

HPCS Program @ Sun

- Jim Mitchell, Sun Fellow
PI, Sun's DARPA HPCS Program

What Sun Learned About Peta-Scale Computing During HPCS Phase II

- HPCS program overview
- Hero architectures & technologies
- The big issues for Petascale computing

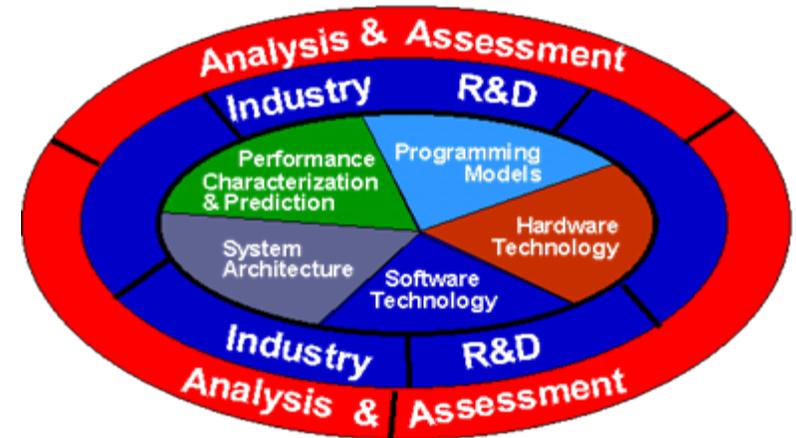
High Productivity Computing Systems

Goal:

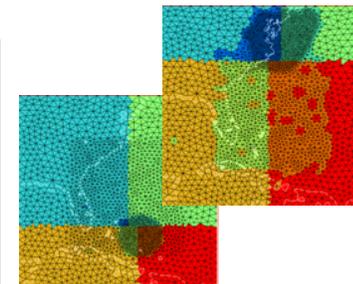
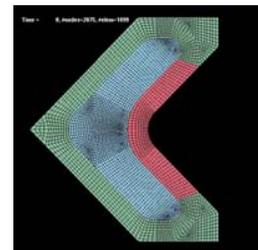
- Provide a new generation of economically viable high productivity computing systems for the national security and industrial user community (2009 – 2010)

Impact:

- Performance (time-to-solution): speedup critical national security applications by a factor of 10X to 40X
- Programmability (idea-to-first-solution): reduce cost and time of developing application solutions
- Portability (transparency): insulate research and operational application software from system
- Robustness (reliability): apply all known techniques to protect against outside attacks, hardware faults, & programming errors



HPCS Program Focus Areas



Applications:

- Intelligence/surveillance, reconnaissance, cryptanalysis, weapons analysis, airborne contaminant modeling and biotechnology

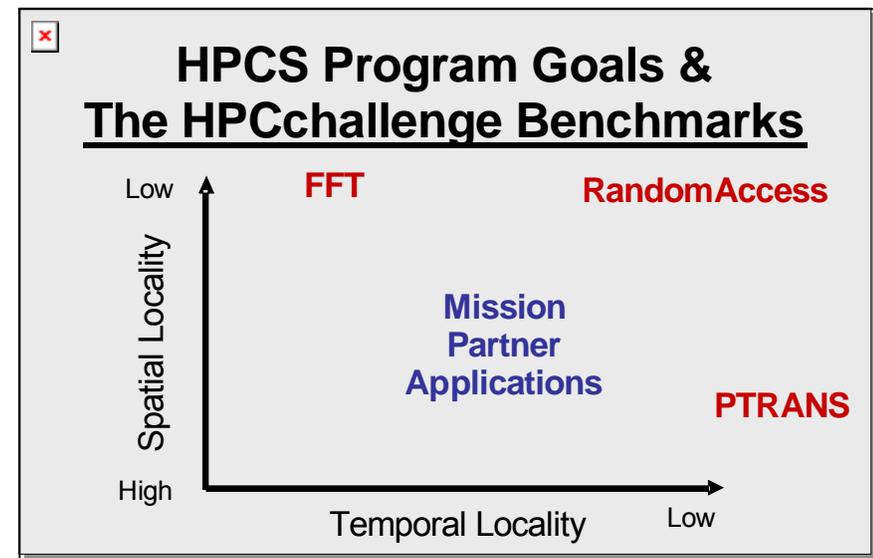
Fill the Critical Technology and Capability Gap

Today (late 80's HPC technology).....to.....Future (Quantum/Bio Computing)

DARPA HPCS Challenge Benchmarks

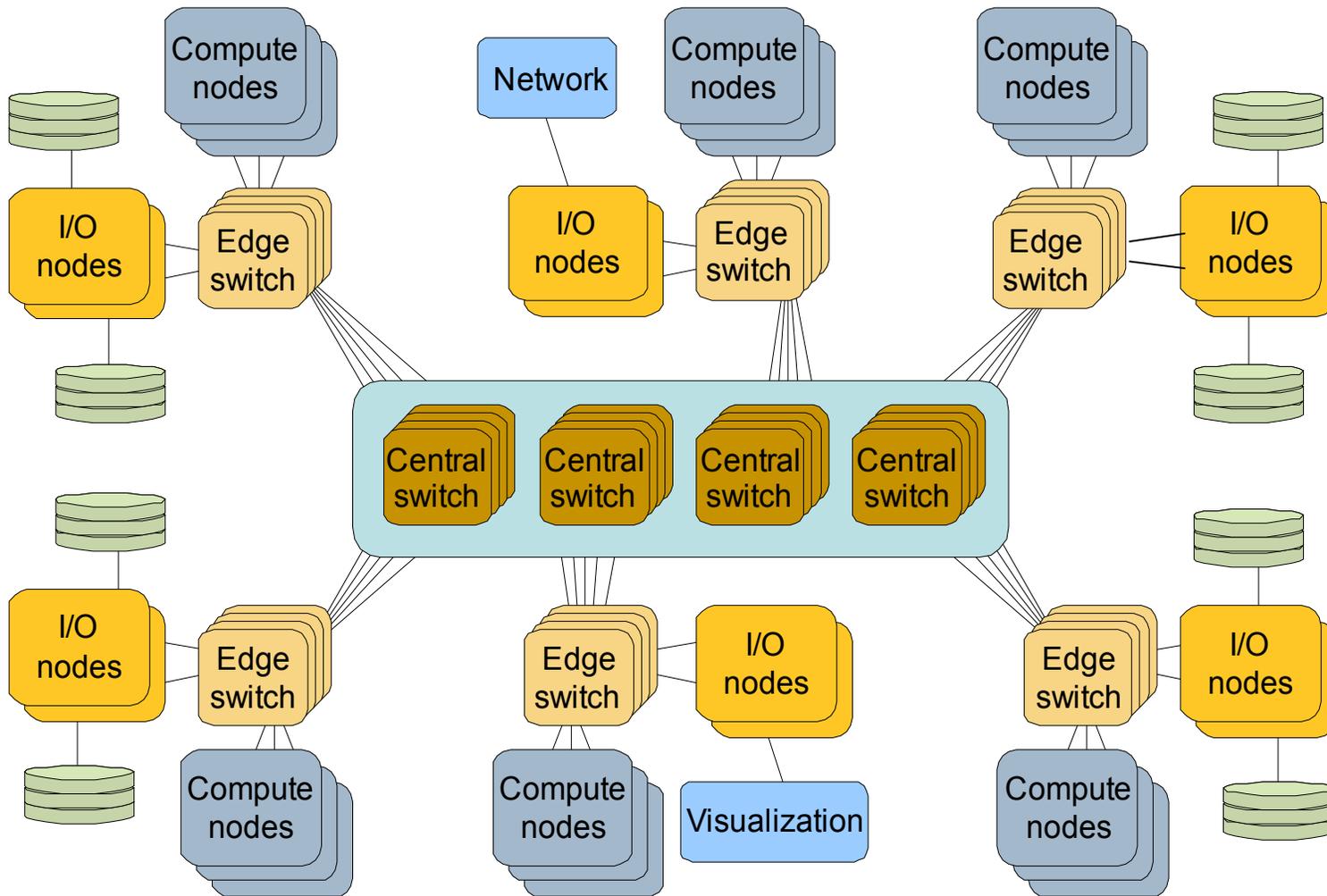
- 4 Benchmarks showing the corners of the envelope
 - > Flops (Linpack)
 - > 2 Petaflops/s sustained (3.2 PetaFlops raw)
 - > GUPS (TableToy)
 - > 64,000 Giga updates/s
 - > Local memory BW (Stream Triad)
 - > 6.5 PetaByte/s bandwidth (2B/Flop)
 - > Bisection BW (PTrans)
 - > 3.2 PetaByte/s, Parallel Matrix Transpose (1B/Flop)

- Challenges are formidable when achieved simultaneously



Hero Architectures & Technologies

Sun's Hero Hardware Architecture



Many Embedded Sun Technologies

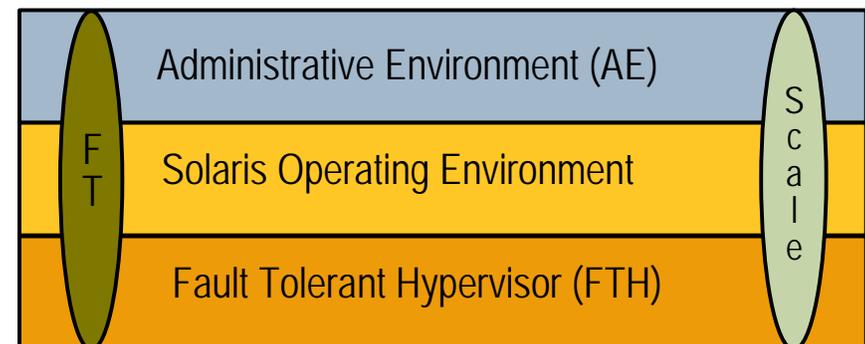
- Proximity Communications (PxC): chips talking to chips without wires
- Electrical and WDM optical on silicon using standard CMOS
- Low latency, high bandwidth interconnect using PxC and optics
- Packaging
- CMT: Lots of threads per chip
- Operating Systems (Open Solaris)
 - > Scale, system management, fault management, fault tolerance (computing correctly through failure), checkpointing
- Compilers and automatic parallelization
- Languages: Fortress, PIL, etc.
- File Systems and Object-based Storage
- Simulation, modeling, productivity and performance analysis
- ...

Execution Model Enabled by Interconnect

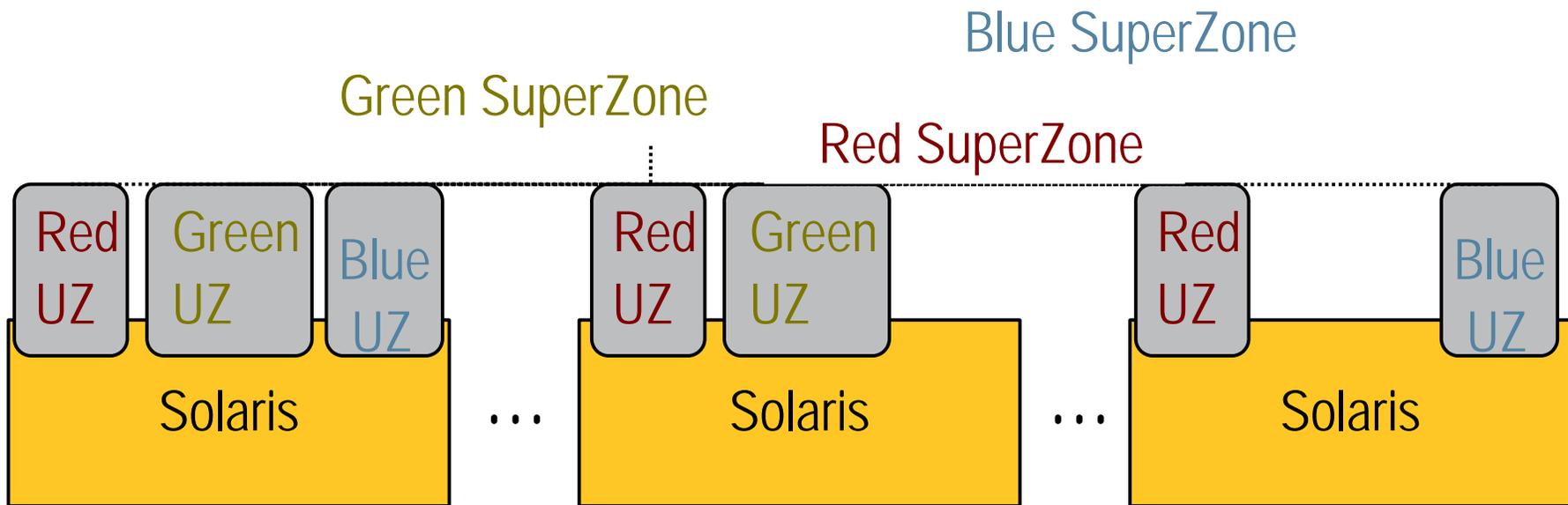
- **Thread rich environment**
 - > Extensions of innovations in T2000 Niagara and Rock CMT processors
- High bandwidth, low latency interconnect is key to scale
- Global load & store operations
- Active messages for global operations
 - > Global barriers, GUPS, MPI, ...
- Support for high productivity programming
 - > PGAS model
 - > Auto-parallelized Fortran & Fortress
- **Storage (and file systems) that scale**

System Software Architecture

- A complete system software stack on each node
 - > Hypervisor with extensions for multi-node awareness
 - > Solaris OS with extensions for multi-node awareness
 - > Administrative Environment services
- Across entire system
 - > Unification by inter-nodal cooperation
 - > Administrative Environment
 - > Name spaces
 - > Processes, Memory, IO, Files
 - > Fault tolerance
- Effective at scale because of thread-rich hardware and massive interconnect



Capacity & Capability in One System



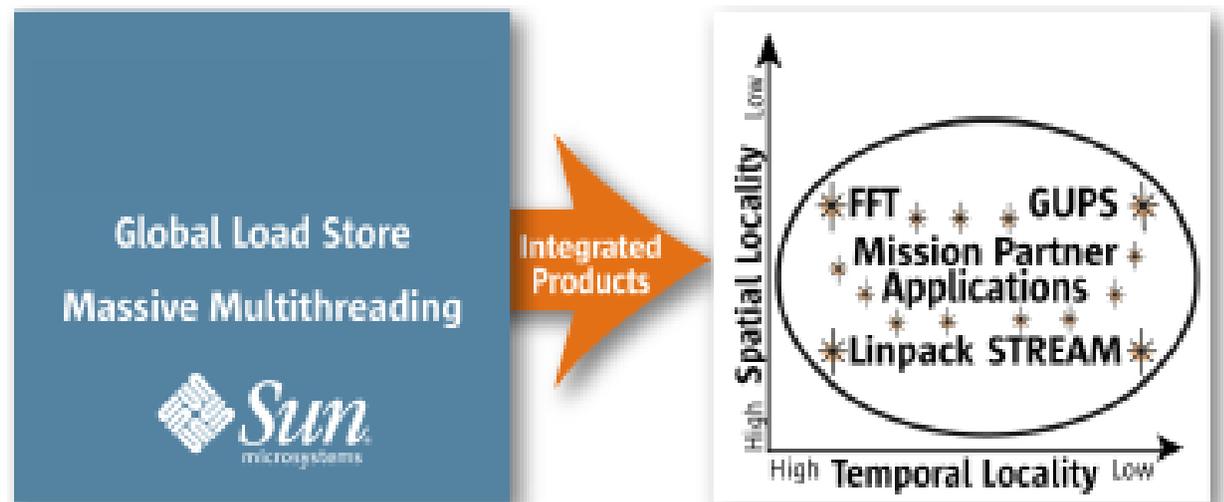
- HPCS SuperZones extend Solaris for HPCS applications
- Many SuperZones possible per system

Program Execution, Fortran & Fortress

- Programming systems
 - > Improvements in static compilation under way
 - > Sun's compilers already achieve 80% of performance of tuned MPI code
 - > Allow automatic parallelization, common PIL binaries
 - > Run-time discovery of exploitable parallelization increases performance
- Fortran studies lessons
 - > Phase II experiments on seven benchmarks show code size reductions from 3x to 10x (5x typical)
- Fortress
 - > Build on experience with Java
 - > Design for language growth (support library writers)
 - > Exploit traditional mathematical notation
 - > Make parallelism easy to use

Beating the HPCS Challenges

Benchmarks	HPCS Goals	Current Best	Evolutionary Trend	Large Sun Hero System	Full-up Hero System
GUPS	64,000	35	350		
Bisection BW PB/s	3.2	0.0	0.1		
STREAM PB/s	6.5	0.1	1.0		
G-HPL PF/s	2.0	0.2	2.0		



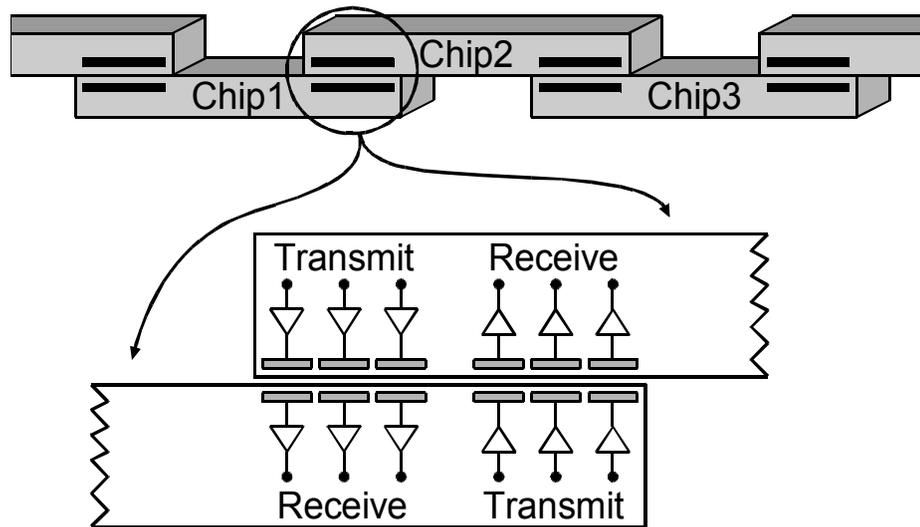
The Big Issues For PetaScale & Beyond

- Scale and complexity dominate
- Interconnects will provide Global Load/Store
- Dynamic application reconfiguration needed
- Automated checkpoint/restart is needed
- Programming productivity needs to get a lot better
- For legacy MPI-based software
 - > MPI collectives will be faster
 - > Reconfiguration on restart needed for capability apps
- For PGAS-based software
 - > Auto parallelization via compilers and runtime support

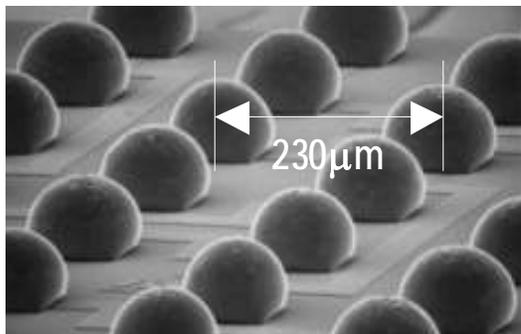
HPCS Program @ Sun

- Jim Mitchell, PI
- Sun's DARPA HPCS Program
- jim.mitchell@sun.com

Look at Proximity Communication

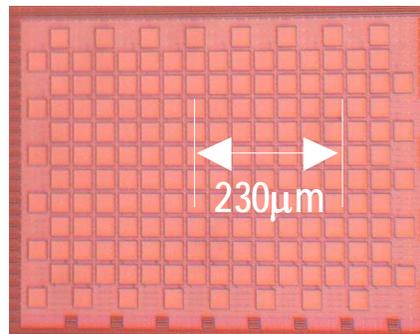


Area solder balls



flipchips.com

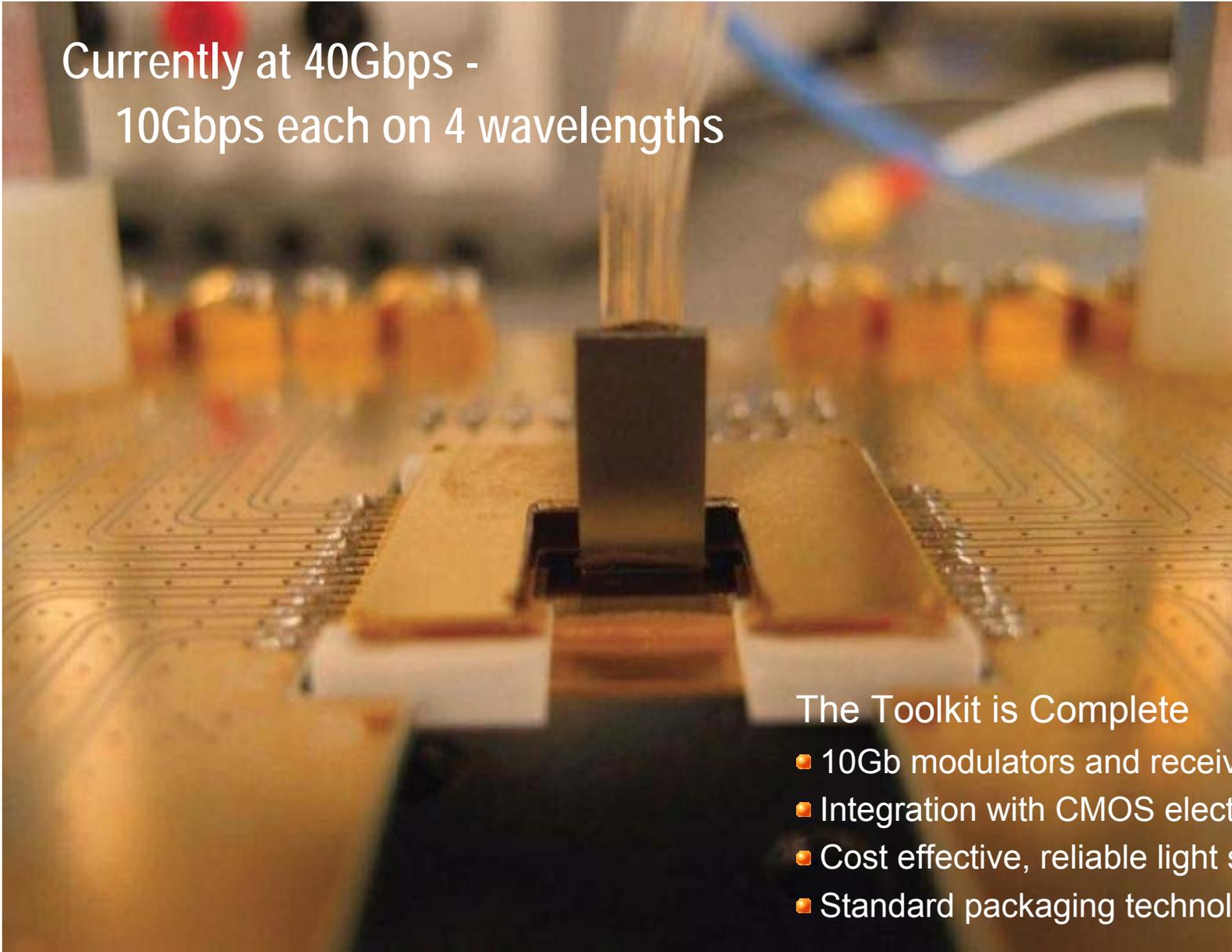
PxC array



TreasureIsland

Characteristic	Advantage
Increases BW/area	Performance
Obviates ESD protection	Power
Shrinks transmitter circuits	Power
Replaces SerDes circuits	Power
Enables smaller chips	Yield/cost
Enables replaceable chips	Yield/cost

Luxtera CMOS Photonics Module at Sun



Currently at 40Gbps -
10Gbps each on 4 wavelengths

The Toolkit is Complete

- 10Gb modulators and receivers
- Integration with CMOS electronics
- Cost effective, reliable light source
- Standard packaging technology

Fortress: Mathematical Notation

Reduce “notational gap” between code and specification

- High programmability
- Easier maintenance

NPB 1 CG Specification

```

z = 0
r = x
□ = rT r
p = r
DO i= 1,25
    q = A p
    □ = □/□pT q □
    z = z□□p
    □0 = □
    r = r - □q
    □ = rT r
    □ = □/□0
    p = r□□p

```

ENDDO

compute residual norm explicitly: $\square r \square = \square x - A z \square$

Fortress Code (Typeset Form)

```

cgitmax = 25
z : Vec = 0
r : Vec = x
p : Vec = r
□ : Elt = rT r
for j □ seq □ : cgitmax □ do
    q = A p
    □ =  $\frac{\square}{p^T q}$ 
    z := z □ □ p
    r := r - □ q
    □0 = □
    □ := rT r
    □ =  $\frac{\square}{\square_0}$ 
    p := r □ □ p
end
□ z, □ x - A z □ □

```