BAE SYSTEMS

DoD-Focused Benchmarks/Metrics, Toolchain and Debug Recommendations

for Workshop on Suite of Embedded Applications and Kernels

Jeffrey Smith, PhD

1 June 2014

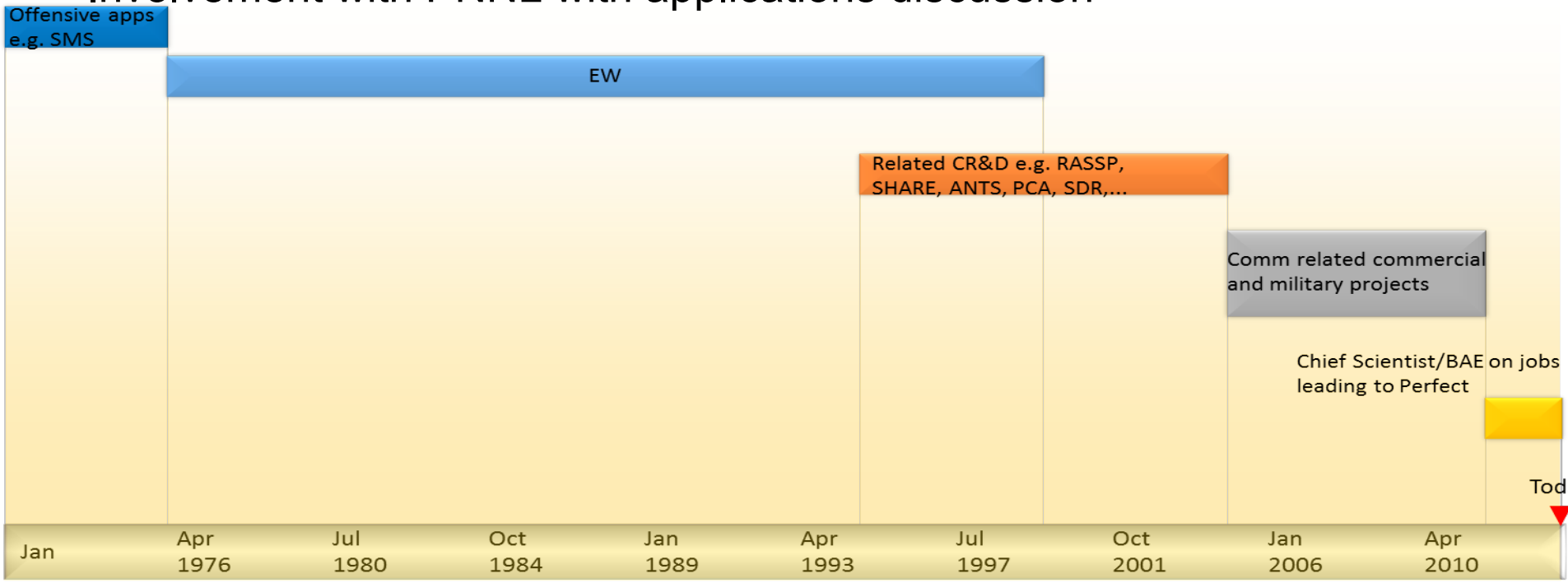




Agenda

BAE SYSTEMS

- Involvement with PNNL with applications discussion



- Points of intersection with SEAK
  - Evaluating, benchmarking and classifying embedded systems
  - Compiler, synthesizers and optimization tools
  - Modeling of application and system behavior

- Discuss lessons learned and recommendation w.r.t. above intersections

New Sensor Payloads and Higher-Level Exploitation Mission Provide Benchmark and Analysis Candidates

BAE SYSTEMS

- Wide Area Persistent Surveillance Poses Extreme Computation Challenges
  - Processing 1.8 billion pixels, at 12 fps, generates on the order of 600 Gb/s (around 6 petabytes of video data per day)

- Benchmark creation through progressively increasing processing pipeline depth and mission capacities
  - Low SWaP processing that scales with Breadth (N objects), Complexity per object (M parameters), Depth (H hierarchy levels), and Data Rates (R) of higher level exploitation missions
  - Characterize load (N, M, H, R) for **processing pipeline** stages, e.g. image pre-processing, segmentation, classification, tracking
  - SWaP constraints of tactical exploitation systems (e.g. UAVs, DCGS-A Intelligence Fusion Servers) for a range of **mission capabilities**, e.g., generating dots from pixels, tracks from dots, activity patterns from tracks, and threat analysis from patterns

- Other Challenge Domains Are Similarly Characterized
  - Software Defined Radios
  - Tactical SIGINT Payload
  - Intelligence, Surveillance & Recon
  - Distributed Networked, Adaptive Electronic Surveillance
  - Monitor Vessel Behavior
  - RT Defense Against Networked Mobile and Spectrum Dynamic Emitters
  - Continuous, Predictive Course of Action Analysis and Execution Monitoring

ARGUS-IS sensor

Persistics data processing pipeline

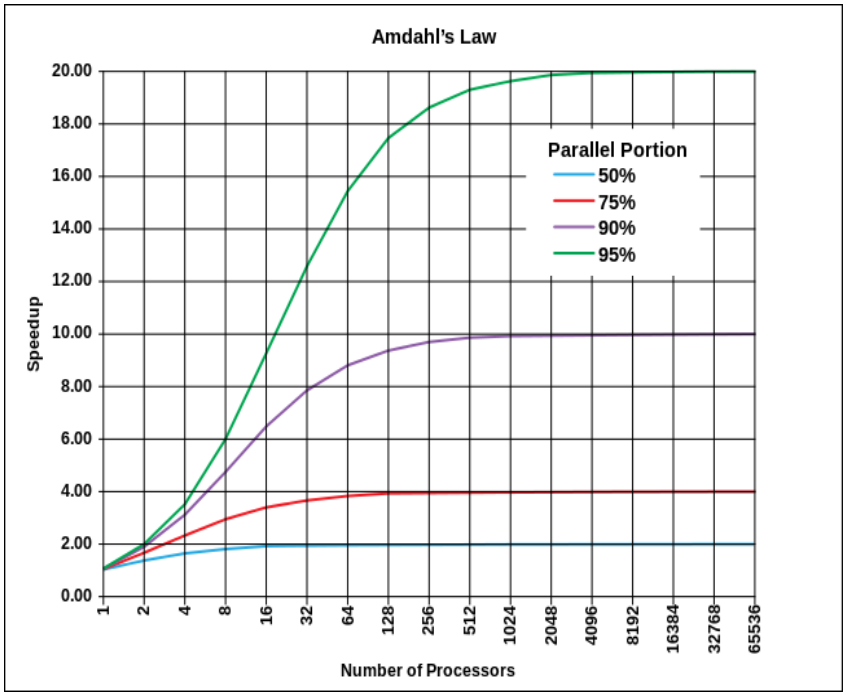
Need New Paradigm to Harness Effective Code Parallelism

BAE SYSTEMS

- ***Amdahl's Law prediction of limit of parallel processing speedup may yield low efficiency in HPC - concurrent execution of independent parallel applications may need more energy***

- ***Poor parallel application software architectures degrade performance well below Amdahl limits***
  - *Unbalanced processing load distributions*
  - *Productivity limits redesign in conventional, hand coded, developments*
  - *Balance interprocessor communication loading and data dependent wait times*
  - *Kernels depend on dataset scaling*

- ***Reverse progress from ability to automatically and verifiably generate parallel and concurrent execution of military applications from graphical specifications***

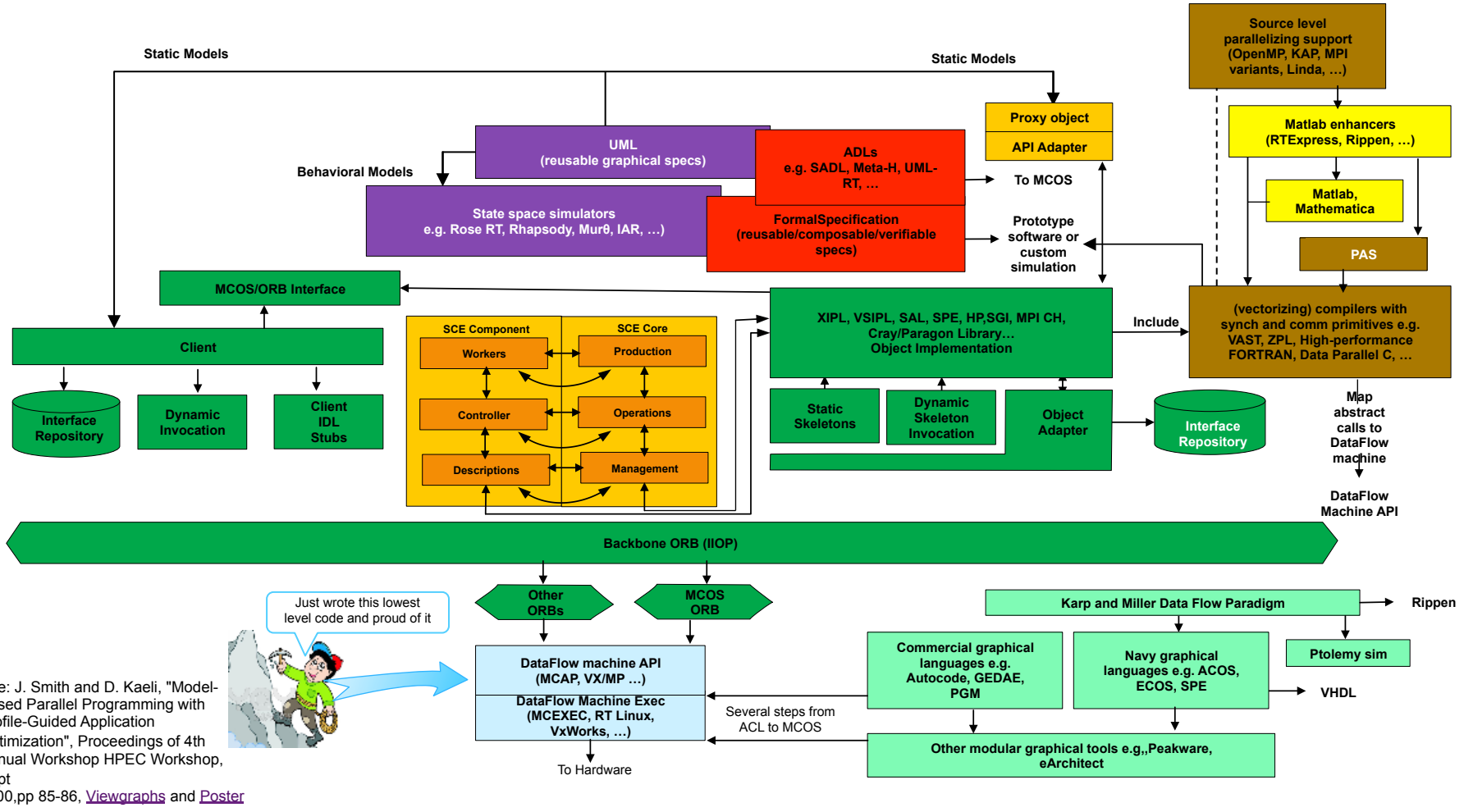


$S = 1/(1-P)+(P/N)$

Upper bound on parallel compute performance depends on exploiting ratio of serial to parallel code

Metric – common comparison between best hand-coded and tool generated benchmarks

Methods to Conceptualize/Apply High Performance Data Flow Applications



Cite: J. Smith and D. Kaeli, "Model-Based Parallel Programming with Profile-Guided Application Optimization", Proceedings of 4th Annual Workshop HPEC Workshop, Sept 2000,pp 85-86, Viewgraphs and Poster

METHOD	METHOD EXAMPLE	CONCEPTUALIZATION
Mountain Climber	Manual C/assembler/macro calls data partitioning	Direct machine level
Parallel Path/APIs and Language Constructs	MPI (variants), OpenMP, KAP, SE, Linda, PVM	Compiler supported partitioning, source level abstraction supported by parallelizing compilers – not yet connected to model
Distributed Objects	RT CORBA	Distribute data to workers, component level abstraction over mountain climber
Accelerated Objects	Agents, SCE	Reduce runtime component brokering, distribute workers to data
Accelerated Functions	Rippen, Autocode, Peakware, GEDAE, RTExpress, PGM, SPW, Khoros	Distribute control, graphical functional flow graph
Refine Abstract Machine (general purpose, data flow "port/connector", …)	SADL, Meta-H, Talaris, SPE (Ptolemy simulation refinement not ADL but similar conceptualization)	Architecture definition language
Model Based & Knowledge Capture Frameworks	UML, SAGE, Mesa	Generate SW from more abstract model level, UML can generate component IDLs & model application compliant with OMG

*** Conceptualization outliers/combos e.g. ZPL (accelerated arrays) , BlueSpec (accelerated functions and objects), Matlab (variant of data flow) "yellow grouping", …**

Toolchain based Recommendations

BAE SYSTEMS

Current

- Model generated specifications and partial code
- Peta-op computers with thousands of threads with manual code generation, verification, configuration and load balancing
- Tools proven for parallel computing but no longer supported to extend to multi-core or cloud environment
- Highly parallel computers, e.g. Tilera, with manually constructed parallelization methods e.g. MPI
- Emergence of automatic parallelization frameworks distributing common operations

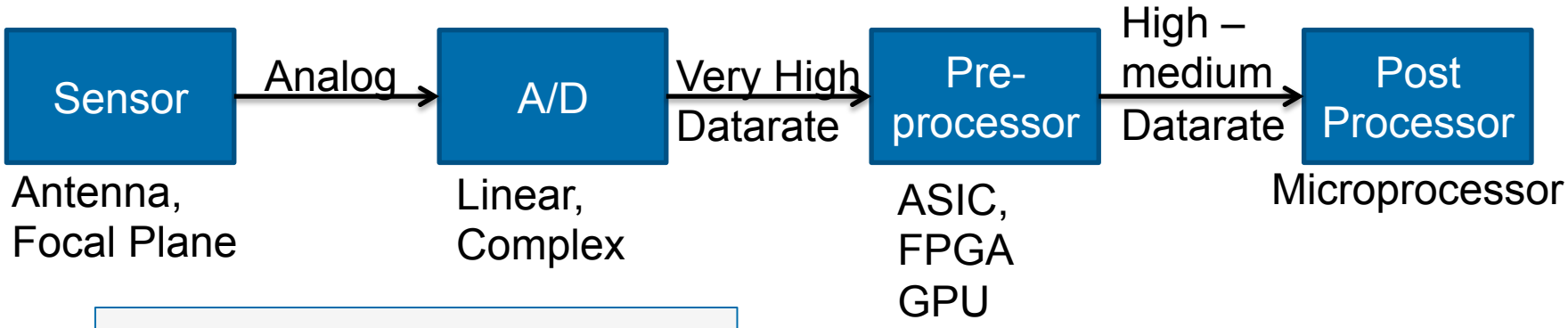
Recommendations

- Functionality captured in graphical spec to support practical design optimization - Help with how to evaluate new applications/ algorithms differently
  - *Rapid partitioning, configuration and evaluation of model driven code generation*
  - *Design iteration to near Amdahl performance limitations*
  - *Support modern RAASP-like program (metrics, support, benchmarks)*
- HP inter and intra kernel interface/ comm mechanism spanning vendor approaches
- General use, improved {pre}compiler technology (e.g. directed profile-guided optimization, adaptive compilers, etc.)

Processing Flow Hasn't Changed for EW, SIGINT & Comms

BAE SYSTEMS

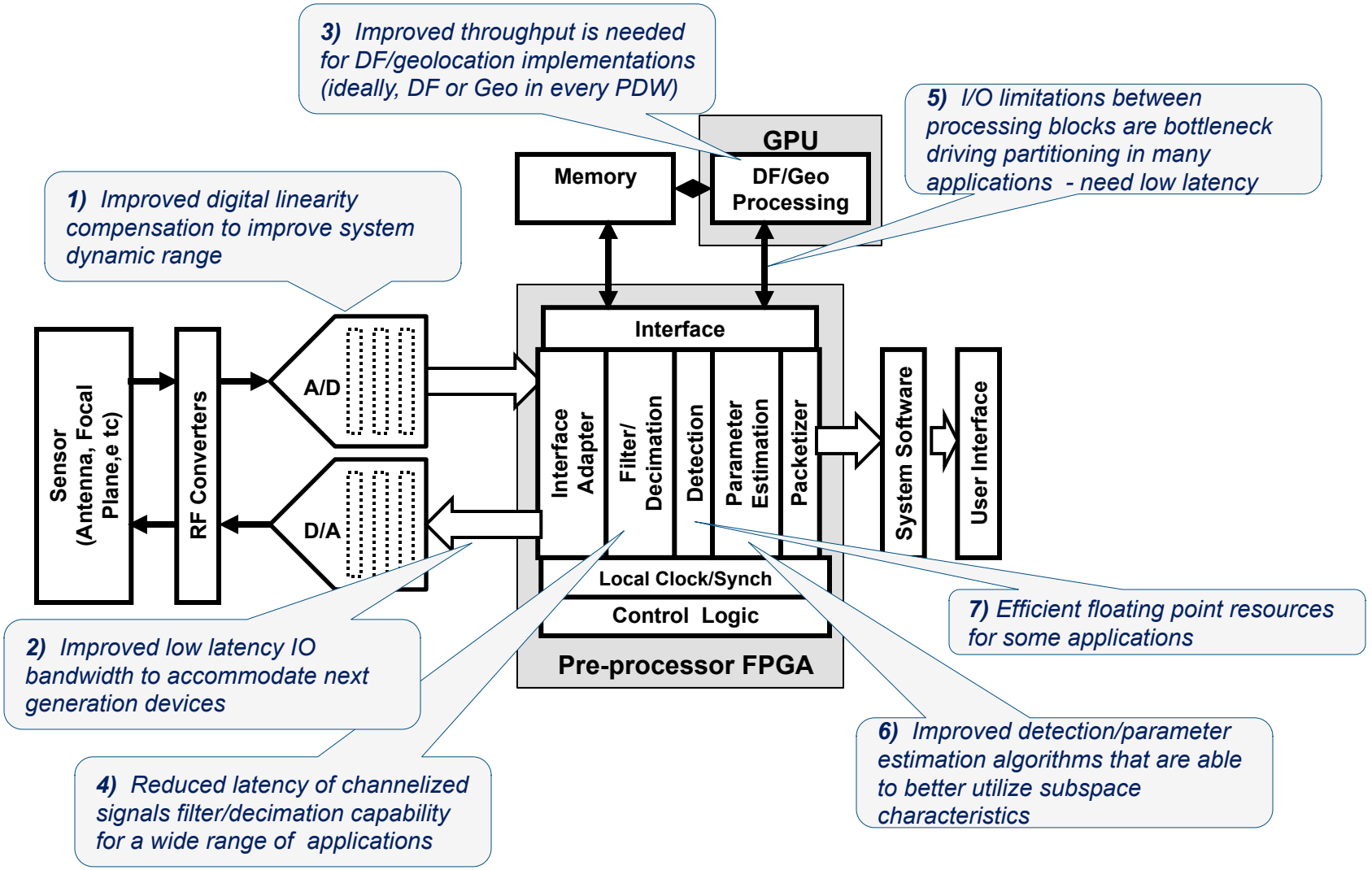
- High-rate sensors outputting multi-gigabyte data streams
  - Improvements in sensors is constantly increasing the volume of data
- Flexibility provided by digital processing is pushing A/D closer to the sensor
- Very high performance, extremely low-latency front-end pre-processing is performed to process the raw data and extract the signal/information of interest
  - Processing typically requires each data sample to be processed
- Once signal or information of interest is extracted microprocessor performs further lower-rate processing



Receive chain shown, transmit is same but in reverse

Current and Projected Digital Processing Architecture Challenges

BAE SYSTEMS



What are SEAK challenges

BAE SYSTEMS

- Non-reliable computations
  - Map to near-threshold operating regions
  - Fabrication issues
- Small reliable LWCs
- Need faster solution than end-end simulations at transistor/gate/module level
- Need to correlate high-level state with gate-level state

- Net Count = 100,000
- Depth = 1,024
- Comprehensive coverage:
  - Single faults → 100,000 * 1,024 = ~100M simulations
- Each simulation runs for ~3 * 1,024 = 3K cycles
- Each cycle is ~100,000 LUT ops
- Each LUT op is ~10 processor instructions
- → Need 100M * 3K * 100K * 10 = 3*10^(17) instructions
- One computer has ~100,000 MIPS
- → ~35 days of running time

Recommendations

BAE SYSTEMS

- Inject faults anywhere at any level
  - Need high-level simulator capable of injecting and simulating effects of (multiple) probabilistic faults at low level
    - Simulate low level effects of faults given models of low-level gates/transistors where a fault could be injected
  - Only need to run a gate level simulation cycle when we want to introduce a fault.
    - All the other cycles in a simulation can be run at RTL (or BSV)

- Correlate high-level state with gate-level state to realistically debug
  - Run bsim instead of RTL by interrupting the BlueSpec simulator to compute and insert a fault.

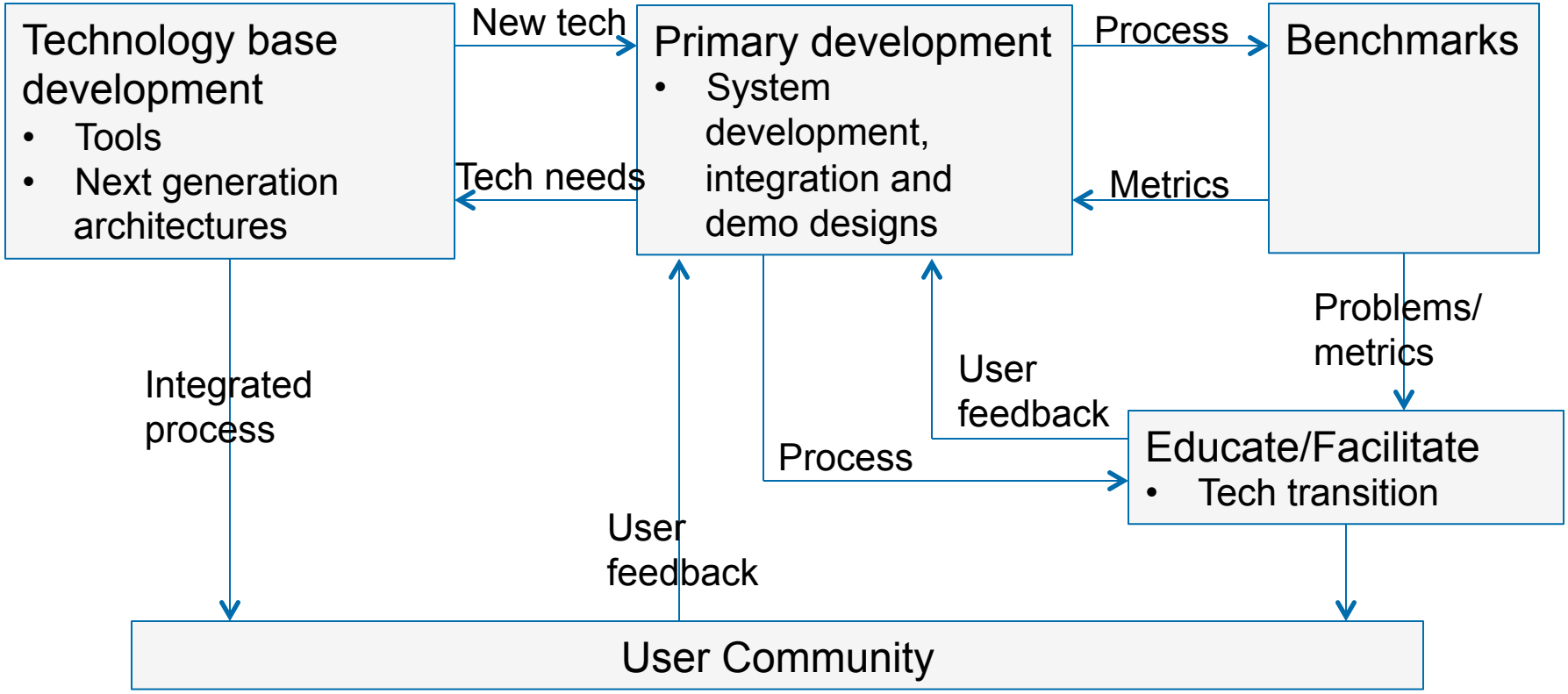
Backup

BAE SYSTEMS

RASSP Goals and Process

BAE SYSTEMS

- Reduce cost and time from concept to development and maintenance
- Main technological currency



Broad Potential: Productivity, Performance, Security

BAE SYSTEMS

Improved Cost	High-level languages & automatic parallelization aid **targeting** parallel platforms, but **do not improve debugging** of parallel programs.	Need to speed parallel system debugging & maintenance, not just development
Scaling Performance with # cores	Programmers today avoid parallelism opportunities to simplify debugging. Unsurprising with debugging for M instruction streams ≥ M times difficulty.*	Need to unleash parallel opportunities without increased cost of bugs.
Security	Most vulnerabilities due to bugs. Isolating & repairing bugs is a central element of security. Parallel bugs most difficult & rapidly expanding.	Need improvement just to keep up; leap has potential for large impact.
	Online defense techniques limited by need to be low impact to applications & systems	Need full instrumentation & control, invisible to applications for security leap

As number of cores explode & programming tools mature, debugging tractability becomes the bottleneck to realizing gains from parallel platforms; & low-impact instrumentation & control is a potential key enabler for security.

* Openshaw and Turton, *High Performance and the Art of Parallel Programming.*

Current Debugging Approaches Provide Little Support - Especially for Parallel and Optimized Codes

BAE SYSTEMS

Debugging Parallel Code	
Require serialization prior to debugging	gdb
Execute single thread until it blocks, then switch threads	MS Research's CHESS
Replace thread model with deterministic concurrency model	George Mason U's MM concurrency test and debug library
Focus on data sharing faults	Intel Go-Parallel

Debugging Optimized Code	
Turn off optimizations when debugging	gdb, Microsoft, Borland, others
Debugger "hides" transforms and provides *transparency*	Zellweger PhD dissertation, 1983
Visualize compiler transforms performed	Convex Computer, 1992
Debug optimized pseudo-code not original source	IBM mods to gdb

Programmers have little to work with

Very old results
Still in the lab
Piecemeal solutions

Poor debugging support will continue to be a major drain on programmer productivity

Why DARPA? Business Unlikely to Solve Due to Business Economics; Large DoD Payoff

BAE SYSTEMS

Debugging economics: business less sensitive to high debug costs

Debug costs per revenue dollar low for mass market commercial software

vs.

DoD measures acquisition cost/product so debug costs amortize at high rate

Also

DoD systems larger, more complex

DoD mission critical systems have to be more fully debugged

- Most commercial software can afford to have users be beta testers

Open source tools: little to no investment by business

Former development tools companies either subsumed or gone: Borland, Corel, Symantec, Rational

No venture investors will touch a proposed new tools venture: little or no expected return on investment.

Developers expect to get tools for free (e.g. Eclipse)