Towards Transputing with Parallella !!

Kazuto MATSUI

NPO CSP Consortium

matsui@csp-consortium.org

30th/May/2015

Introduction

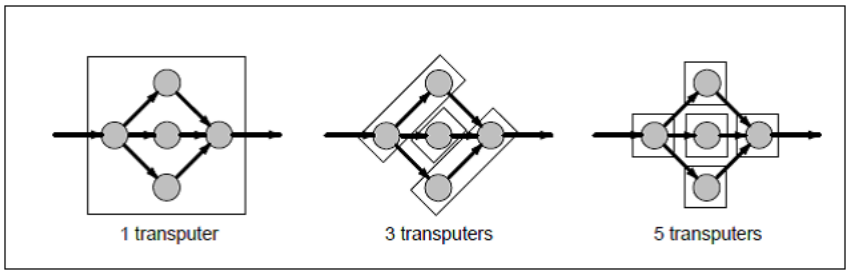
Epiphany's chip consists of 16/64 of cores which also provides a shared memory and a message passing architecture.

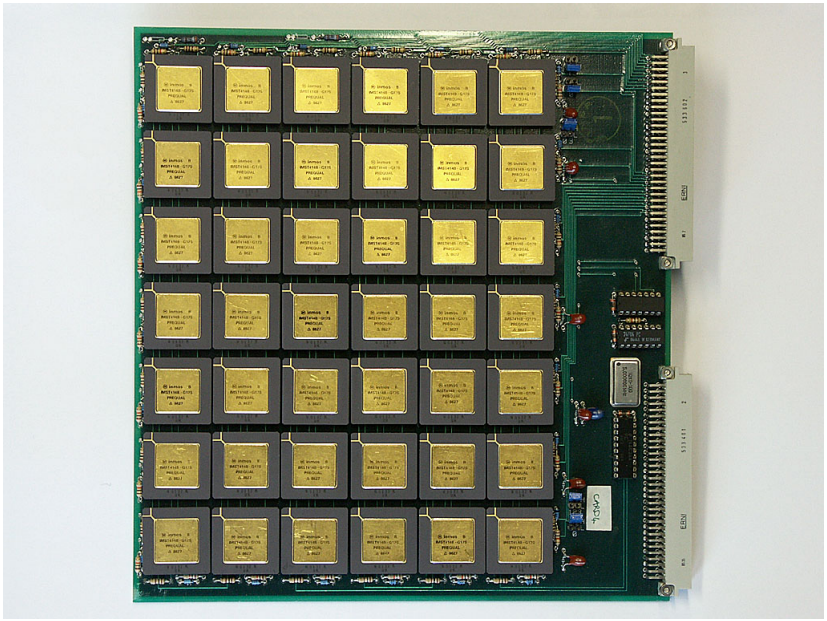
To achieve the scalable performance, a parallel programming based on both a shared memory and a message passing is the key issue.

But many users are mostly using only a shared memory.

So my talk is to fill the gap of this use.

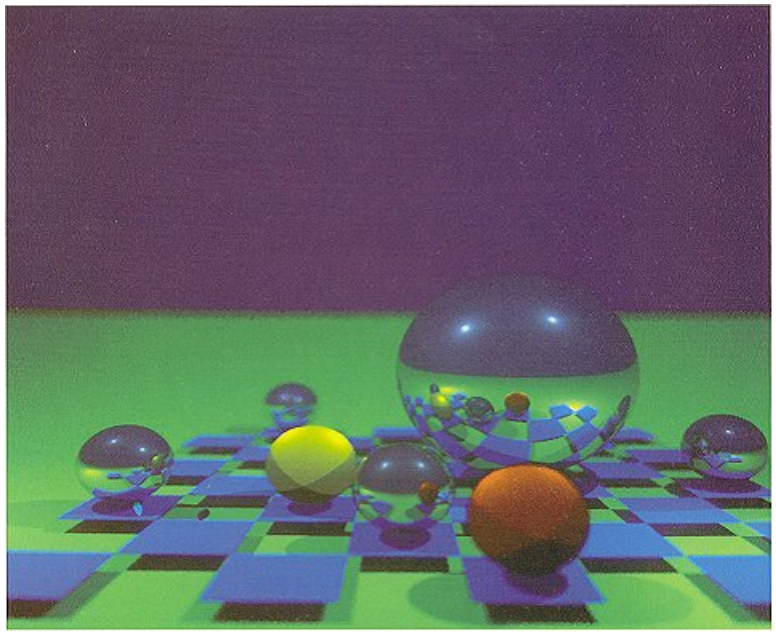
Transputing in the Past



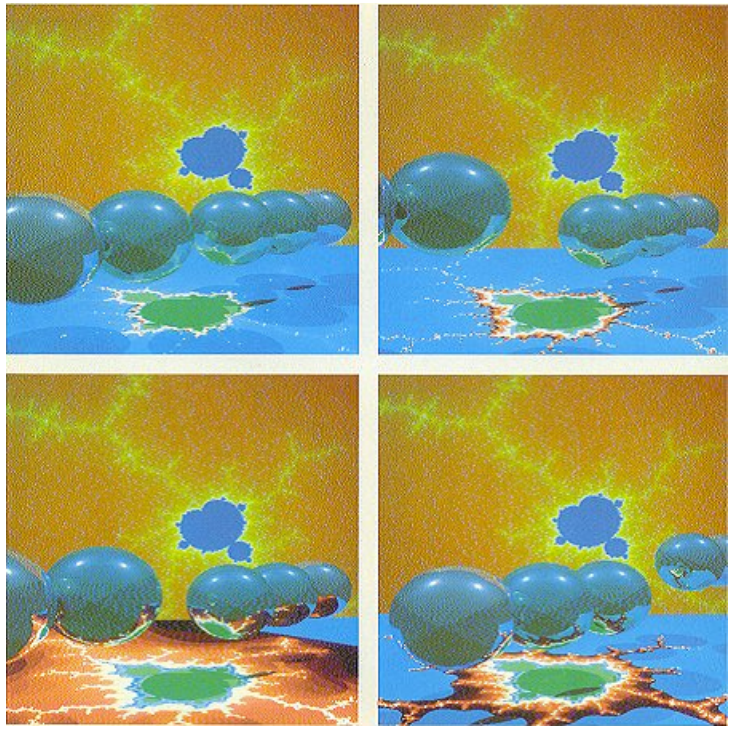


We very much enjoyed parallel computing with Inmos Transputer[1] from mid 80's to mid 90's. That's why I like to call it , say Transputing !!

Excellent Transputer Demos !!



Raytracing



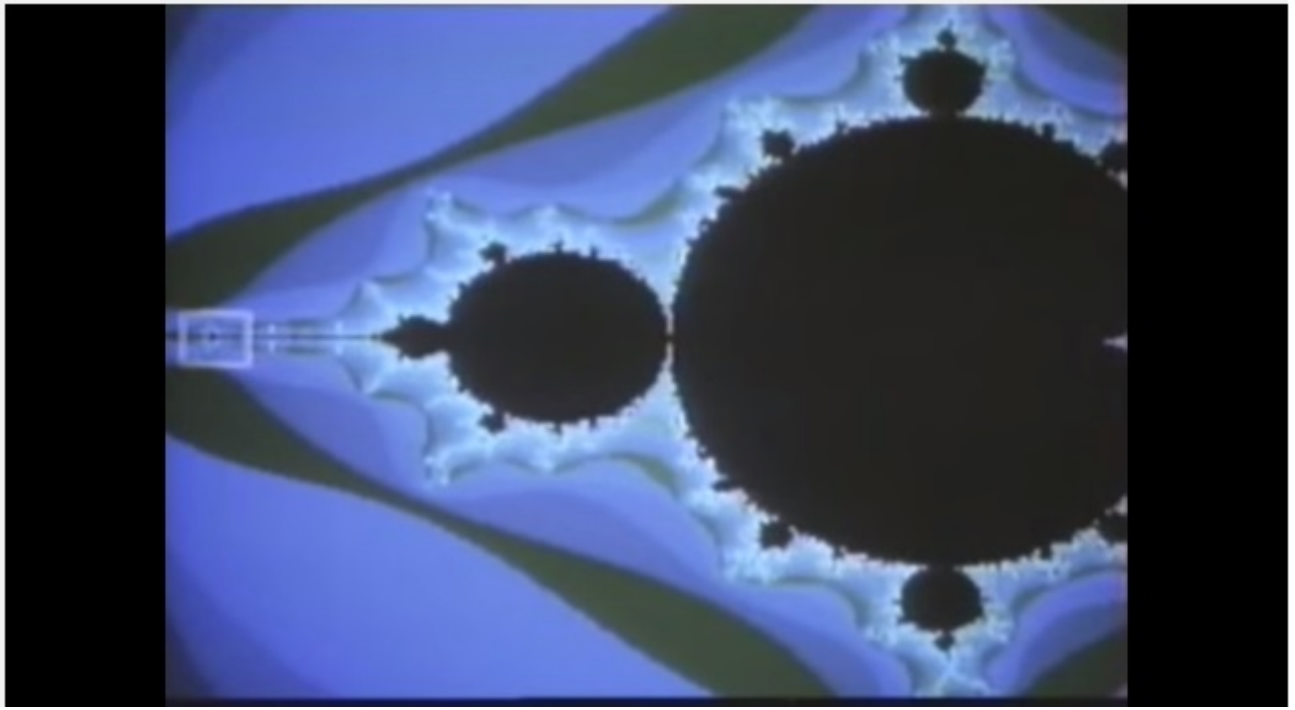
Newton's Cradle

A hundred of pipelined Transputers demo was so amazing !! Using farming algorithm, over 90% of scalability was gained[3].

A collision of each sphere in Newton's cradle was so realistic !!

transputers	speed	relative speed	linearity %
1	164.0	1.00	100.0
2	327.6	2.00	99.9
4	654.0	3.99	99.7
8	1296.4	7.91	98.8
16	2601.6	15.87	99.2
32	5189.5	31.65	98.9
64	10300.0	63.15	98.7
80	12500.0	76.37	95.5

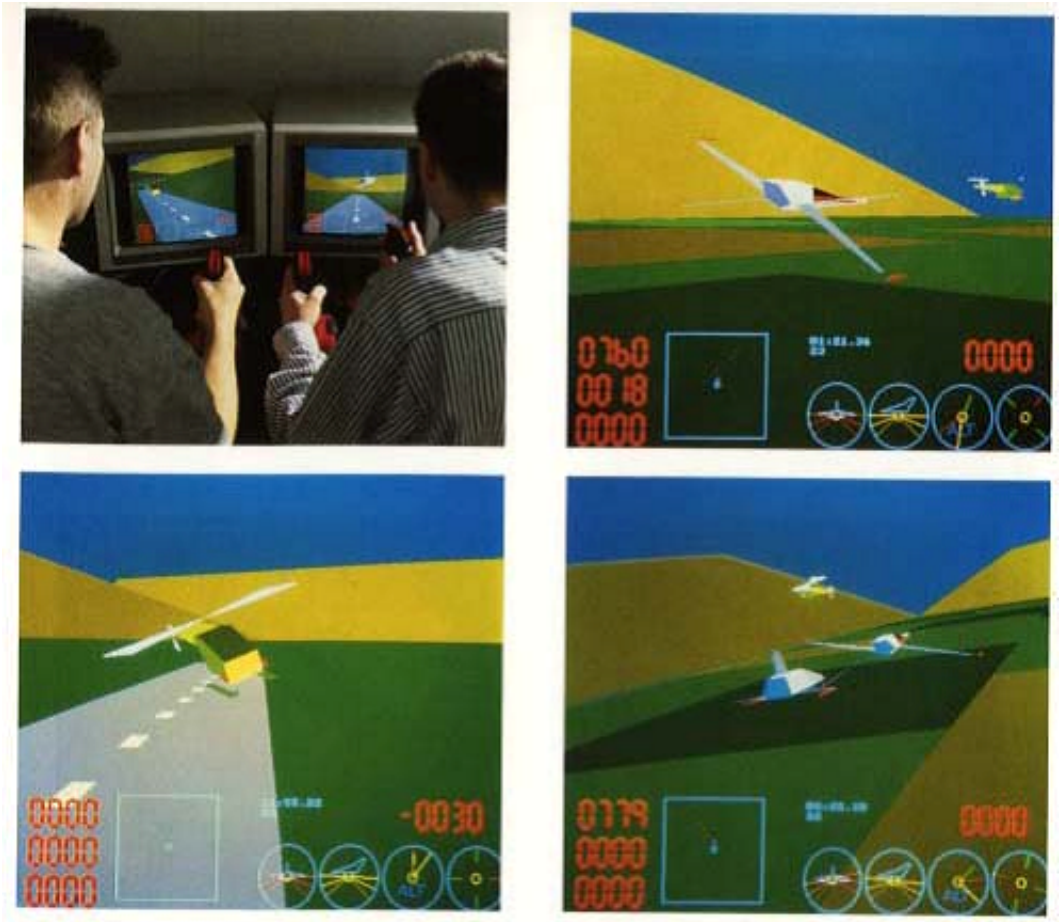
Note: Copied from Jamies' technical note[3].



Inmos had also Mandelbrot set demo running on a hundred of Transputers.

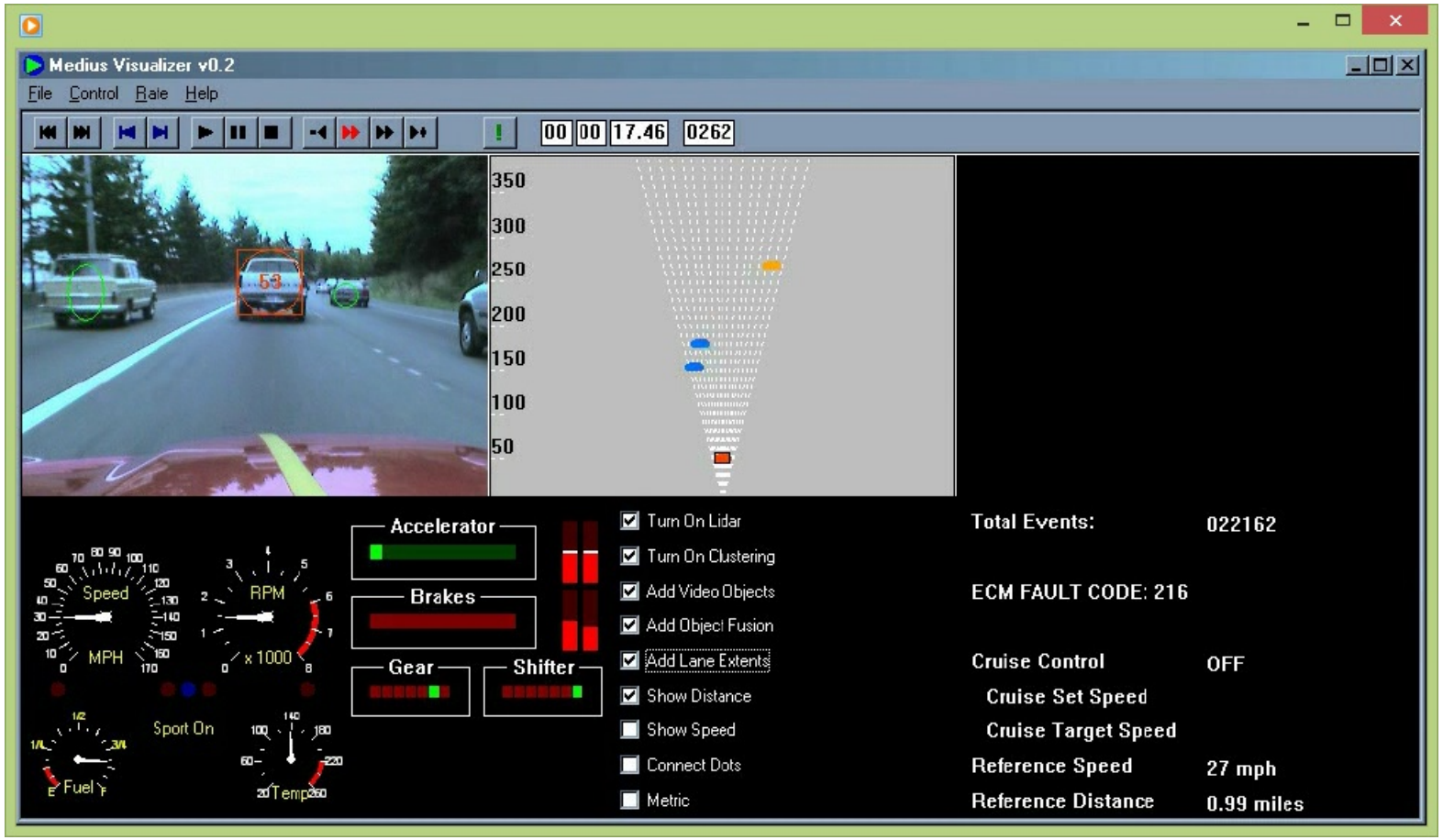
This was another surprising demo.

Flight Simulator(1987)



An object flied over the PC via synchronous communications channels. This idea would be a good reference to Parallella.

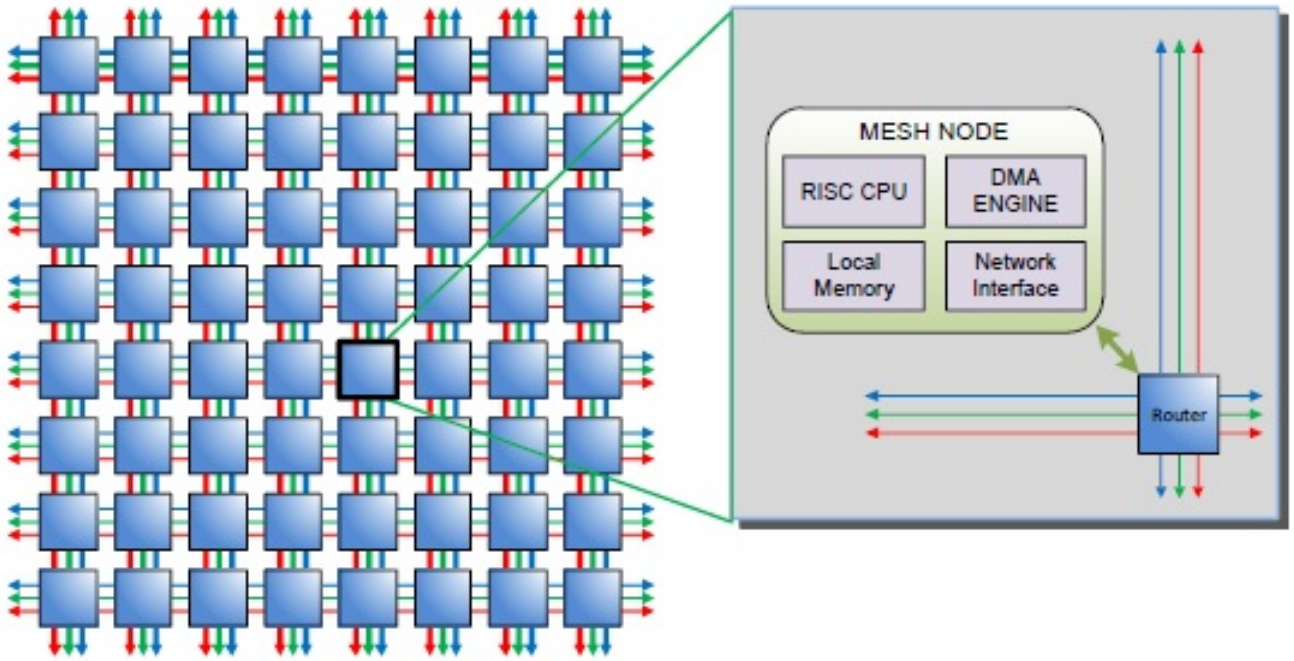
Video Tracking (Handel-C)



As cars are moving forward, tracking image data move to adjacent core quickly. This event driven model requires a high speed serial links between cores.

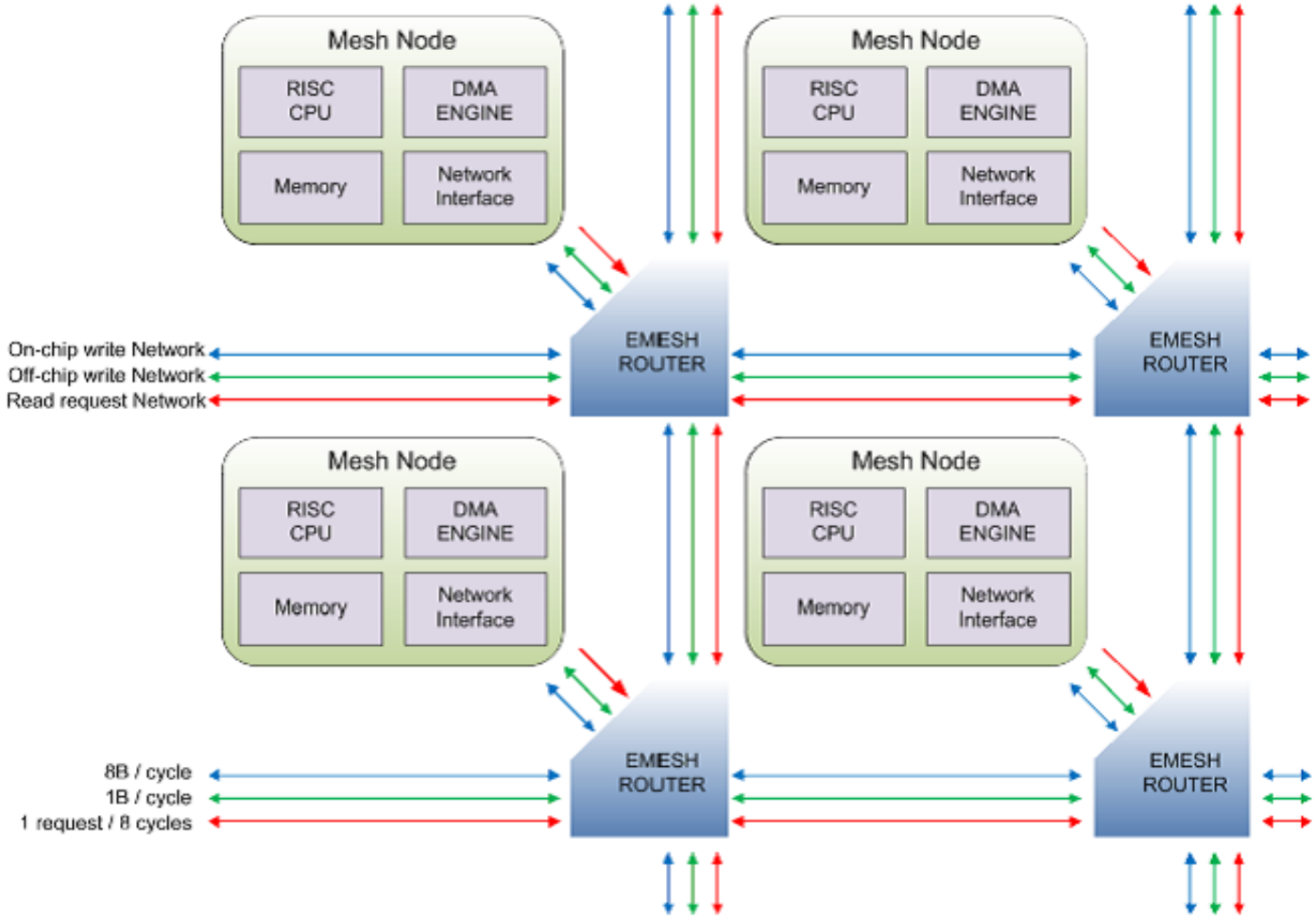
Epiphany Chip Architecture

http://www.adapteva.com/

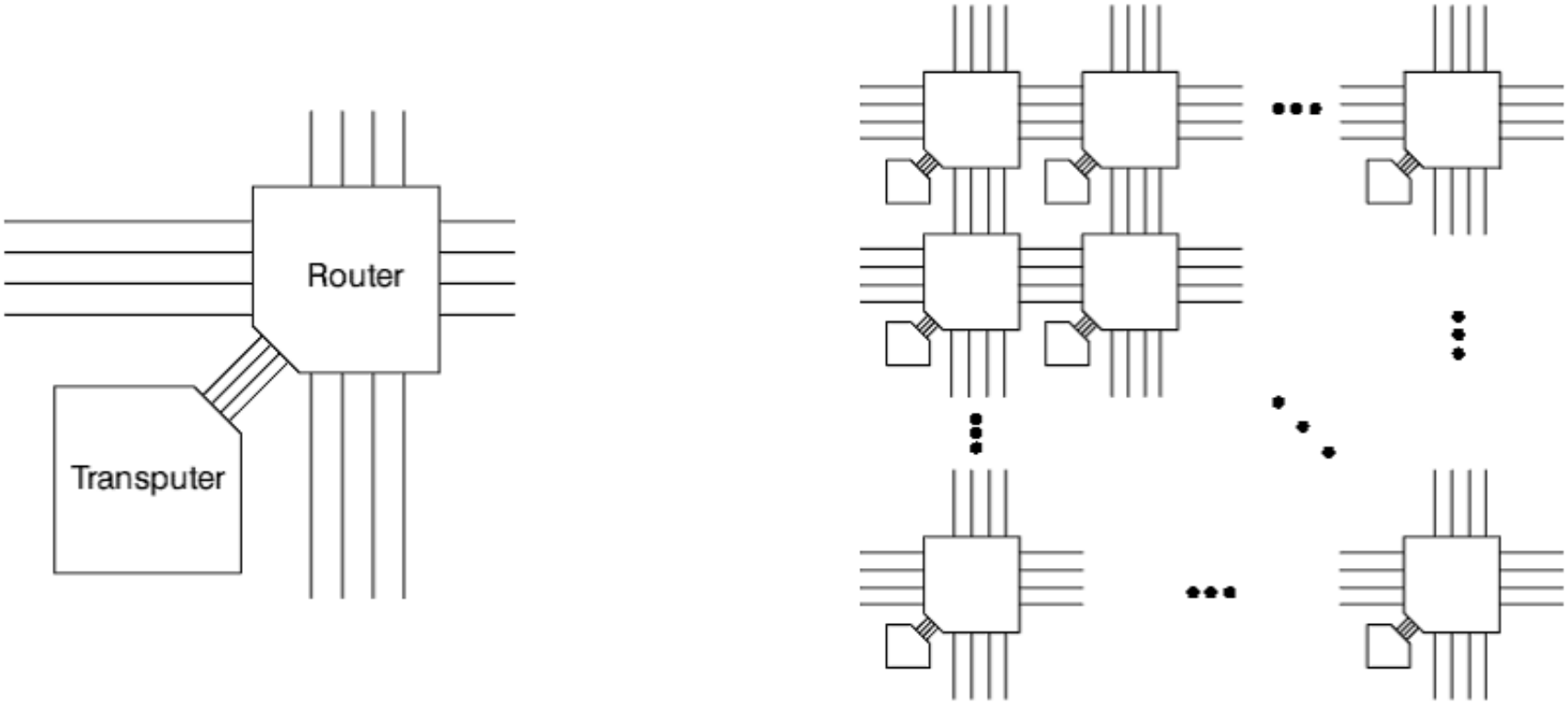


Epiphany architecure defines a multicore, scalable, shared-memory, parallel computing fabric. It consists of a 2D array of compute nodes connected by a low-latency mesh network-on-chip.

eMesh Network-on-Chip Structure

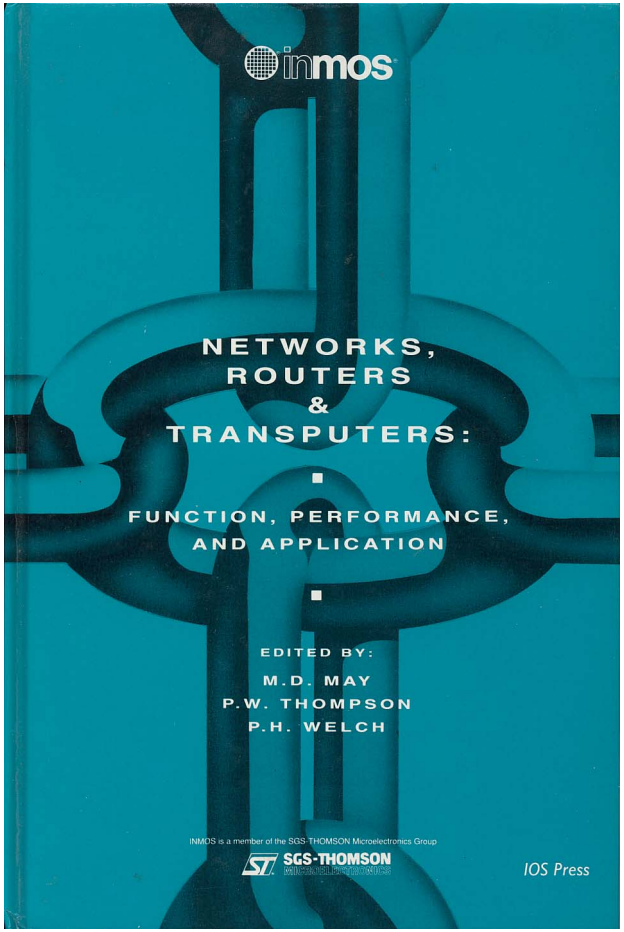


Transputer Networks



Epiphany NoC(Network-on-Chip) architecture is similar to the Transputer[1].

The 2nd generation of Transputer



In early 90', C104 (asynchronous switch) chip appeared. An interconnect structure of the CPU and the router is the same as NoC. What does this mean ?

Programming Models

The Epiphany architecture is programming-model neutral and compatible with most popular parallel-programming methods, including Single Instruction Multiple Data (SIMD), Single Program Multiple Data (SPMD), Host-Slave programming, Multiple Instruction Multiple Data (MIMD), static and dynamic dataflow, systolic array, shared-memory multithreading, message passing, and communicating sequential processes (CSP).

Adapteva anticipates that with time, the ecosystem around the Epiphany multicore architecture will grow to include many of these methods.

The key hardware features in the Epiphany architecture that enables effective support for parallel programming methods are:

- *General-purpose processors that support ANSI C/C++ task level programming at each node. Shared-memory map that minimizes the overhead of creating task interfaces.*
- *Distributed-routing technology that decouples tasks from.*
- *Inter-core message-passing with zero startup cost.*
- *Built-in hardware support for efficient multicore data-sharing.*

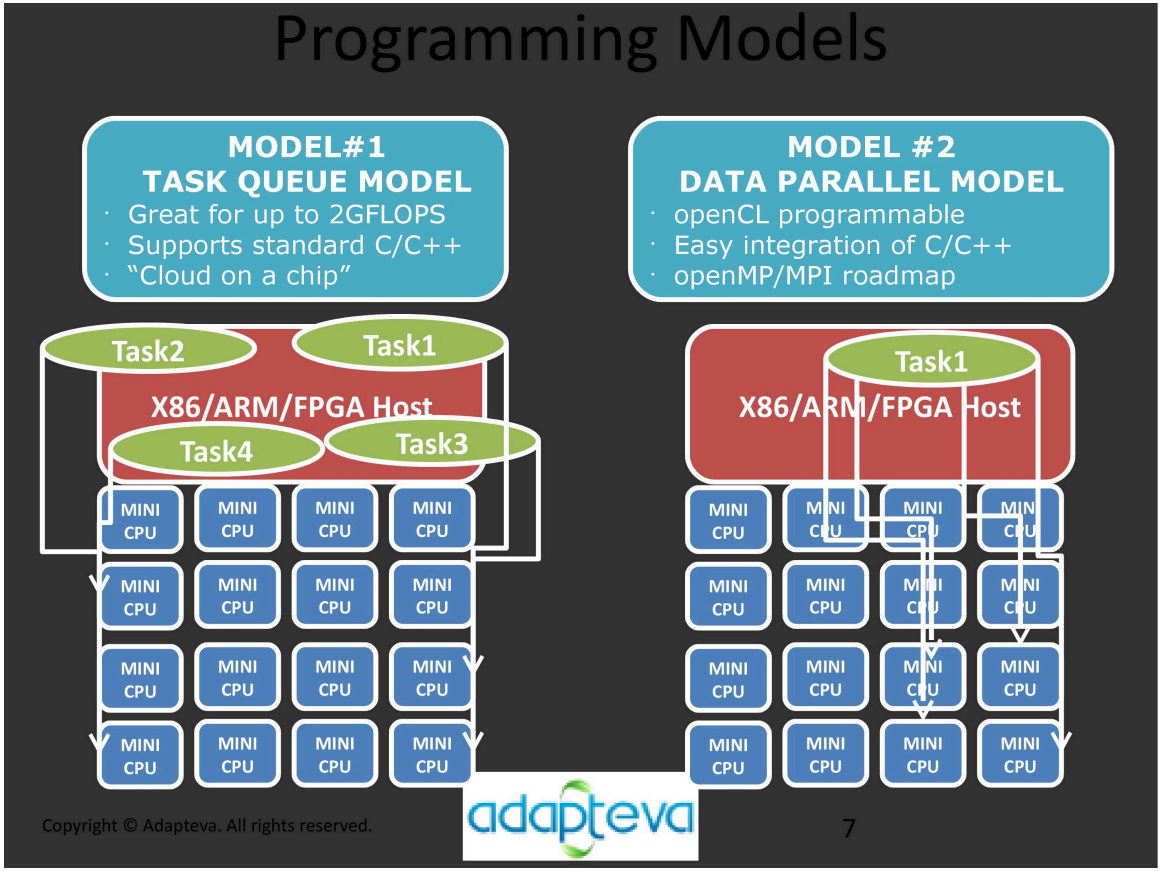
Note: This sentences are written in Chapter 2, Epiphany Architecture Reference.

Programming Languages Supported

Currently, following programming languages are mainly used.

- OpenCL
- OpenMPI
- Go
- Erlang
- Python
- ANSI-C/C++

Computing Models



Task parallel and data parallel are fine but CSP[2]/occam[5] model covers more ...

Next Target is occam model!

We(CSP/occam community[12]) are looking at occam or occam-like programming languages to use Parallella.

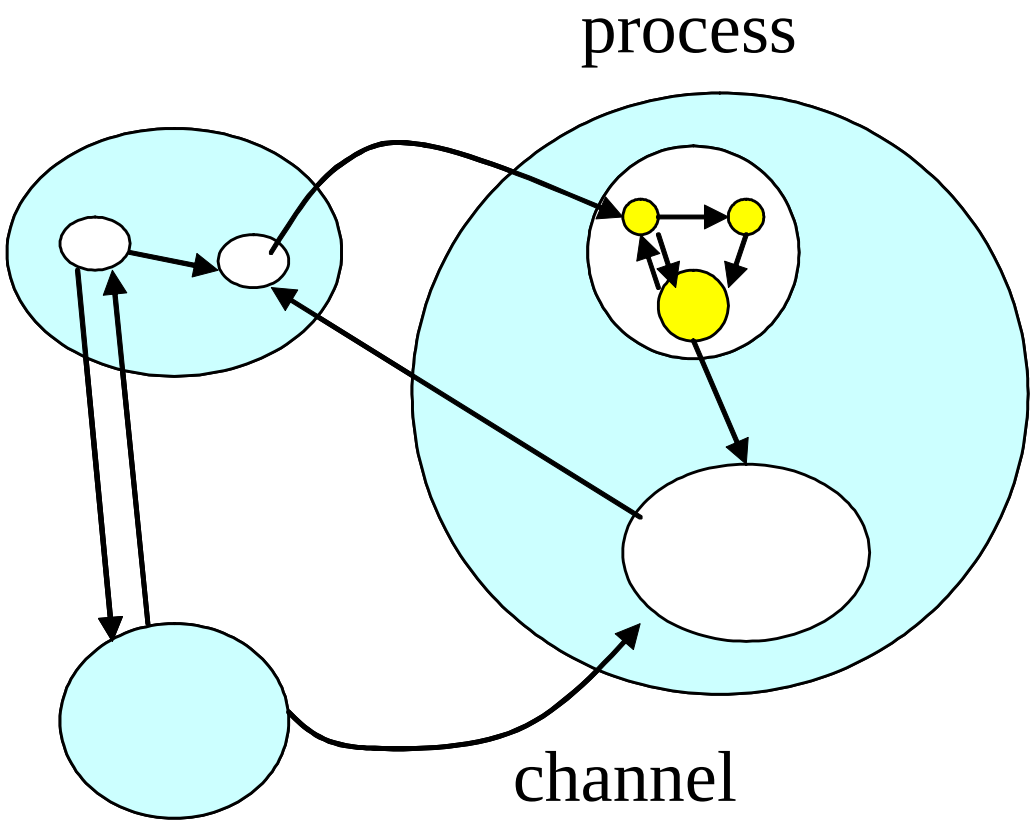
- occam-pi(http://www.cs.kent.ac.uk/projects/ofa/kroc/)
- Transterpreter(http://www.transterpreter.org/)
- JCSP (http://www.cs.kent.ac.uk/projects/ofa/jcsp/)
- C++CSP(http://www.cs.kent.ac.uk/projects/ofa/c++csp/)
- Others (PyCSP/Python-CSP/CPH/GroovyCSP/etc)

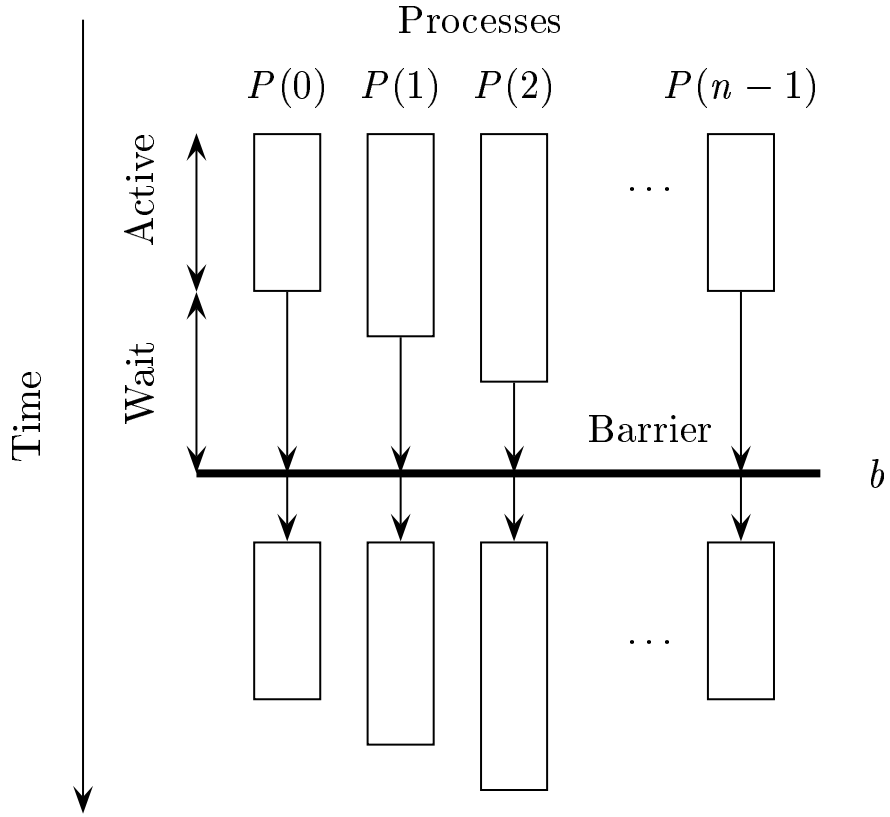
Actually, occam-pi was ported by Halmstad University at Sweden [11].

Note: CSP/occam model supports more higher level of concurrency [8].

occam-pi Processes

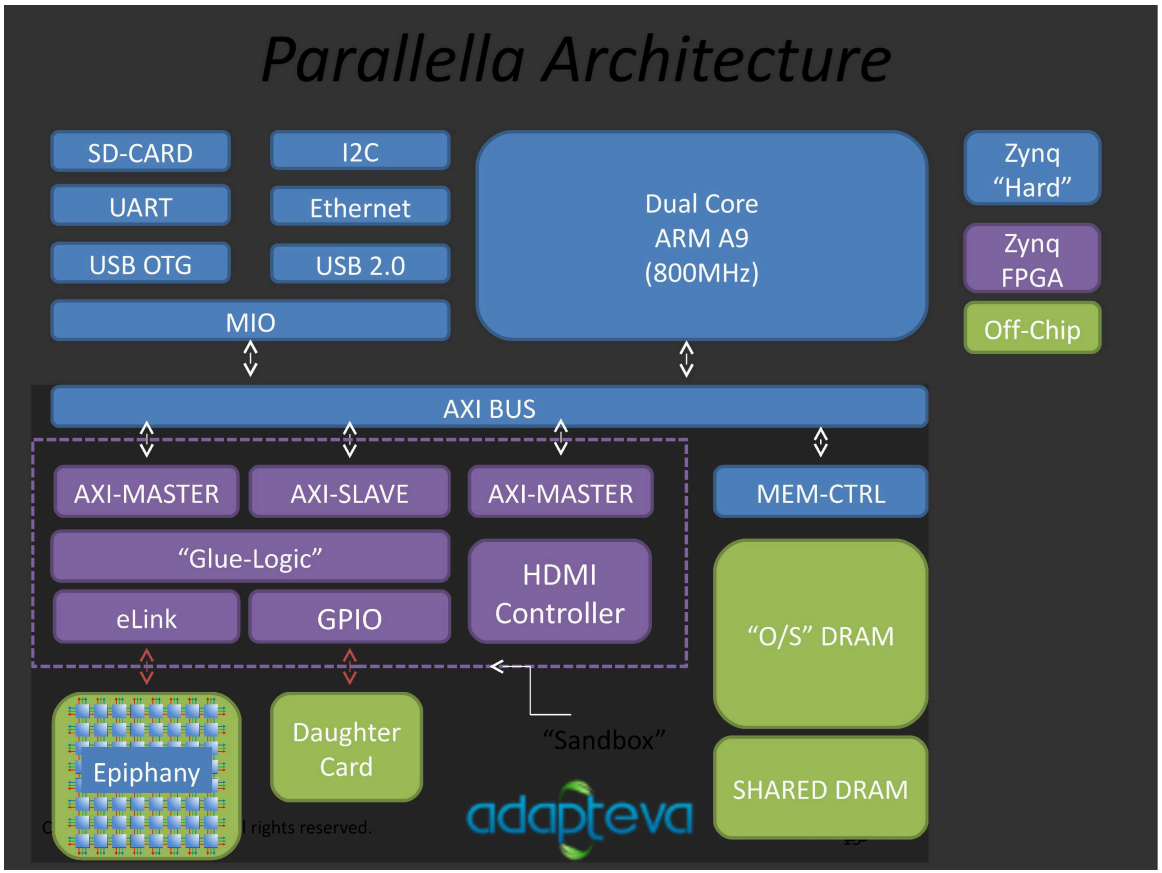
$$
\begin{array}{lll}
\texttt{PROC} & ::= & \texttt{STOP} \mid \texttt{SKIP} \\
& & \mid \mathit{chan}!e \mid \mathit{chan}?v \\
& & \mid \texttt{SEQ}\ [P_1, P_2, \cdots, P_n] \\
& & \mid \texttt{PAR}\ [P_1, P_2, \cdots, P_n] \\
& & \mid \texttt{PRI PAR}\ [P_1, P_2, \cdots, P_n] \\
& & \mid \texttt{ALT}\ [\mathit{cond}_1\ P_1, \cdots, \mathit{cond}_n\ P_n] \\
& & \mid \texttt{PRI ALT}\ [\mathit{cond}_1\ P_1, \cdots, \mathit{cond}_n\ P_n] \\
& & \mid \texttt{WHILE}\ b\ P \\
& & \mid \texttt{IF}\ [b_1\ P_1, b_2\ P_2, \cdots, \texttt{TRUE}\ P_n] \\
& & \mid \texttt{CASE}\ e\ [e_1\ P_1, e_2\ P_2, \cdots, \texttt{ELSE}\ P_n] \\
& & \mid \texttt{CLAIM}\ \mathit{to.chan}!, \texttt{CLAIM}\ \mathit{to.chan}? \\
& & \mid \texttt{CHAN MOBILE ch} \\
& & \mid \texttt{PAR BARRIER}\ b\ [P_1(b), P_2(b), P_3(b)] \\
& & \mid \texttt{MOBILE BARRIER b} \\
& & \mid \texttt{CREW} \\
& & \mid \cdots
\end{array}
$$


process
channel



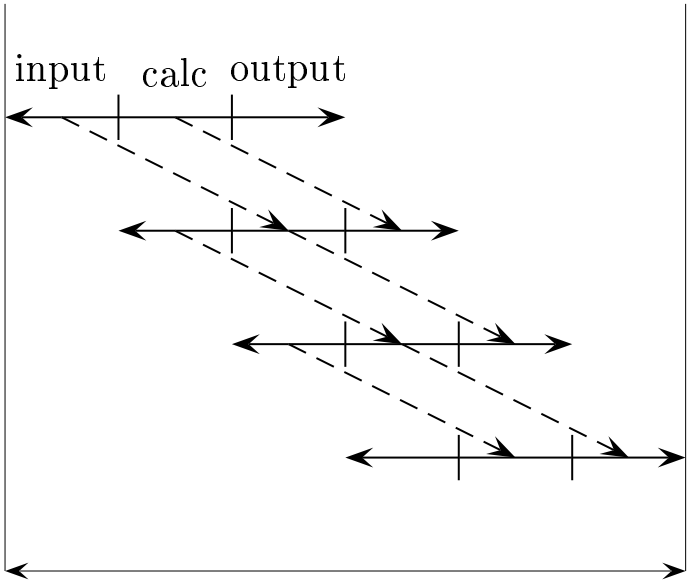
Computing time for each process $P(0), P(1), \cdots, P(n-1)$ is different but all are waiting to reach barrier b.

Parallella Architecture



When applying Parallella to a specific application, using `I/O PAR` , a performance of eLink or GPIO increases significantly !! Since input and output can be overlapped[4].

I/O PAR



```
INT[N] a, b, c, d:
 SEQ
   input(a)
   PAR
     input(b)
    calculate(a)
   SEQ
     PAR
       input(c)
       calculate(b)
     output(a)
     PAR
       input(d)
       calculate(c)
     output(b)
     PAR
       calculate(d)
       output(c)
     output(d)
```

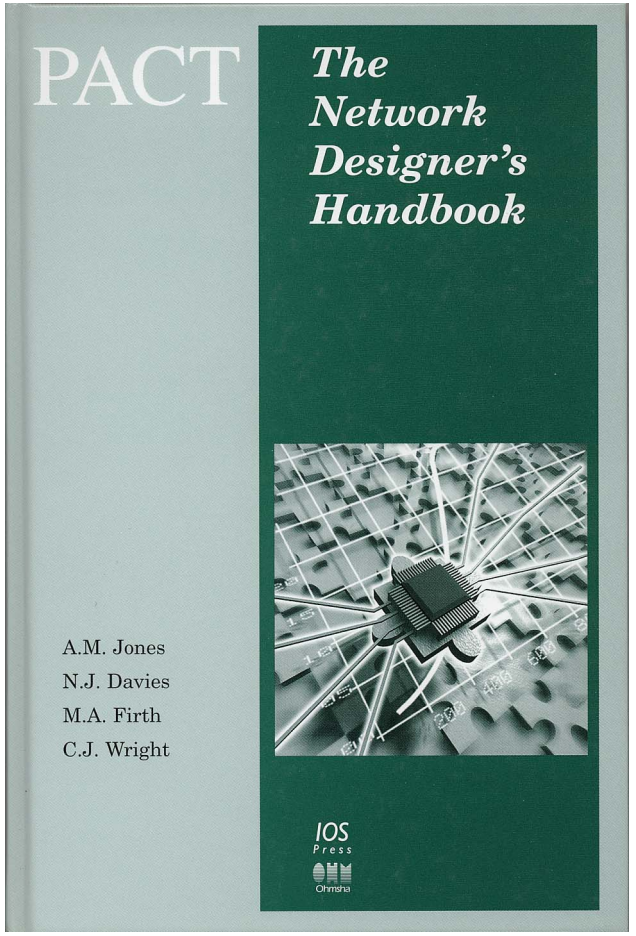
Hybrid Model of Parallel Computing

Hybrid model of parallel computing is a trend of industry. Actually, GPU is introducing MPI library[13]. So why not doing with Parallella ?

- Shared Memory
- Message Passing
- Barrier Synchronization
- Mobile
- I/O PAR
- ...

Expected Silicon Modification

- 32Kbyte of the local memory is too small !!
- Network configuration to support network topologies is needed.
  - Mesh
  - Torus
  - Butterfly
  - Hypercube
  - Clos



This book is also useful to understand traits of network topologies !!

Conclusion

As Epiphany chip architecture is similar to Transputer, a process model is proposed to increase performance.

Hopefully, many of you understand this !!

Any questions ?

References

[1] http://en.wikipedia.org/wiki/Transputer

[2] C.A.R. Hoare, *Communicating Sequential Processes*, Communications of the ACM 21(8), pp. 666-677, August 1978.

[3] Jamie Packer, *Exploiting concurrency: a ray tracing example*, INMOS Technical Note 7, 1987.

[4] Phil Atkin, *Performance Maximisation*, INMOS Technical Note 17, 1987.

[5] INMOS Limited, *occam 2 Reference Manual*, Prentice Hall, 1988.

[6] M.D. May, P.W. Thompson and P.H. Welch, *Networks, Routers and Transputers: Function, Performance and Applications*, IOS Press, 1993.

[7] Peter H. Welch, George Justo and Colin Willcock, *High-Level Paradigms for Deadlock-Free High-Performance Systems*, Transputer Applications and Systems '93, IOS-Press, pp. 981-1004, 1993.

[8] Peter H. Welch and David C. Wood, *Higher Levels of Process Synchronisation*, Parallel Programming and Java, Proceedings of WoTUG20, IOS Press, pp. 104-129, 1997.

[9] A.M. Jones, N.J. Davies, M.A. Firth and C.J. Wright, *The Network Designer's Handbook*, IOS Press, 1997.

[10] Adapteva, *Epiphany Architecture Reference*, REV 14.03.11, page 13, 2013.

[11] Zain-ul-Abdin, *Programming of Massively Parallel Processor Arrays*, Centre for Research on Embedded Systems (CERES), Halmstad University, Sweeden.

[12] http://www.wotug.org/

[13] https://developer.nvidia.com/mpi-solutions-gpus