Challenges of Parallel Processor Design

Martti Forsell (VTT Oulu)
Ville Leppänen (University of Turku)
Martti Penttonen (University of Kuopio)

May 18, 2009

Contents

- Moore's law
- Latency
- Slackness
- PRAMs on Chip
  - Paraleap
  - Eclipse
  - Moving threads

Moore's law

- 1 component on IC in 1959
- 50 component on IC in 1965
  Moore: maybe 65000 components on IC in 1975
  16 years — 2^{16}-fold
- 2^{32} (not 2^{48}) components on IC in 2007
  "Packing density doubles every 18 months"
- "Laws" for clock cycles, bandwidth, ...
- Not until eternity! size, heat, quantum effects
- What to do with all those components? Multiple cores?

Latency

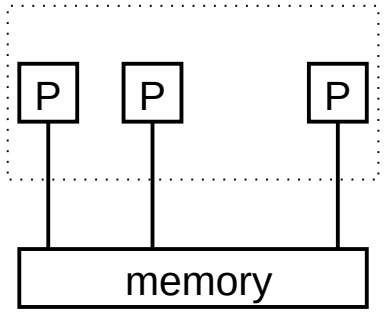
- moving data needs time
- overhead of components
- latency of about 100 clock cycles
- want to process but must wait for data
- caches - clever enough?
- multiple cores - what to do with them?
- threads become important

Slackness

Does latency imply inefficiency?

- What to do instead of waiting? Some other thread
- Are there parallel threads? Yes, PRAM algorithmics
  Multiple threads per processor core: *slackness*
- Is it technically possible to run multiple threads?
  Bandwidth requirements for internal network
- Any number of processors
- Different structure of computer
- New software (at least libraries)

PRAM



Multiple processors running synchronously, shared memory.

```
proc compact(A)
for i=0..n-1 pardo
  if A[i]=0 then C[i]=0 else C[i]=1
E=prefix-sum(C)
for i=0..n-1 pardo
  if A[i]<>0 then B[E[i]]=A[i]
return B
```

PRAM continued

$O(1)$ time assuming `prefix-sum` in $O(1)$ time

```
prefix-sum(C) = (C[1],C[1]+C[2],C[1]+C[2]+C[3],...)
```

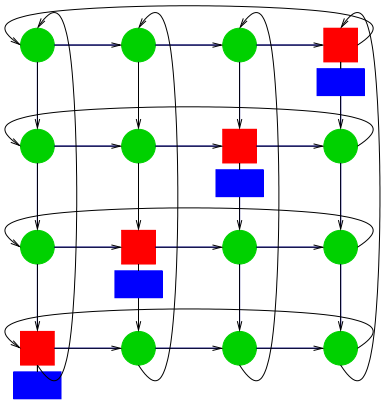
A lot of progress in 80'ies and 90'ies.
Hypothesis: $NC = P$, where $NC = \bigcup_k ParTime(\log^k n)$
Hence, for most problems there are highly parallel algorithms.

Culler et al. 1993. PRAM is not realistic. Synchronous immediate access to memory is not possible. PRAM is passé ! Try DMM!

Now: DMM is passé ? Try PRAM!

Slackness

- Assume program uses sp virtual processors, while computer has p real processors. We have *slackness* s in computation.
- Assume each data fetch requires ϕ hops in network. In time unit $p\phi$ bandwidth need is created.
- ϕ is not constant, therefore network must be sparse, for example sparse torus



PRAM on Chip

What changed in fifteen years?

- DMM never became very popular
- Dead end in commodity processor speedup
- Space on chip $\Rightarrow$ PRAM on chip becomes possible

PRAM on chip

- Paraleap (Vishkin et al.)
- our Eclipse (Forsell et al.)
- our Moving threads (Leppänen et al.)

PRAM on Chip design challenges

1. Enough parallelism to cover latency? Yes by PRAM theory
2. Enough communication bandwidth? Use sparse network
3. Efficient management of slackness on hardware?
4. Programming not too difficult?

Paraleap

Vishkin's XMT (Eplicit MultiThreading) model. Not as tightly synchronous as PRAM.

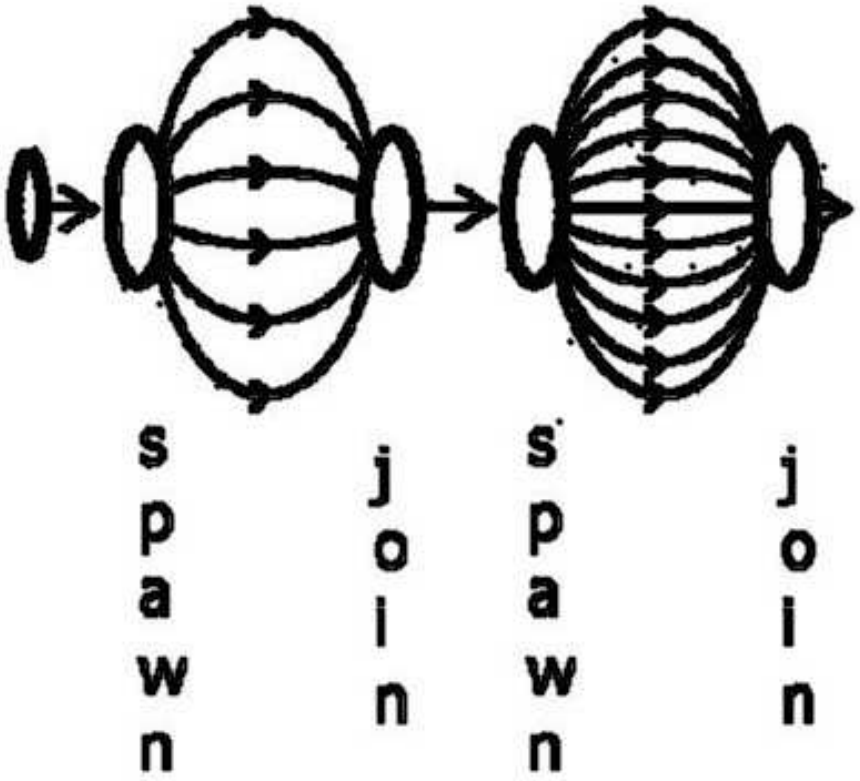


Fig. A.2 Parallel and serial mode

PRAM and XMT are similar

PRAM Pseudo-Code

```
for i = 0 to m-1 pardo
{
  C[i] = A[i] + B[i];
}
```

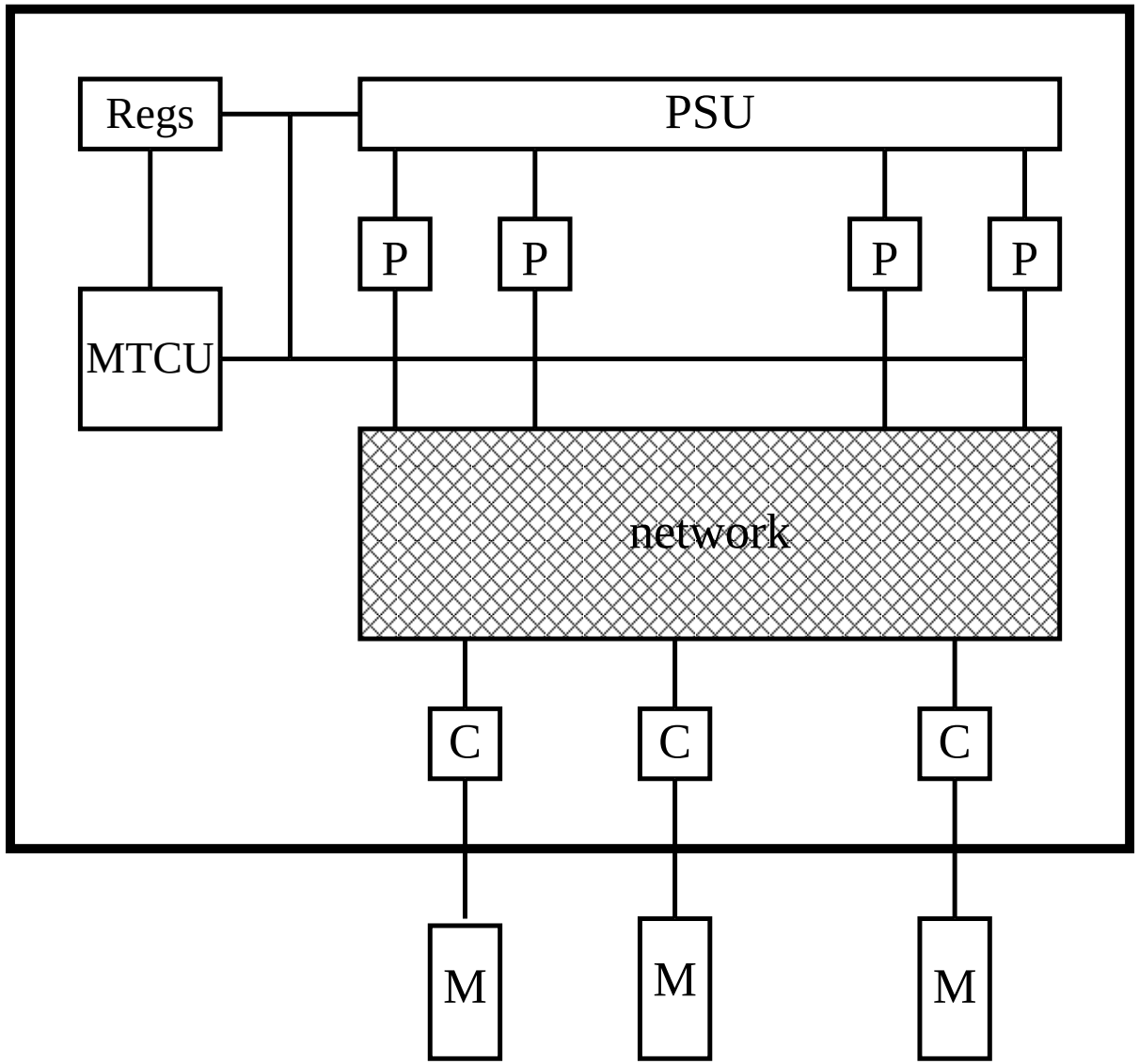
XMTC Code

```
spawn(0, m-1)
{
  C[$] = A[$] + B[$];
}
```

PRAM and XMT are different

```
int x = 0;
Spawn(0, n) /* Spawn n threads; $ ranges 0 to n – 1 */
{ int e = 1;
   if (A[$] not-equal 0)
       { PS(x,e);
         D[e] = A[$] }
}
n = x;
```

Structure of Paraleap

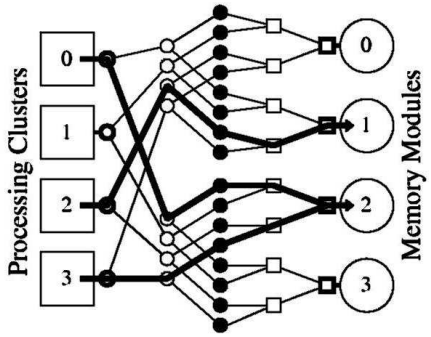


How does Paraleap work?

- At spawn TCU gets the number of parallel threads and TPU's get the code for running the thread
- At the beginning and whenever a thread is completed, a TPU asks the TCU for a new thread
- TCU uses the prefix-sum for pointing to the next thread if any remain
- When all threads have been completed, control returns to the MPU

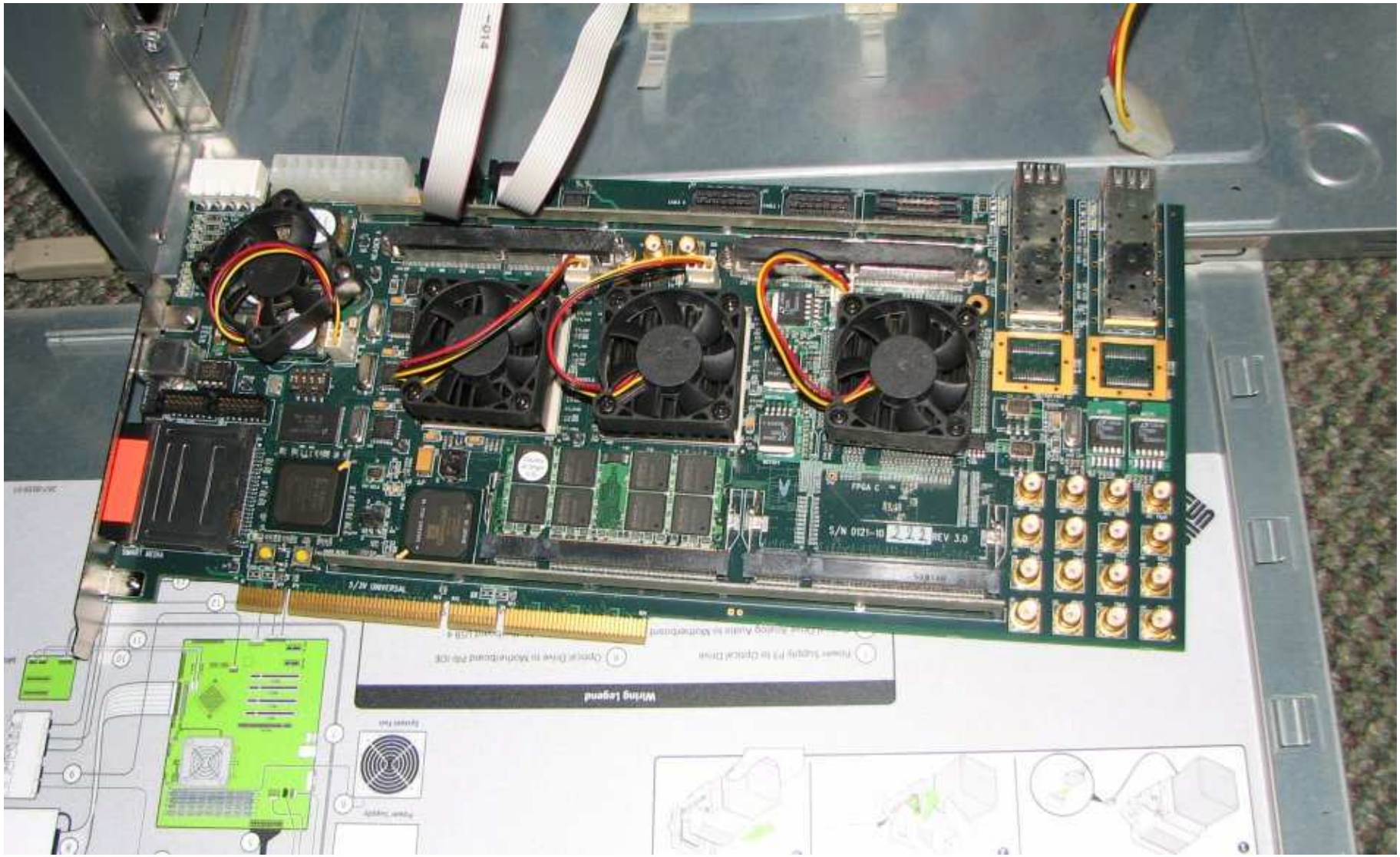
Implementation issues

- Prefix-sum is actually implemented sequentially. It is claimed to be fast enough. Really? How scalable?
- Internal network is a mesh of trees

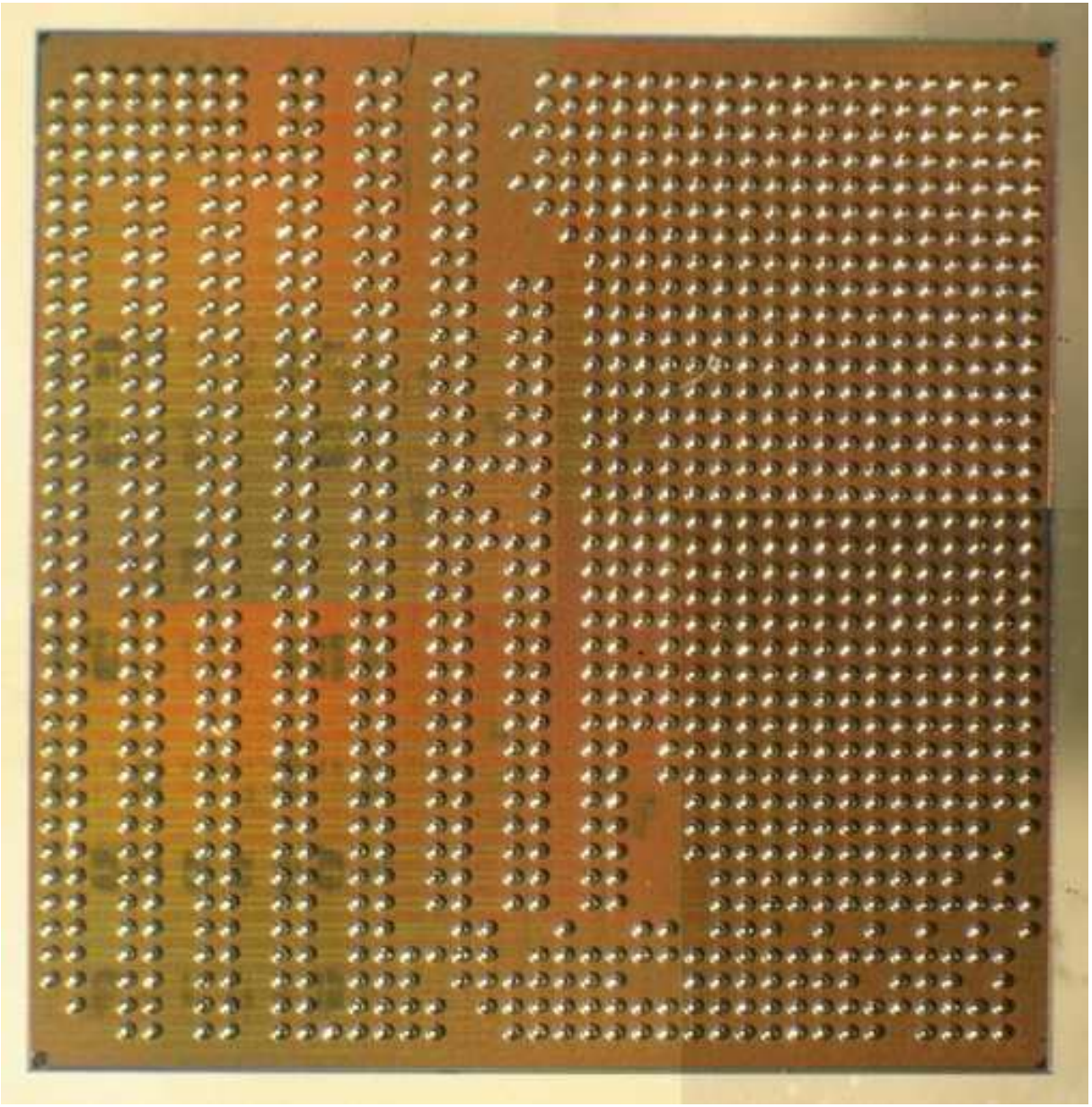


- Implemented on FPGA (Field Programmable Gate Array) at 75 MHz
- Current version has 64 TPU's in 4 clusters of 16 TPU's sharing some functional units and network access

Paraleap exists



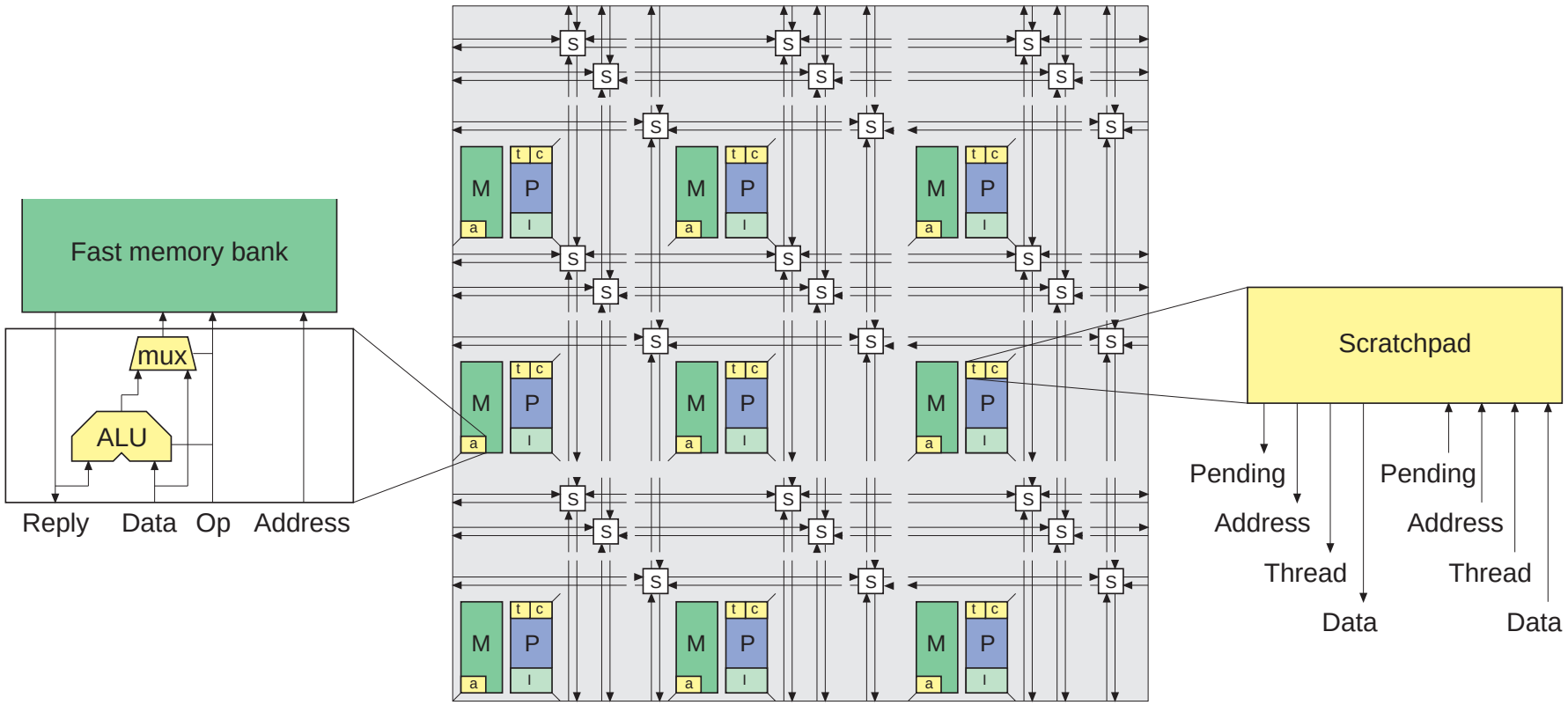
Paleap goes ASIC



Eclipse

- strong PRAM models on chip
- interleaved multithreading exploits slackess of algorithms
- chained sequential functional units
- supports instruction level parallelism of sequential code
- sparse mesh
- local memories and "scratchpads" (used for multioperations)
- compiler, simulated running,
- FPGA implementation planned

Structure of Eclipse



Moving threads

- Processors have local memory
- For data access, process with environment registers moves to the processor that has the data
- No two-way traffic for a read. Fewer but bigger data packets
- Tentative design exists, simulations by software

CUDA project

- use NVIDIA graphic processor as shared memory parallel computer
- cheap processing power
- special libraries written

Conclusions

- PRAM on chip seems feasible
- Breakthrough?
- A lot of work remains to be done
- For popular introduction in Karelian, see http://opastajat.net "luvekkua karjalakse" (The same appeared in Finnish in Tietojenksittelytiede)