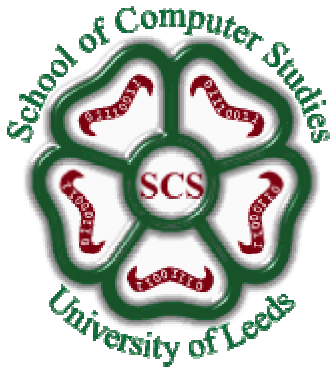
SMART OPTOELECTRONIC NETWORKS FOR MULTIPROCESSORS

B. Layet[a], I. Gourlay[b], P. M. Dew[b] and J. F. Snowdon[a]

[a] Dept. of Physics, Heriot-Watt University, Edinburgh, EH14 4AS, U. K.

[b] School of Computer Studies, Leeds University, Leeds, LS2 9JT, U. K.

Heriot-Watt University



Introduction

This work, which is still at an early stage, is focussed on identifying novel architectural enhancements that optoelectronic interconnect technology may enable within high performance multiprocessor computing systems.

Key question is how to harness the capabilities of optoelectronic interconnect technology? Including:

- High bandwidth
- Fine-grain processing capabilities - ‘smart-pixels’

Awareness of current trends in high performance computing is important. A significant trend is the emergence of the computational cluster:

- Based on off-the-shelf PCs or workstations
- Cost-effective high performance (when dynamic memory management not important)

Motivation for our Research

In the next few pages we outline the thinking that identifies a particular role that optoelectronic interconnects could play in multiprocessors. We briefly discuss:

- What are the requirements of the applications programmer and how are these satisfied? ➔ Shared Abstract Data Type.
- How are these supported at a low-level? ➔ Concurrent atomic operations.
- How a smart optoelectronic network might enable these operations.

And, of course, as well as ease of use, high performance must be enabled.

Shared Abstract Data Types (SADT)

- Extension of Abstract Data Types to include concurrency.
- Hide issues of communication and synchronisation from the application programmer.
- Useful, for example, in process management (dynamic load balancing).

Example: Travelling Salesman Problem (TSP)

This is a branch and bound algorithm that operates by expansion of a tree of nodes. A heuristic used to determine node most likely to contain optimal tour. Each processor removes a node from a queue and either expands, prunes or updates solution.

- Nodes of tree are held in a prioirity queue SADT
- Accumulator SADT used to note the best tour so far.

Concurrent atomic operations

These are low-level operations such as integer addition or the exchange of the contents of a word with a new value. They serve as building blocks for efficient SADTs and can enable high performance.

Example: Barrier synchronisation

- Serial bottleneck results if a lock-based mechanism is used to access a shared variable (which counts the number of processes that have reached the barrier). Scalable performance is obtained if concurrent access is possible.

Example: Scalable shared queue (SADT) [1]

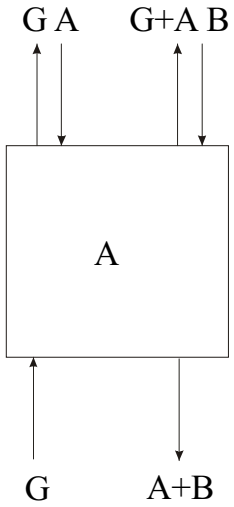
- Again an efficient implementation makes use of concurrent atomic operations to permit concurrent access to the queue.
- Use as concurrent Priority Queue to perform load balancing of tasks between processors

Smart optoelectronic network

Use of distributed network intelligence within the optoelectronic interface to enable architectural enhancements.

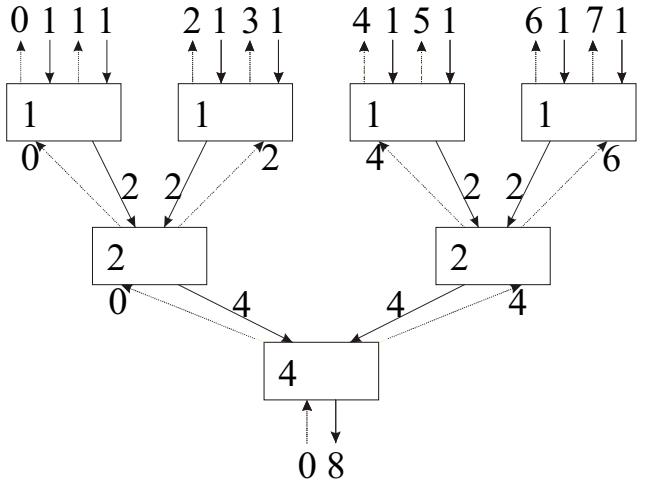
For example:

- Direct support for combining of concurrent atomic operations [2]



read&add switch

(G = global variable, A and B are initial input values)

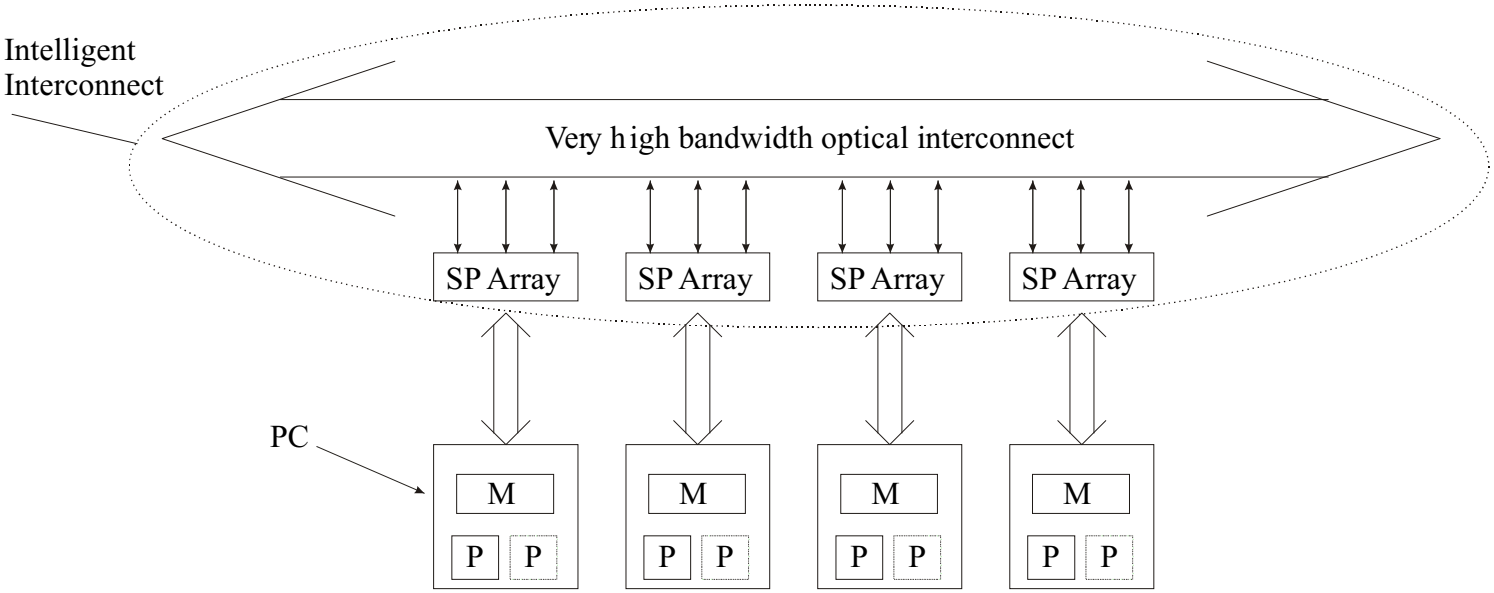


combining network

(performing array index assignment)

System architecture

- Coarse-grain computation on PCs or workstations
- Fine-grain computation ‘in the network’ i.e. in the smart pixel network interface – provides support for concurrent atomic operations



Key feature of the architecture

- Permits off-loading of the process management onto the fine-grain layer, leaving the main processors to perform the actual computation.

Recall the TSP. Process management and computation are cleanly separable:

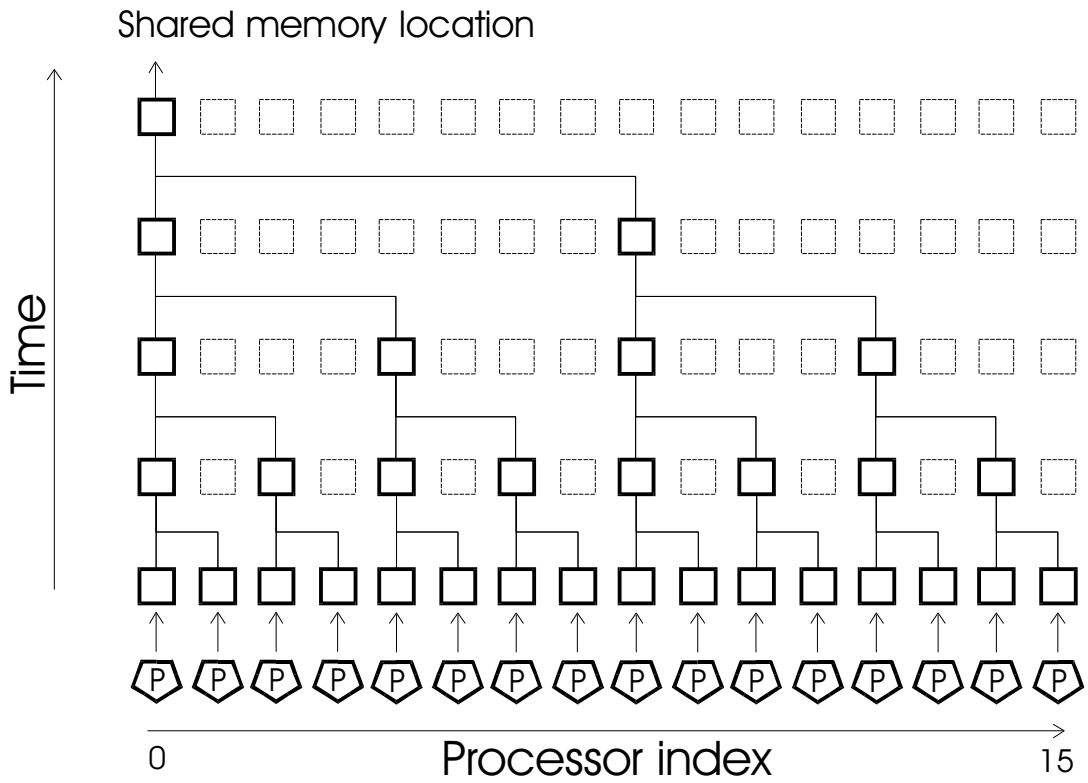
- Fine-grain layer performs process management (maintaining a tree of tours)
- Coarse-grain layer performs computation (evaluation of the cost of tours)

Work is currently in progress to identify an appropriate computational model for this system architecture (e.g. a variant on Bulk Synchronous Parallel (BSP) model) and to evaluate the parameters of the model (e.g. communication bandwidth, network granularity). For the latter, a specific system architecture is needed, and a low-level system model or set of models. A suitable model of the optics has been developed. This will guide the detailed design of the system architecture.

A combining tree in smart-pixels

Opposite is a mapping onto 16 SPAs of a tree with its root at node 0. Each SPA holds a part of the 16 trees with roots at nodes 0-15. 2x1 combining switches are used.

Alternative approach maps a multistage interconnection network onto the SPAs. Each SPA requires a 2x2 combining switch [3] for each stage.

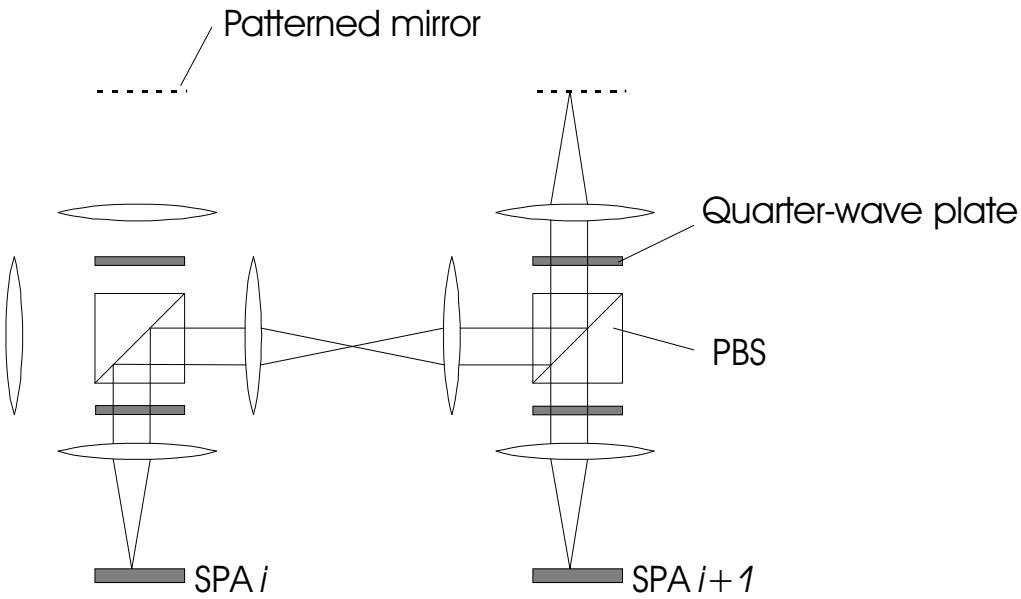


Still another option is a software combining tree (the commonly taken approach). This might exist in ‘smart memory’ tightly coupled to the SPA, if not distributed within it.

Optical system

Familiar free-space optical system using polarisation beam-combination. Can cascade the unit shown opposite to connect as many SPAs as desired.

- Free-space optics in principle provides best performance possible
- If *this system* is modulator-based then it is unidirectional with nearest-neighbour links only.
- Other technologies permit bidirectionality (VCSELs) and long-distance links (patterned waveplates).



(SPA = smart-pixel array)
(PBS = polarisation beam-splitter)

In fact, a hybrid system with microlens arrays next to SPAs is studied.

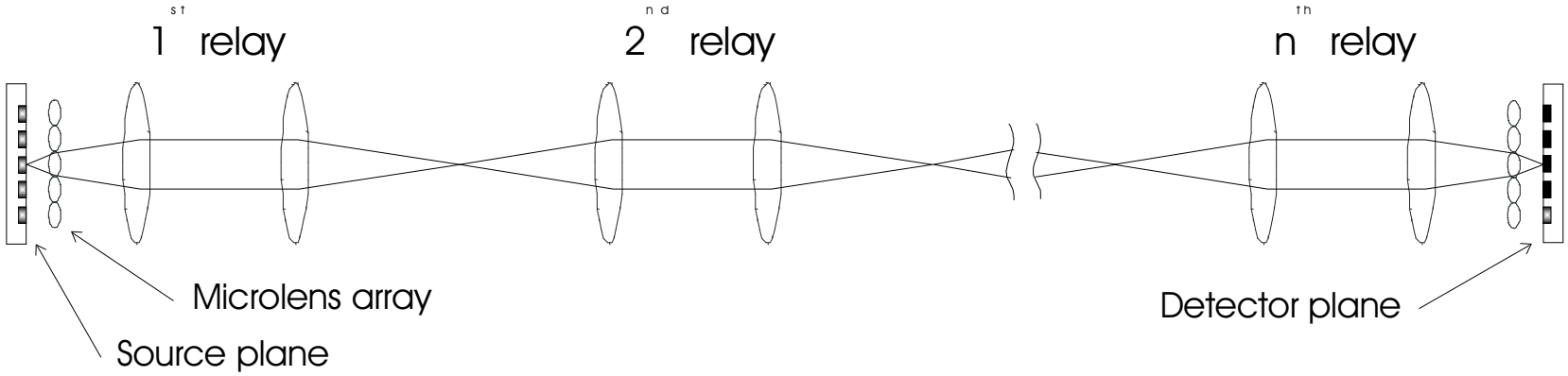
Low-level optics modelling

Path of beam from source to detector shown below.

- Macrolens image relays modelled in Code V. Fabrication tolerances of macrolenses are modelled, lenses are aligned exactly. Many samples taken; find 0.95 percentile values of performance metrics.
- Microlenses modelled using Gaussian beam propagation. M^2 parameter used in final stage.

Determine the variation with number of relays, n, of

- Macrolens performance: Spot size, spot shift range (distortion)
- Hybrid system performance: No. channels (μlens diameter), required detector size



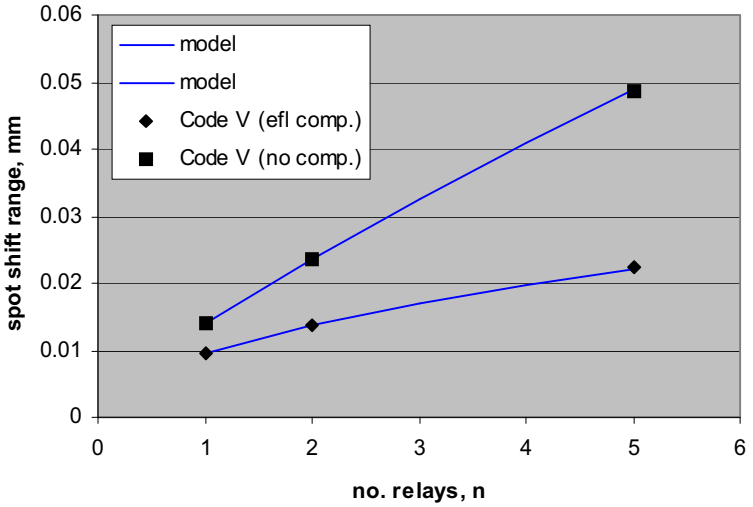
Development of a simple model for macrolens system for $n > 2$

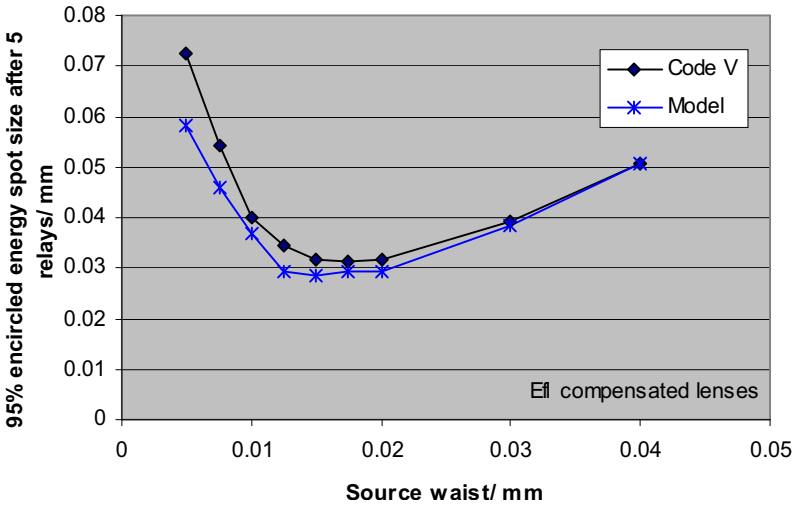
For spot size use: $s^2 = g^2 + d^2$,

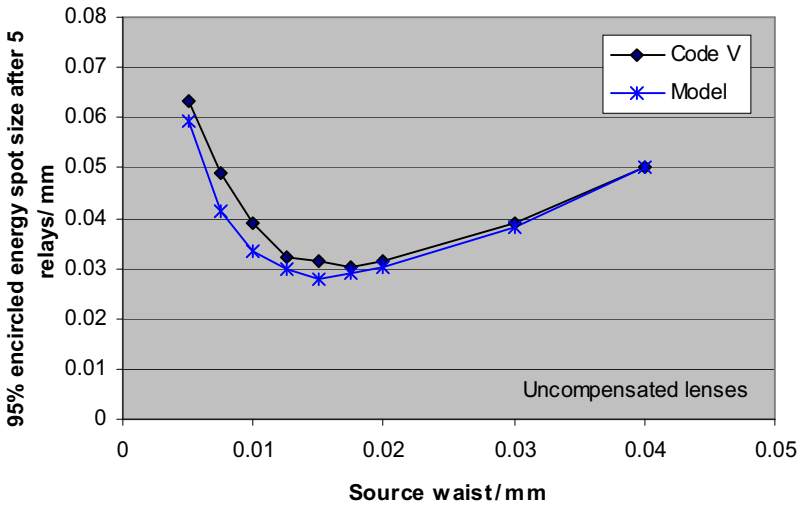
g = geometrical spot size

d = diffraction-limited spot size.

Random errors grow as $\sqrt{n}$, systematic errors grow as n. Determine constants from $n = 1, 2$ results.

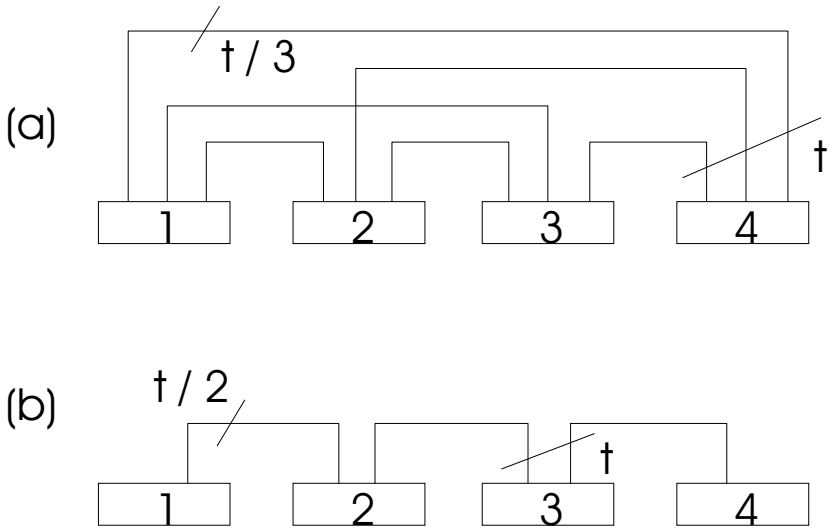


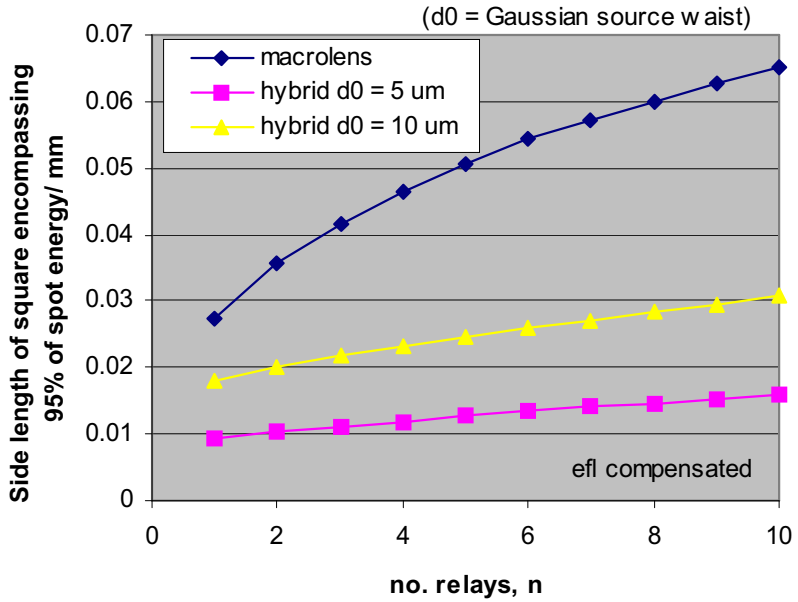




Area bounding 95% of spot energy

- Determined by both spot size and spot shift.
- The number of beams within the field of the optics is greater than the feasible number of optoelectronic i/o ports, *t*, even for large *n*.





- A (physical) topology with longer distance links uses some spare capacity.
- Consider 4-node networks opposite a) completely-connected b) linear array

Optimum interconnect topology (physical topology)

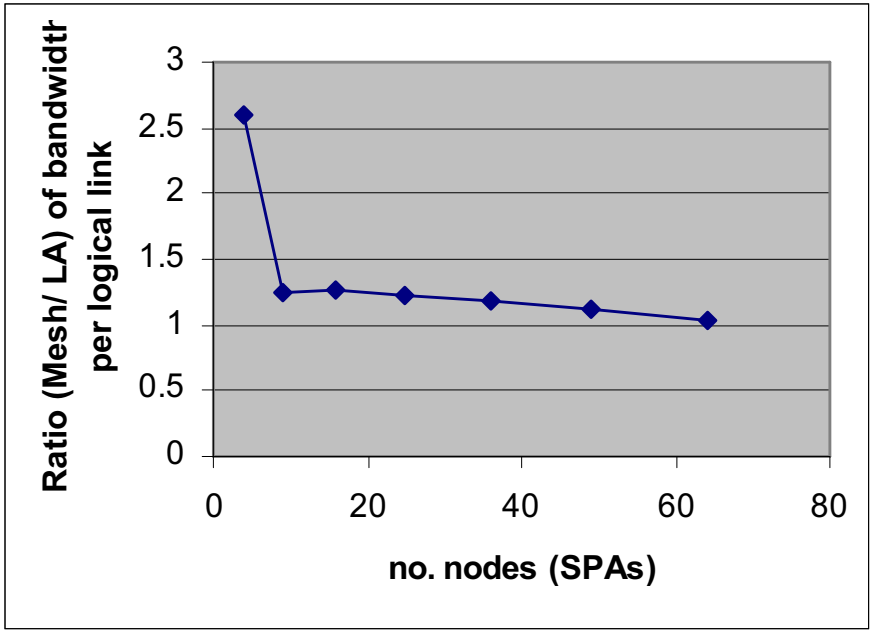
Compare physical implementations with and without longer distance optical links. Because higher connectivity topologies do not scale so well consider practical examples of "mesh" and linear array (LA). Recall that these must be embedded in the 1-dimensional physical structure of the optics.

Comparison of mesh and LA based on:

- Embedding mesh logical topology into these physical topologies
- Compare bandwidth of each logical link. Assume bandwidth depends on capactive charging of detector,

$$bandwidth \propto \frac{optical\ power}{\det ector\ area}$$

For most systems the mesh shows a small advantage. Latency is also better.



Conclusions and Future work

- Free-space optics can relay more beams than the number of i/o ports on a chip. Multipoint interconnect with entirely optical pathways uses spare capcity.

We have a model of the optics for system design. We must choose models for
- Optoelectronic devices
- Electronic sub-systems

Thus enabling performance of the selected intelligent interconnect to be assessed.

These results will interact with the high-level work on algorithm design and the specification of a set of primitives that should be supported. Ultimately, a discrete-event simulation, using a selected computational model with parameters fed upwards from low-level models, will evaluate algorithm performance.

References
[1] J. M. Nash et al. "A Scalable Shared Queue on a Distributed Memory Machine", The Computer Journal **39**, 483 (1996)
[2] A. Gottlieb et al. "Basic Techniques for the Efficient Coordination of Very Large Numbers of Cooperating Sequential Processors", ACM Transactions on Programming Languages and Systems **5**, p. 164 (1983)
[3] S. R. Dickey "Systolic Combining Switch Designs", PhD Thesis, New York University (1994)