiple channels per inter-processor link
- channel routing (with C104 packet router)
- protection and memory management

3 MORE MEMORY:
- 16 kbyte fully associative cache/internal RAM

SGS-THOMSON

3

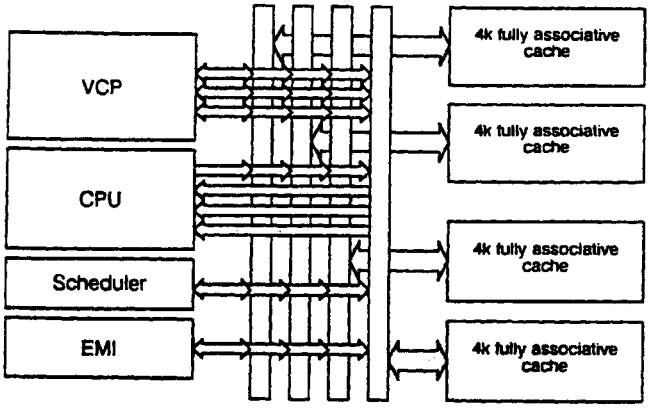
inmos

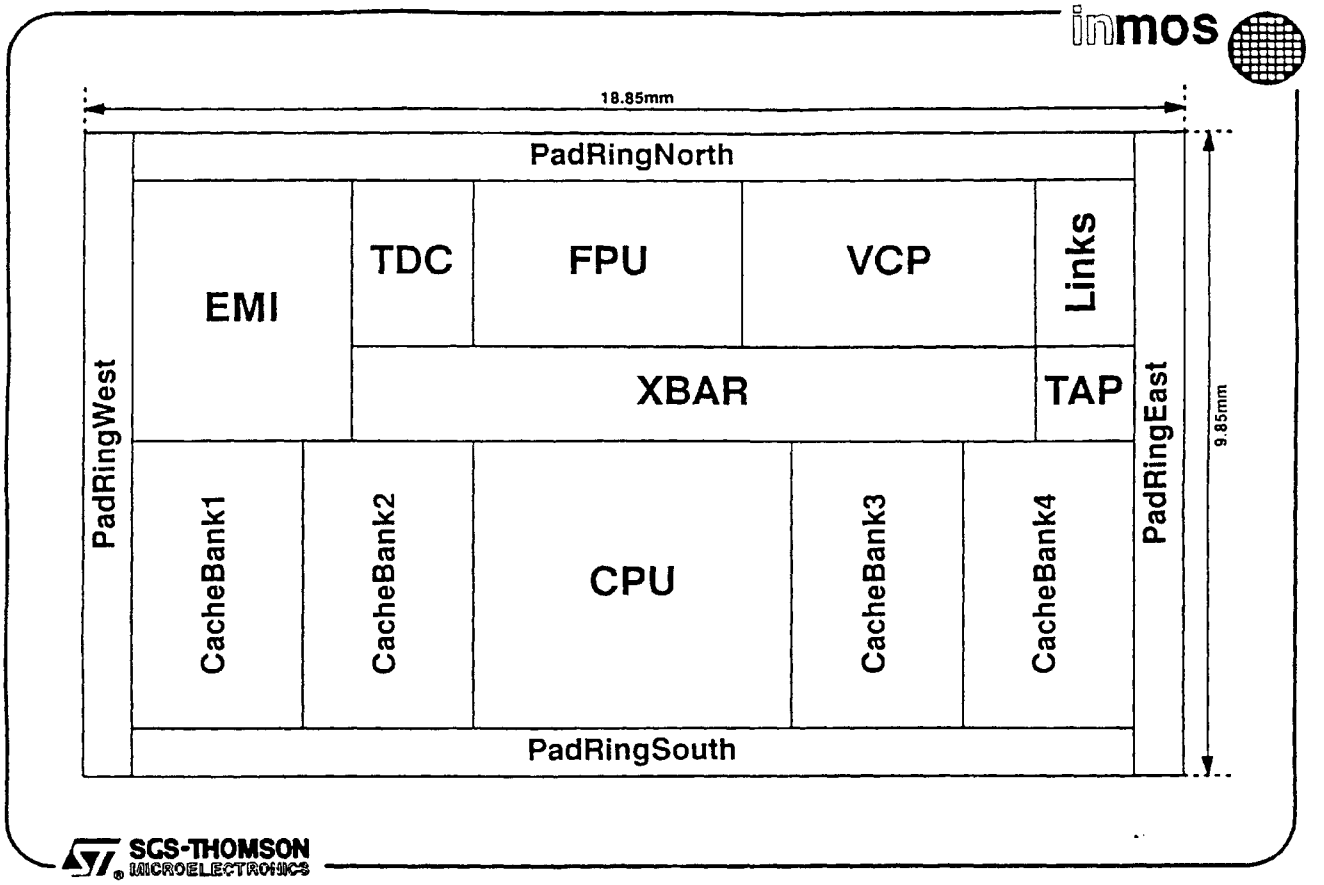
T9000 chip overview

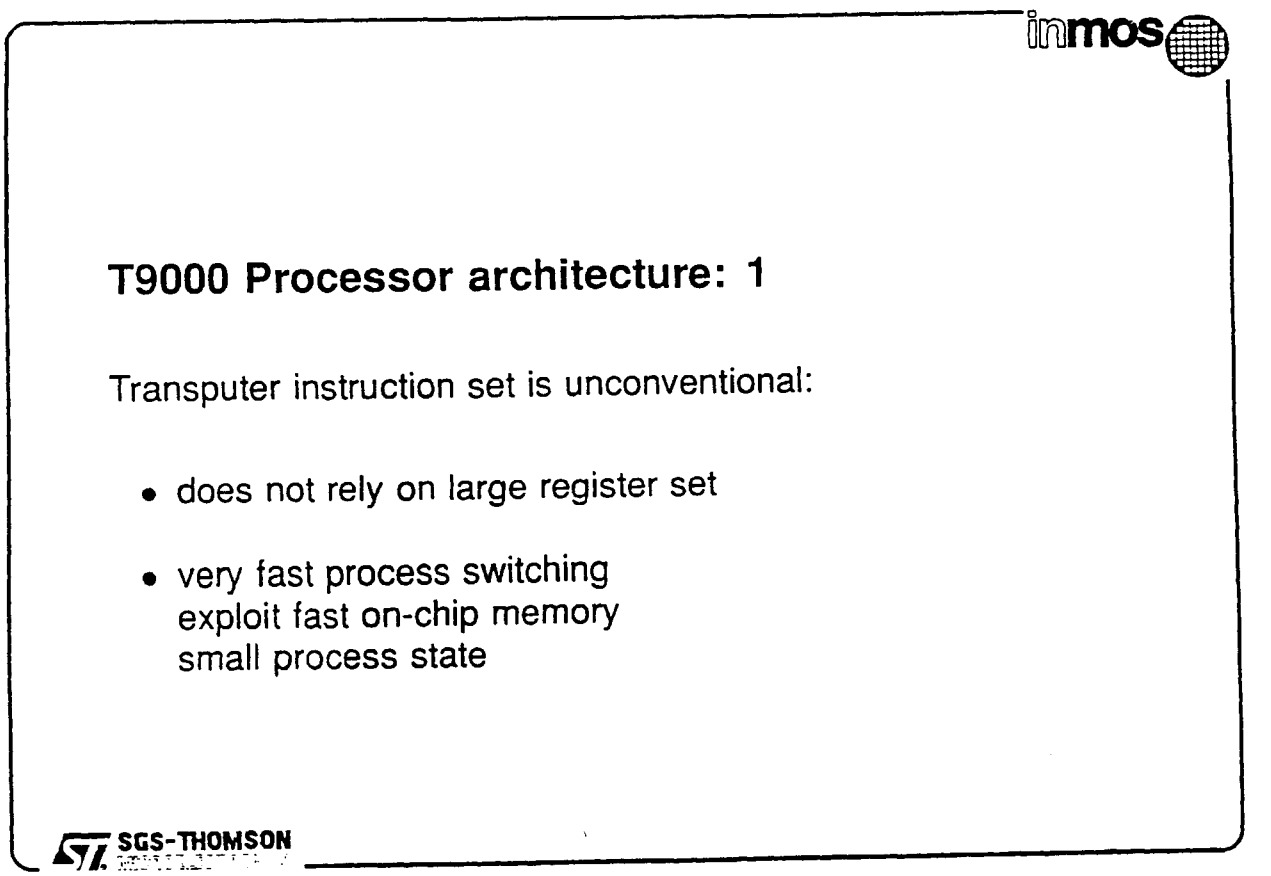
Superscalar processor -

Virtual channel processor -

Multi-banked fully associative cache -



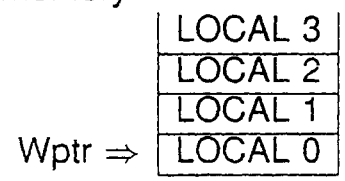

inmos
18.85mm
PadRingNorth
EMI
TDC
FPU
VCP
Links
XBAR
TAP
PadRingWest
PadRingEast
9.85mm
CacheBank1
CacheBank2
CPU
CacheBank3
CacheBank4
PadRingSouth
SGS-THOMSON MICROELECTRONICS


inmos
T9000 Processor architecture: 1
Transputer instruction set is unconventional:
• does not rely on large register set
• very fast process switching
exploit fast on-chip memory
small process state
SGS-THOMSON

inmos

T9000 Processor architecture: 2

Instruction pointer:	Iptr
Workspace pointer:	Wptr
Integer evaluation stack:	Areg Breg Creg
FP-evaluation stack:	FPAreg FPBreg FPCreg
Local variables:	memory



SGS-THOMSON

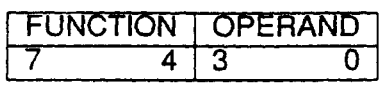
6

inmos

T9000 Instruction set

IDEA: fixed sized, 1-byte instructions
very simple implementation of instruction decoder
dense compiled code

FORMAT:



FUNCTIONS:

load constant	LDC n:	push n onto integer stack
load local	LDL n:	push mem[Wptr + 4*n] onto stack
load local pointer	LDLP n:	push Wptr + 4*n onto stack
load non-local	LDNL n:	load mem[Areg + 4*n] into Areg
load non-local pointer	LDNLP n:	load Areg + 4*n into Areg
jump	J n:	Iptr := Iptr + n

SGS-THOMSON

PREFIX and OPERATE

PROBLEM: 4 bit operand

prefix PFIX n: prepend n onto next operand
eg PFIX 4; LDC 2 has effect of LDC 42

PROBLEM: 16 functions

operate OPR n : treat n as opcode for operation
eg OPR 5: ADD
Areg := Areg + Breg; Breg := Creg

PFIX and OPR together: PFIX 3; OPR 3 = OPR 33 = XOR

PRACTICE: variable sized instructions:
- 1 or 2 bytes without operand
- 1 thru 8 bytes with operand

SGS-THOMSON

8

T9000 Floating point instruction set

IDEA: works like integer set
- use integer stack to perform address calculations
- fp stack loaded from memory
- fp registers tagged with length

FPLDNLSN floating point load non-local single
push mem[Areg] onto FP stack; pop integer stack

FPSTNLSN floating point store non-local single
store FPAreg into mem[Areg]; pop both stacks

FPMUL floating point add
FPAreg := FPBreg * FPAreg; FPBreg := FPCreg

FPLDNLMULSN equivalent to FPLDNLSN; FPMUL (saves 2 bytes)

SGS-THOMSON

9

inmos

T9000: pipelining the instruction set

PROBLEM: Each instruction has low semantic content.
Unsuitable for execution on a multistage pipeline.

SOLUTION: *group* several dependant instruction together
and execute on a pipeline

EXAMPLE: LDL x; LDL y; ADD is equivalent to

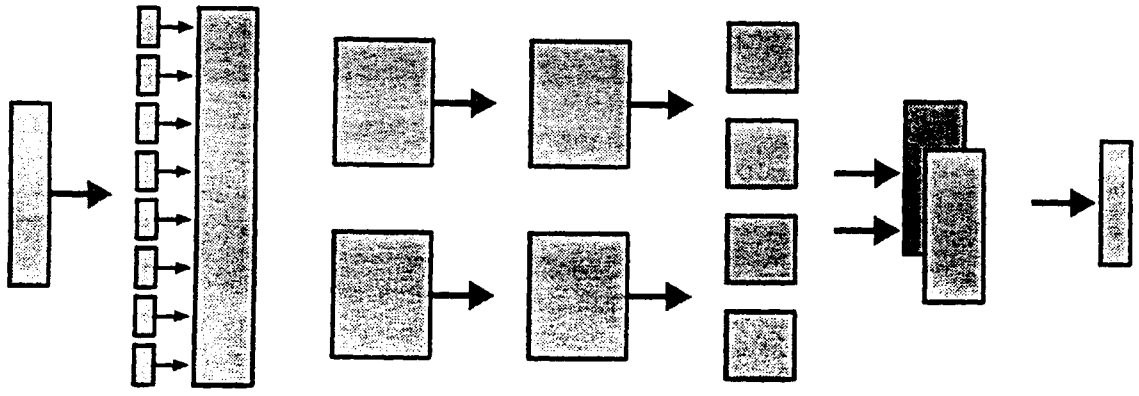
Areg := Mem[Wptr + 4*x] + Mem[Wptr + 4*y]

SGS-THOMSON

10

inmos

Pipeline Structure



FETCH
1 word/ cycle

DECODE
Group up to 8 bytes/cycle

LOCAL
Two reads from tri-ported workspace cache

ADDRESS
Two non-local addresses/ cycle

CACHE
Two reads/ cycle from 4 cache banks

ALU/FPU
One integer or floating point operation

WRITE
1 write/ cycle

SGS-THOMSON MICROELECTRONICS

inmos

T9000: use of pipeline - complex multiply

```
typedef struct {float re, im;} complex;

complex *x, *y, *z;

x->re = (y->re * z->re) - (y->im * z->im);
x->im = (y->re * z->im) + (y->im * z->re);
```

SGS-THOMSON

inmos

T9000: use of pipline - real part

ldl y			
fpldnlsn	`FPAreg = y->re`	Group 1:	4 instructions
ldl z			6 bytes
fpldnlmulsn	`FPAreg = y->re * z->re`		
ldl y			
ldnlp 1		Group 2:	6 instructions
fpldnlsn	`FPBreg = y->re * z->re`		8 bytes
	`FPAreg = y->im`		
ldl z			
ldnlp 1			
fpldnlmulsn	`FPBreg = y->re * z->re`		
	`FPAreg = y->im * z->im`		
fpsub	`FPAreg = (y->re * z->re) -`		
	`(y->im * z->im)`	Group 3:	3 instructions
ldl x			5 bytes
fpstnlsn			

SGS-THOMSON

inmos

T9000: use of pipline - imaginary part

```
ldl y
fpldnlsn      FPAreg = y->re                    Group 4:  5 instructions
ldl z                                                     7 bytes
ldnlp 1
fpldnlmulsn   FPAreg = y->re * z->im
-------------------------------------------------------------------
ldl y
ldnlp 1                                         Group 5:  5 instructions
fpldnlsn      FPBreg = y->re * z->im                      7 bytes
              FPAreg = y->im
ldl z
fpldnlmulsn   FPBreg = y->re * z->im
              FPAreg = y->im * z->re
-------------------------------------------------------------------
fpadd         FPAreg = (y->re * z->im) +
                       (y->im * z->re)          Group 6:  4 instructions
ldl x                                                     6 bytes
ldnlp 1
fpstnlsn
```

SGS-THOMSON

inmos

T9000: Flow through pipeline

Fetch												
Decode	G1	G2	G3	G4	G5	G5	G6	G6	...	...	...	...
Local	...	G1	G2	G3	G4	G4	G5	G5	G6	G6	...	...
Address	...	...	G1	G2	G3	G3	G4	G4	G5	G5	G6	G6
Cache	...	...	...	G1	G2	G2	G3	G3	G4	G4	G5	G5
ALU/FPU	...	...	...	...	G1	G1	G2	G2	G3	G3	G4	G4
Write	...	...	...	...	...	...	G1		G2		G3	
Cycle	1	2	3	4	5	6	7	8	9	10	11	12

SGS-THOMSON

T9000: execution of Group 2

ldl y; ldnlp 1; fpldnlsn; ldl z; ;dnlp 1; fpldnlmulsn

executed over cyles 3 through 9

Cycle	3	4	5 and 6	7 and 8	9
ldl y	Local				
ldnlp 1		Address			
fpldnlsn			Cache		
ldl z	Local				
ldnlp 1		Address			
fpldnlmulsn			Cache	FPU	

Cache access occupies 2 cycles because preceeding group executes 2 cycle FPU operation.

SGS-THOMSON

16

T9000: Pipeline activity on Cycle 5

Fetch		
Decode	G5	decoding
Local	G4	ldl y, ldl z
Address	G3	
Cache	G2	fpldnlsn fpldnlmulsn
ALU/FPU	G1	fpldnlmulsn
Write		
Cycle	5	5

SGS-THOMSON

17

inmos

T9000: Pipeline activity on Cycles 7 and 8

Fetch			
Decode	G6	decoding	G6
Local	G5	ldl y, ldl z	G5
Address	G4	ldnlp 1	G4
Cache	G3		G3
ALU/FPU	G2	fpldnlmulsn	G2
Write	G1		
Cycle	7	7	8

SGS-THOMSON

inmos

T9000: pipeline performance

Fetch												
Decode	G1	G2	G3	G4	G5	G5	G6	G6	...	...	...	...
Local	...	G1	G2	G3	G4	G4	G5	G5	G6	G6	...	...
Address	...	...	G1	G2	G3	G3	G4	G4	G5	G5	G6	G6
Cache	...	...	...	G1	G2	G2	G3	G3	G4	G4	G5	G5
ALU/FPU	...	...	...	...	G1	G1	G2	G2	G3	G3	G4	G4
Write	...	...	...	...	...	...	G1		G2		G3	
Cycle	1	2	3	4	5	6	7	8	9	10	11	12

6 cycles for 3 groups
for 13 instructions gives > 2 instructions/cycle
gives about 108 MIPS
for 3 fp operations gives 25 MFLOPS

SGS-THOMSON

inmos

T9000: summary

- 1 micron CMOS 3 level metal process.
- 185sq mm die in excess of 2.3 million transistors.
- 3 - 8 watts power dissipation @50Mhz.
- Packaged in a 208 pin CQFP.
- T9000 available Q1 92.
- C104 packet through routing switch available Q2 92.
- C100 link protocol converter available Q1 92.

SGS-THOMSON MICROELECTRONICS