Adaptive Computing in NASA Multi-Spectral Image Processing

Master's Defense

Mark L. Chang

Adviser

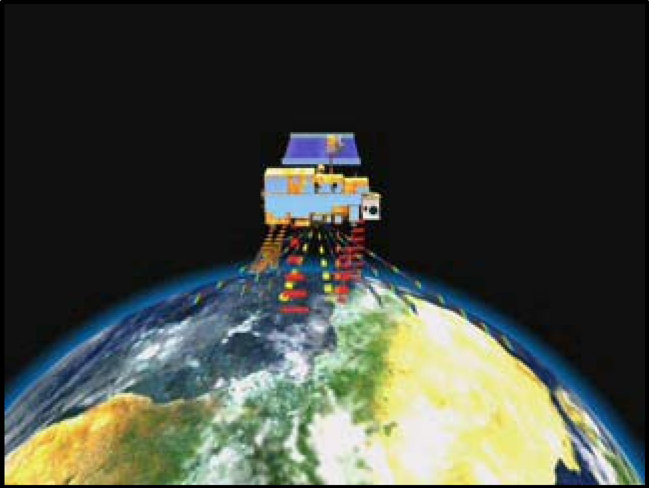
Scott A. Hauck

Observing the Earth

- NASA launches Earth Science Enterprise (ESE)
  - Initiative to study Earth as an environmental system
- Three parts:
  - Earth Observing System (EOS)
    - Fleet of Earth-observing satellites
  - EOS Data and Information System (EOSDIS)
    - Network of computers to process, store, and distribute data
  - Scientists to study data
- Goal: further understand the Earth’s natural processes and the effect humans have upon them

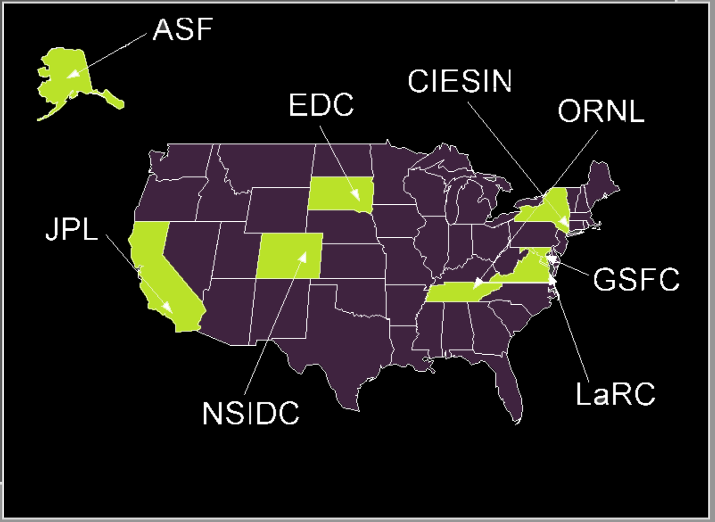
The Terra Satellite

- December 17, 1999, Vandenberg AFB
  - Launch of the first Earth Observation System (EOS) satellite
- 6-year mission to study Earth
- Characteristics
  - The size of a small school bus
  - Polar orbit
  - Carries five advanced sensors



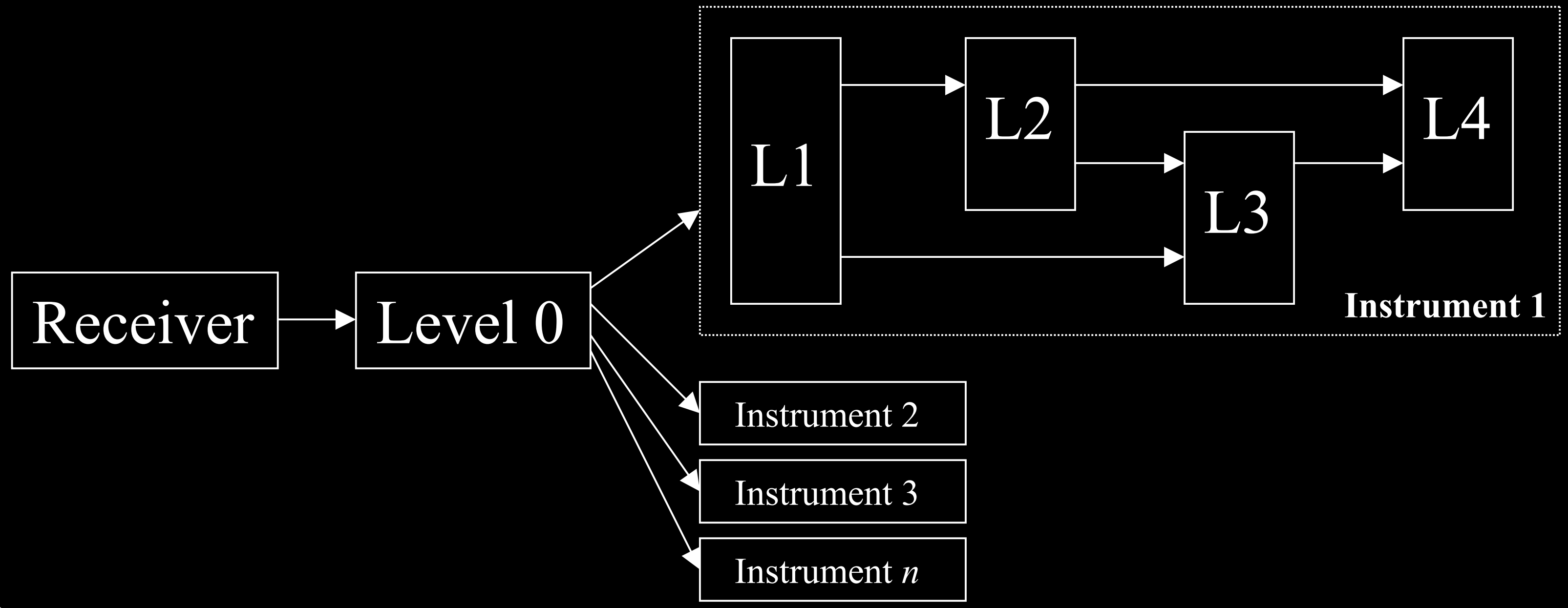
EOSDIS

- Advanced network of computing resources
  - Manages past and current Earth science data holdings
  - Data generation, archiving, distribution
- Eight Distributed Active Archive Centers (DAACs) distributed throughout the country
  - Each serves a particular Earth Science discipline
- Currently using traditional workstations and parallel computers



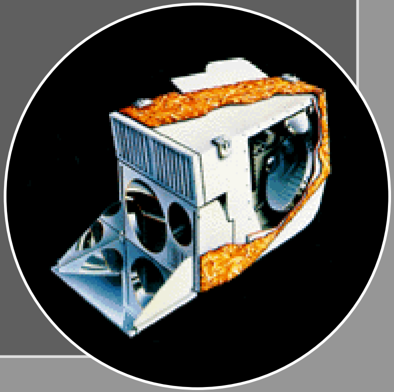
Data Flow

- EOS divides telemetry processing into five levels with the following flow:



Too Much of a Good Thing?

- Terra satellite
  - Average daily data volume: 1 TByte
  - Average processing load: 11 GFlops
- MODIS instrument aboard Terra accounts for over half the daily data and processing load
- All of EOS: (14 satellites, 28 instruments)
  - Average daily data volume: 2.5 TBytes
  - Average processing load: 34 GFlops
- Current NASA-supported data holdings total ~125 TBytes



Possible Solutions

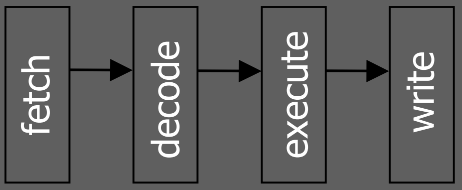
- General Purpose Processors (GPPs)
  - Intel Pentium, DEC Alpha, UltraSPARC
  - Workstations, parallel computers, supercomputers
- Special-purpose processors/coprocessors (Hybrid)
  - Digital Signal Processors (DSPs), Math Co-Processors, etc.
- Application Specific Integrated Circuits (ASICs)
  - Fully customized chips dedicated to the task

- Programmable Hardware (FPGAs)
  - Hardware speed with software flexibility

General Purpose Processor

- Software instructions direct GPP to perform arithmetic, logic, branching, and data transfer functions
- Generalized to perform any arbitrary computation

```
add    r1, r2, r3
mult   r3, r4, r5
ld     0x00fc, r6
bne    r4, r5, loop
```



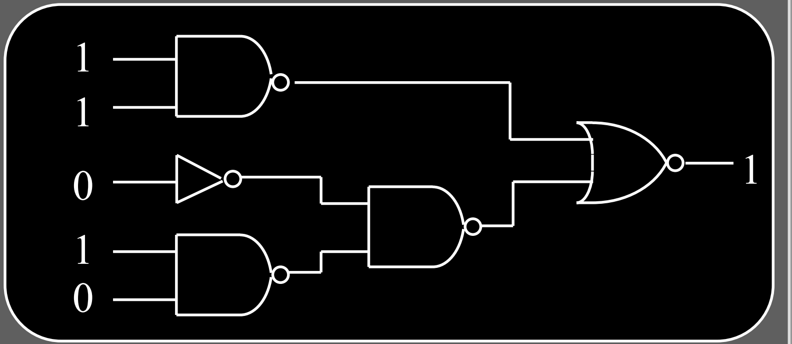
Processor Pipeline

...on the other hand

Programmable Hardware

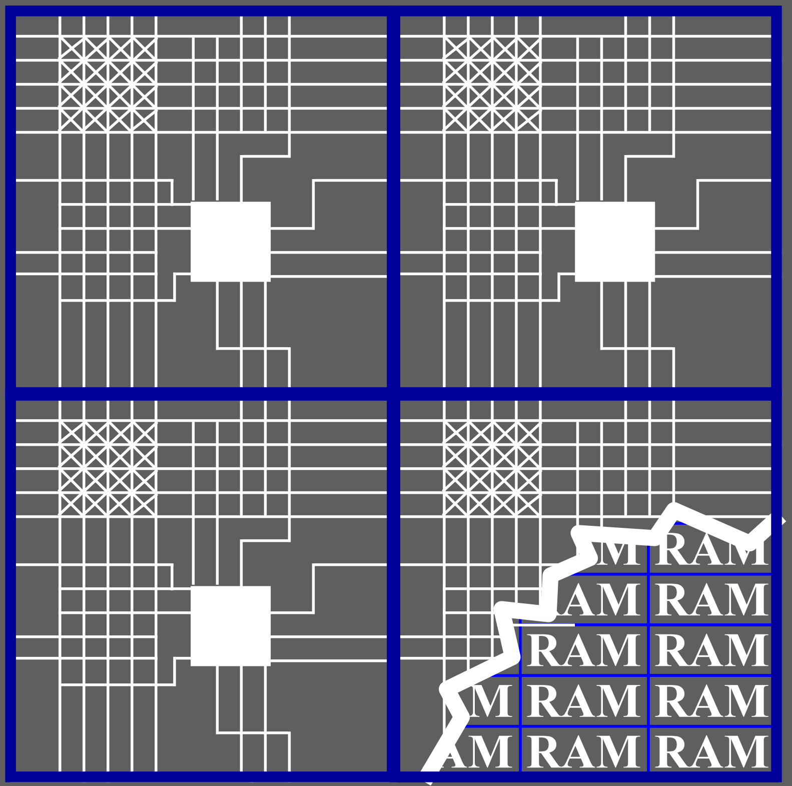
- User-customizable hardware
  - "Blank" hardware that user can program/reprogram
  - Fast like hardware, flexible like software

```
add   r1, r2, r3
mult  r3, r4, r5
ld    0x00fc, r6
bne   r4, r5, loop
```



Field Programmable Gate Arrays

- User-programmable:
  - Logic blocks
  - Interconnection fabric
- Logic block
  - Two-input Boolean
  - Full adders & ROM/RAM arrays
- SRAM-based
  - Infinitely reconfigurable
  - Relatively low programming time compared to EEPROM



Why Adaptive Computing?

- Same adaptive compute "engine" can be used for all instruments on all satellites
  - Instrument dependent processing
  - Data products involve many different algorithms
- Algorithms often change over the lifetime of the instrument
  - Calibration error, decay, damage, assumption errors
  - New algorithms and data products
- Not enough volume to offset ASIC development costs

Why Not Adaptive Computing?

- Lack of quality, easy-to-use development tools
- Current mappings are done by hand
  - Hardware description languages (Verilog, VHDL) to describe circuits
  - Manual algorithm partitioning
  - C program interface to adaptive computing "engine"
- Requires intimate knowledge of
  - Algorithm architecture
  - Target hardware architecture

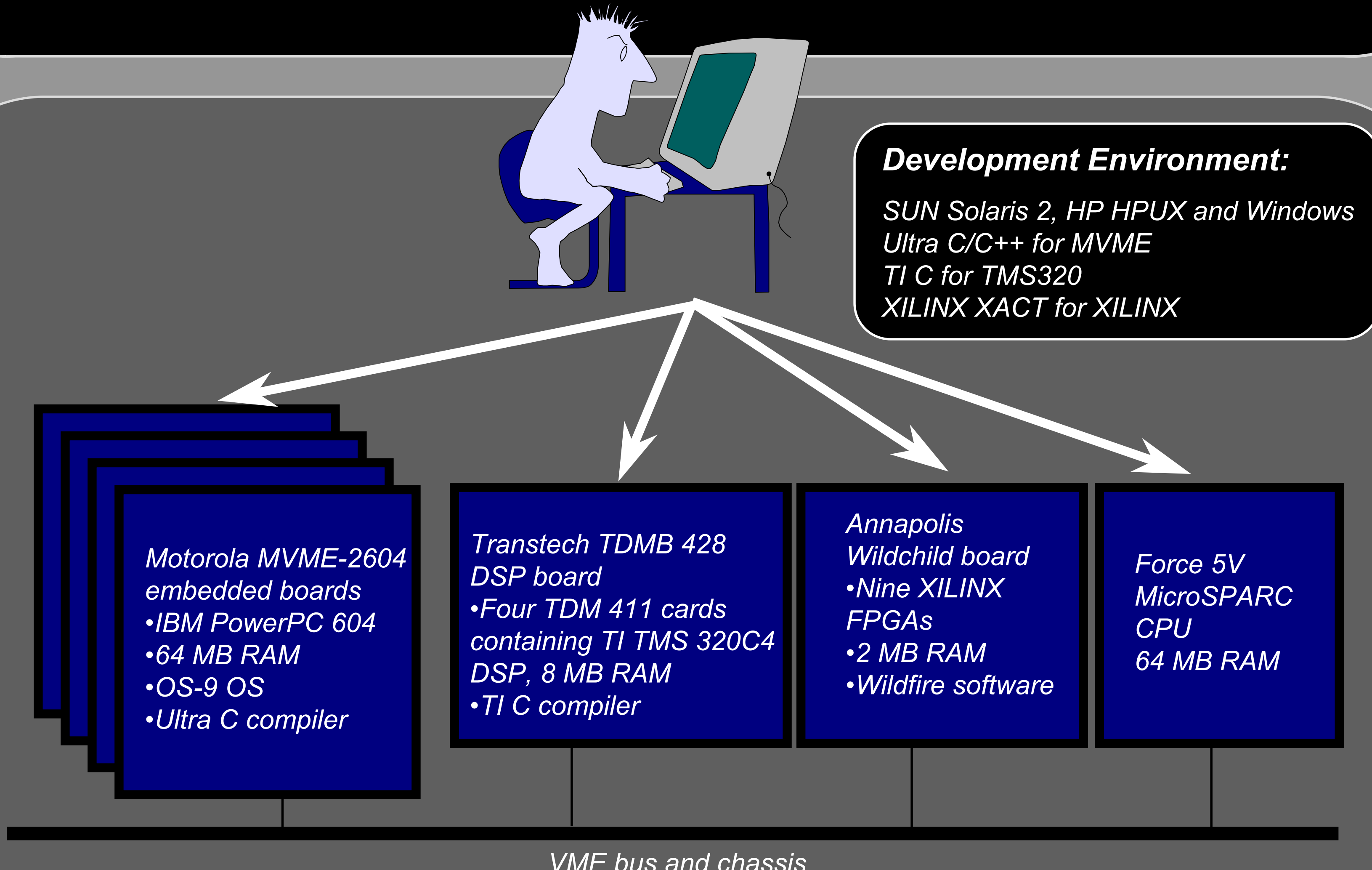
MATCH Compiler

- MATCH == MATlab Compiler for Heterogeneous computing systems
- MATLAB codes compiled to a configurable computing system *automatically*
  - Embedded processors, DSPs, and FPGAs
- Performance goals
  - Within a factor of 2-4 of the best manual approach
  - Optimize performance under resource constraints

MATCH Compiler Framework

- Parse MATLAB programs into intermediate representation
- Build data and control dependence graph
- Identify scopes for fine-grain, medium grain, and coarse grain parallelism
- Map operations to multiple FPGAs, multiple embedded processors and multiple DSP processors
- Automatic parallelization, scheduling, and mapping

MATCH Testbed



Motivation for MATCH

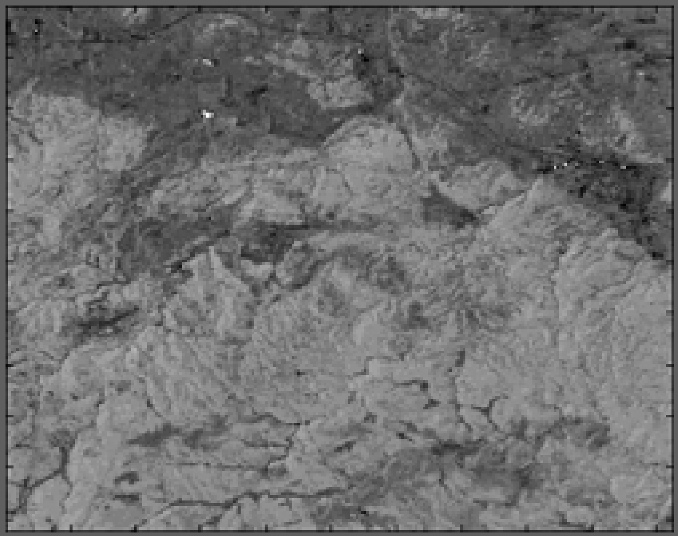
- NASA scientists prefer MATLAB
  - High-level language, good for prototyping and development
- NASA applications are well-suited to the MATCH project
  - Lots of image and signal processing applications
  - Same domain as users of embedded systems
  - High degree of data parallelism
  - Small degree of task parallelism
- NASA has an interest in adaptive technologies (ASDP)

How Can I Help?

- Research and develop “driver” applications
- Benefits compiler development in many ways
  - Defines critical functions that the compiler should support
  - Provides sample codes that can be used in testing the compiler
  - Establish a performance baseline/Benchmark for the MATCH compiler
  - Discover possible optimization techniques that can be automated by the compiler
- Investigate NASA image processing applications

Multi-spectral Image Classification

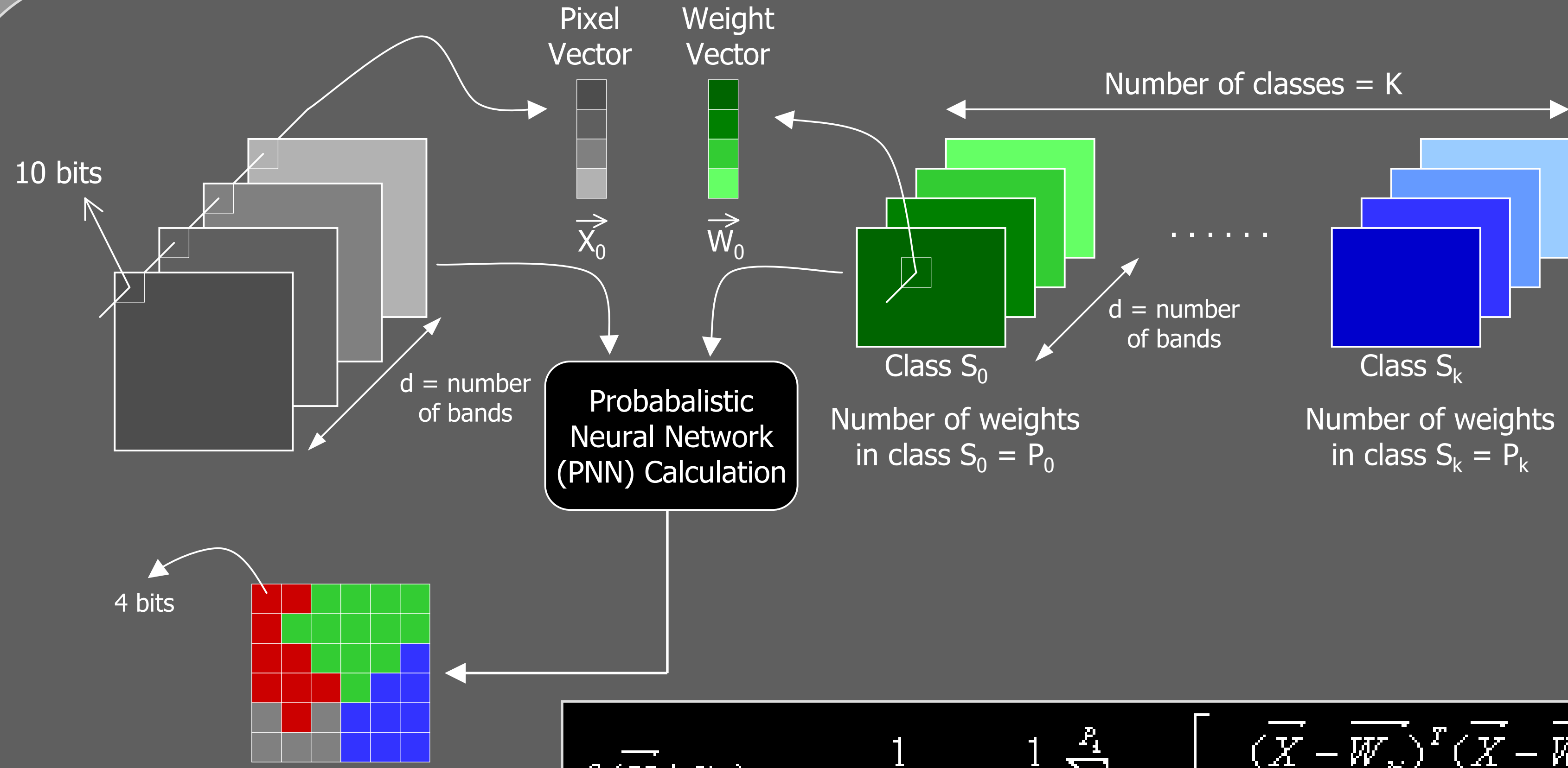
- Want to classify a multi-spectral image in order to make it more useful for analysis by humans
  - Used to determine type of terrain being represented
  - Similar to data compression & clustering analysis



→

```
Pixel[000][000] = Forest
Pixel[123][123] = Urban
Pixel[255][212] = Tundra
Pixel[410][230] = Water

etc…
```

Multi-Spectral Classification



$$f(\overline{X} \mid S_k) = \frac{1}{(2\pi)^{d/2}\sigma^d} \frac{1}{P_k} \sum_{i=1}^{P_k} \exp\left[-\frac{(\overline{X} - \overline{W}_{ki})^T (\overline{X} - \overline{W}_{ki})}{2\sigma^2} \right]$$

MATLAB Iterative

```
for p=1:rows*cols
   % load pixel to process
   pixel = data( (p-1)*bands+1:p*bands );

   class_total = zeros(classes,1);
   class_sum   = zeros(classes,1);

   % class loop
   for c=1:classes

      class_total(c) = 0;
      class_sum(c) = 0;

      % weight loop
      for w=1:bands:pattern_size(c)*bands-bands
         weight = class(c,w:w+bands-1);
         class_sum(c) = exp( -(k2(c)*sum( (pixel-weight').^2 ))) + class_sum(c);
      end

      class_total(c) = class_sum(c) * k1(c);
   end
   results(p) = find( class_total == max( class_total ) )-1;
end
```

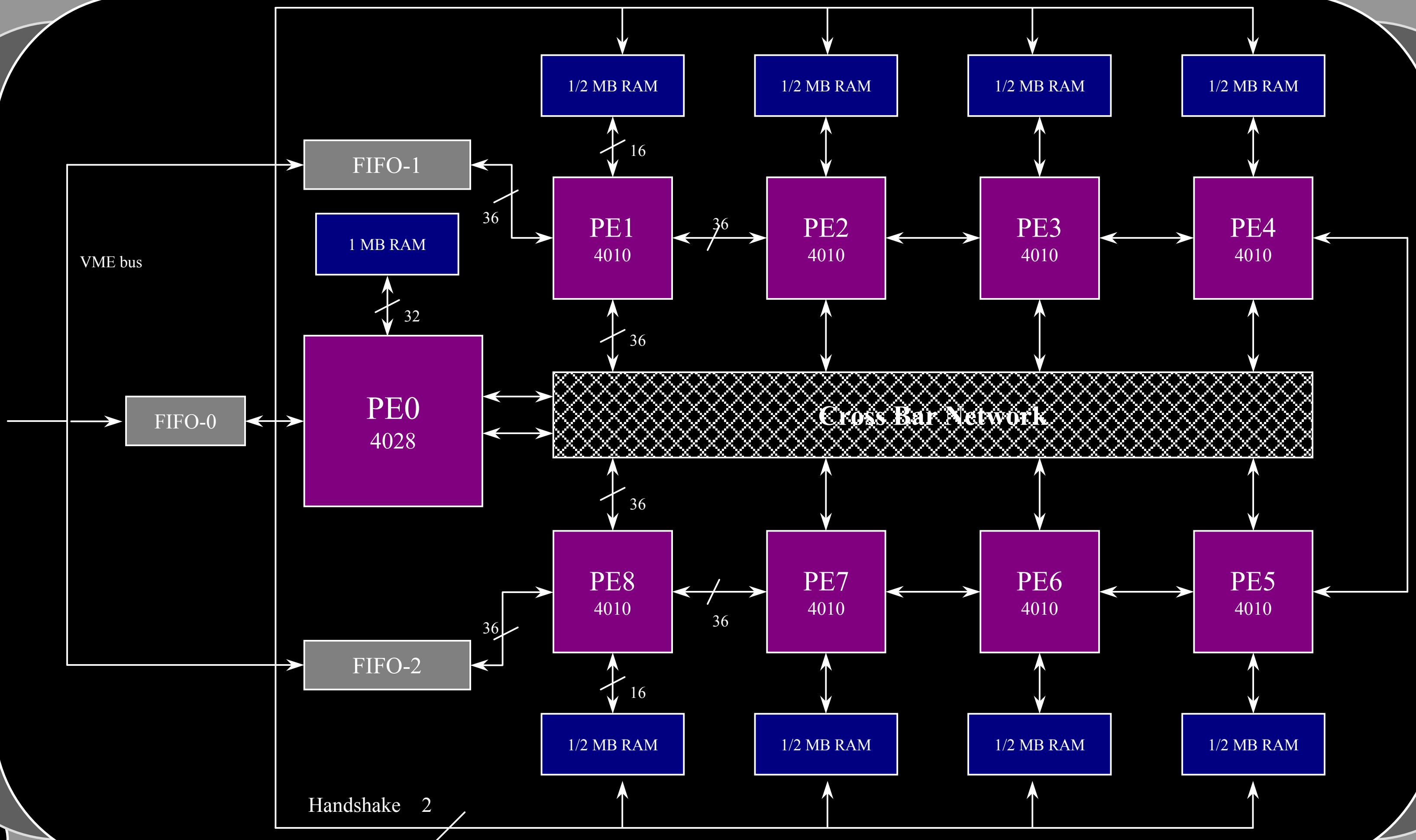
$$f(\overline{X} \mid S_k) = \frac{1}{(2\pi)^{d/2}\sigma^d} \frac{1}{P_k} \sum_{i=1}^{P_k} \exp\left[- \frac{(\overline{X} - \overline{W_{ki}})^T (\overline{X} - \overline{W_{ki}})}{2\sigma^2} \right]$$

MATLAB Vectorized

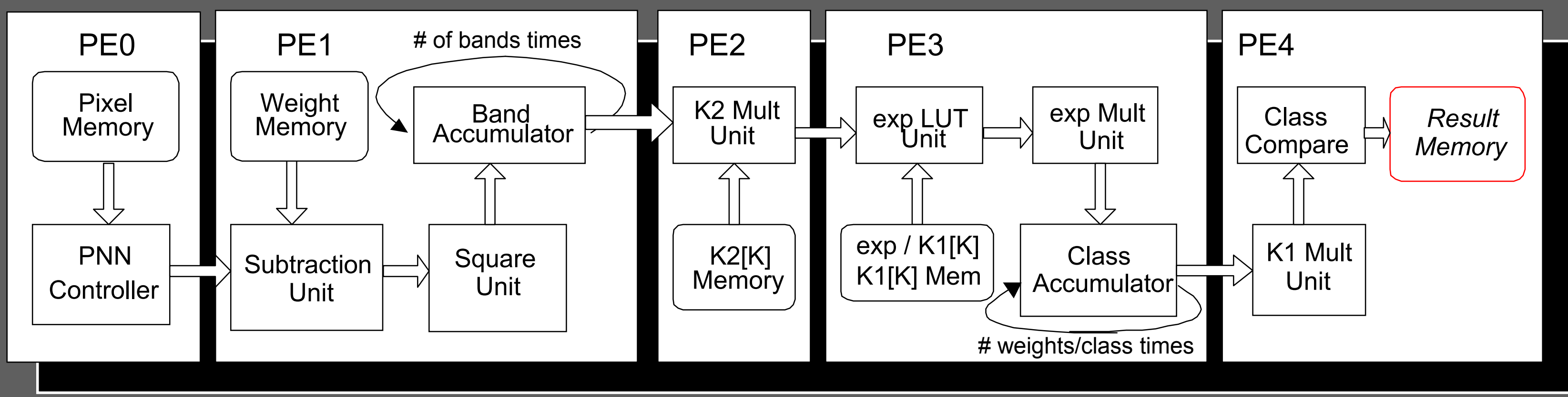
```
% reshape data
weights = reshape(class',bands,pattern_size(1),classes);

for p=1:rows*cols
   % load pixel to process
   pixel = data( (p-1)*bands+1:p*bands);

   % reshape pixel
   pixels = reshape(pixel(:,ones(1,patterns)),
                    bands,pattern_size(1),classes);

   % do calculation
   vec_res = k1(1).*sum(exp( -(k2(1).*sum((weights-pixels).^2)) ));
   vec_ans = find(vec_res==max(vec_res))-1;
   results(p) = vec_ans;
end
```

$$f(\overline{X} \mid S_k) = \frac{1}{(2\pi)^{d/2}\sigma^d} \frac{1}{P_k} \sum_{i=1}^{P_k} \exp\left[-\frac{(\overline{X} - \overline{W_{ki}})^T (\overline{X} - \overline{W_{ki}})}{2\sigma^2} \right]$$

WildChild Architecture



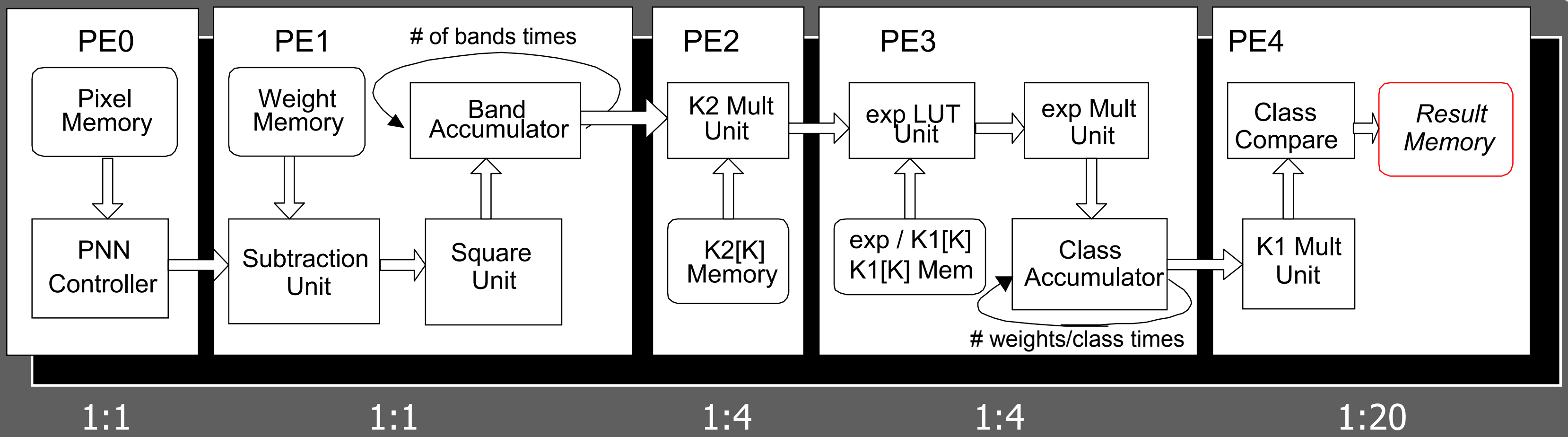
Initial FPGA Mapping



5% 67% 85% 82% 82%

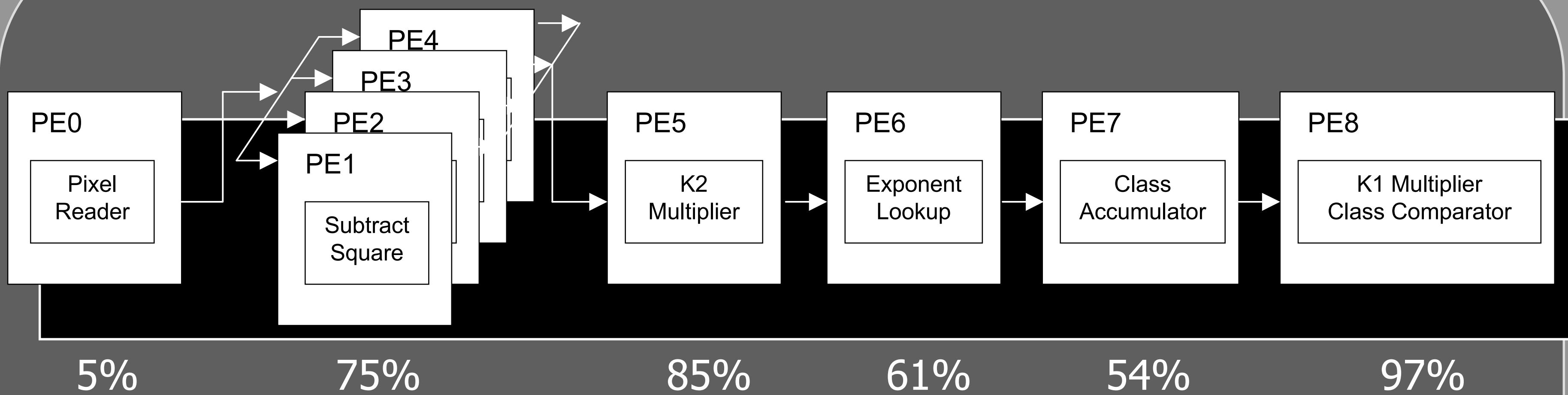
$$f(\overline{X} \mid S_k) = \frac{1}{(2\pi)^{d/2}\sigma^d} \frac{1}{P_k} \sum_{i=1}^{P_k} \exp\left[-\frac{(\overline{X} - \overline{W_{ki}})^T (\overline{X} - \overline{W_{ki}})}{2\sigma^2} \right]$$

Improving the Mapping



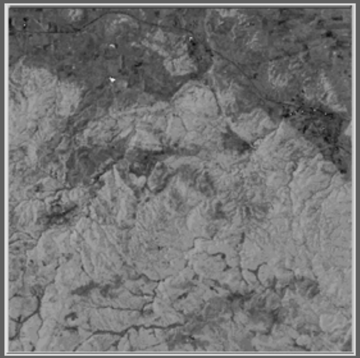
- Improve speed of PNN
  - Utilize all eight processing elements
  - Time-multiplex low-rate functions
  - Vary precision of multipliers/lookups

Optimized Mapping

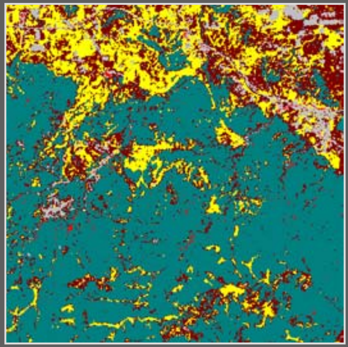


$$f(\overline{X}\,|\,S_k) = \frac{1}{(2\pi)^{d/2}\sigma^d}\frac{1}{P_k}\sum_{i=1}^{P_k}\exp\left[-\frac{(\overline{X}-\overline{W_{ki}})^T(\overline{X}-\overline{W_{ki}})}{2\sigma^2}\right]$$

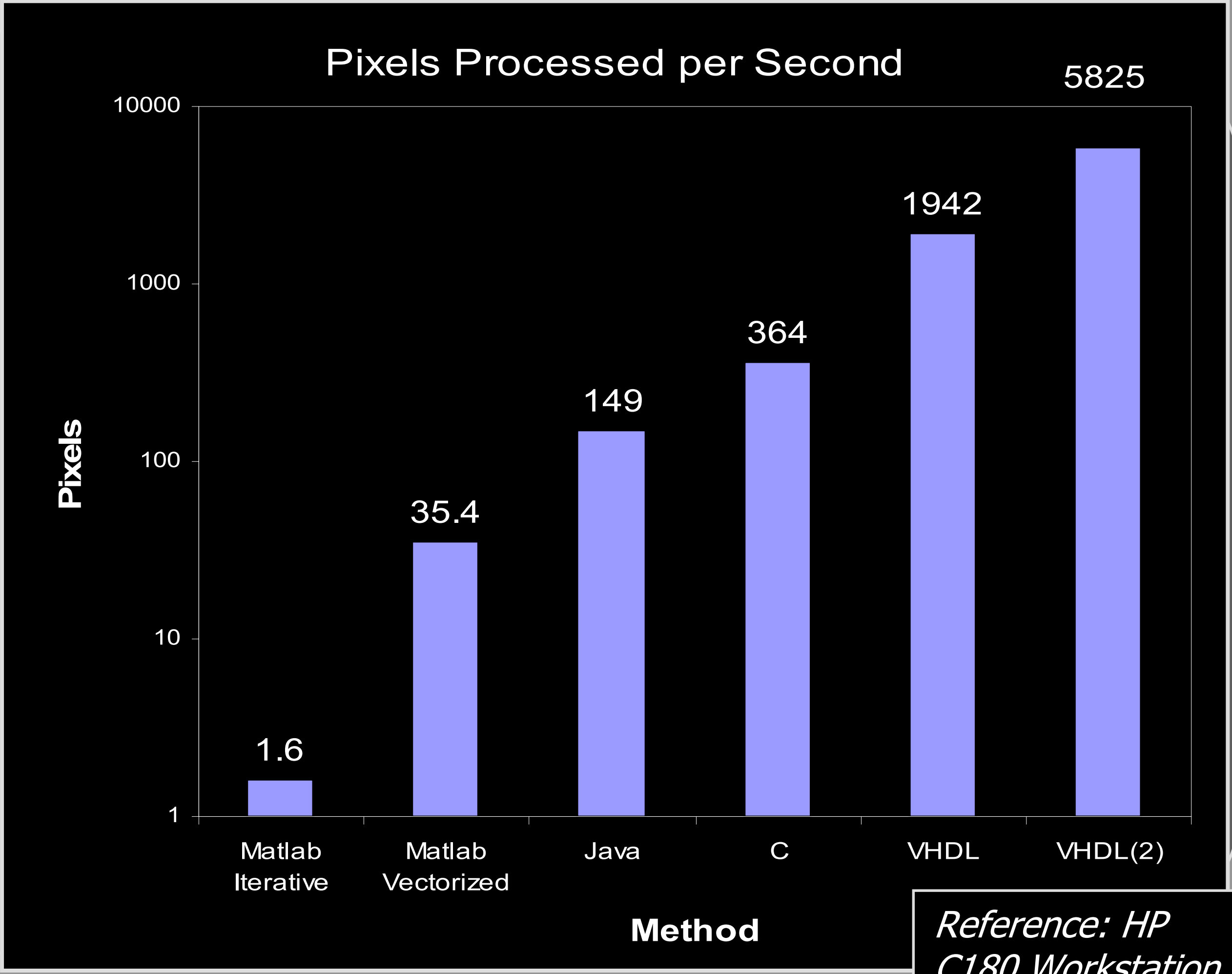
Results



Raw Image Data



Processed Image

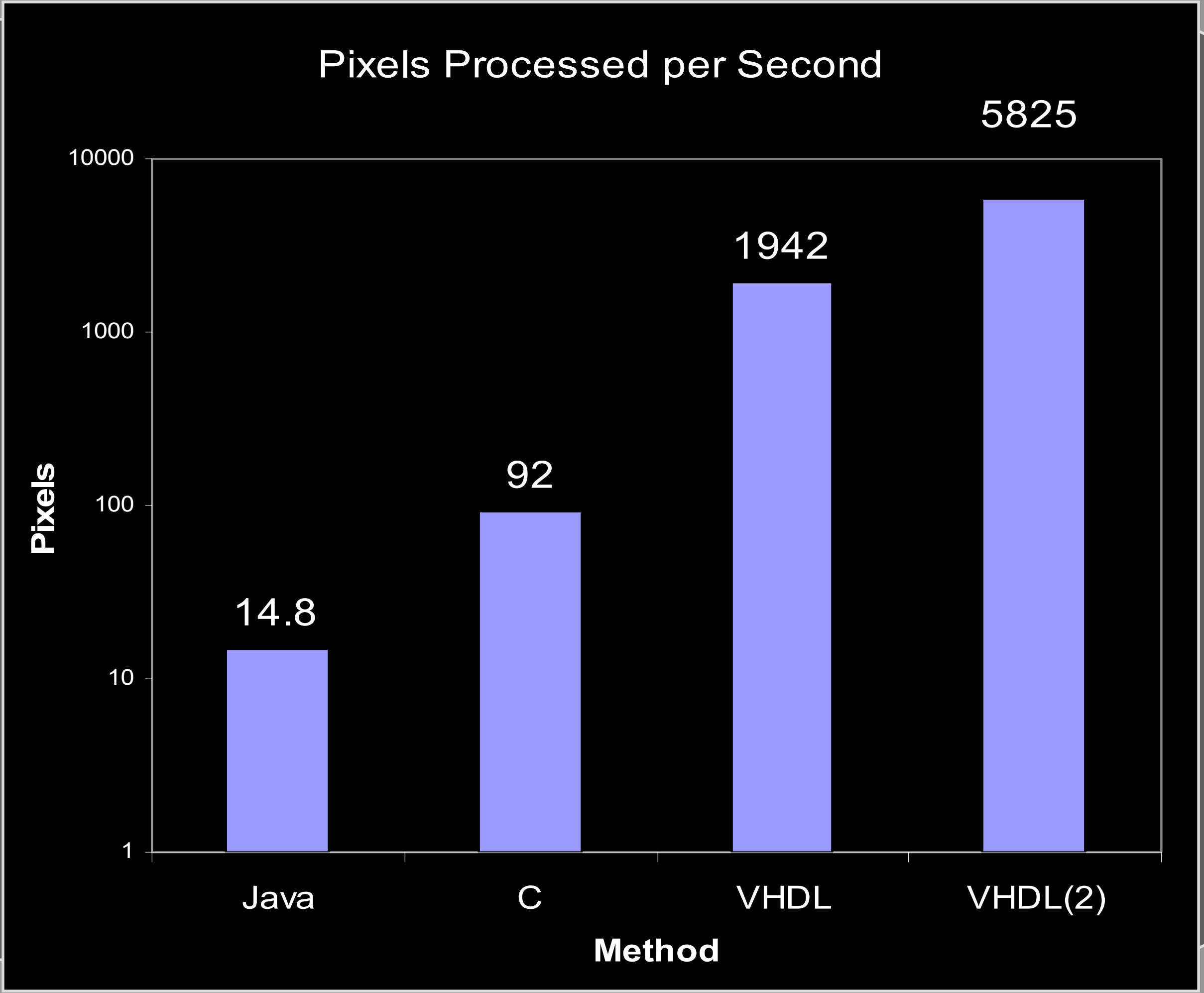


Reference: HP C180 Workstation

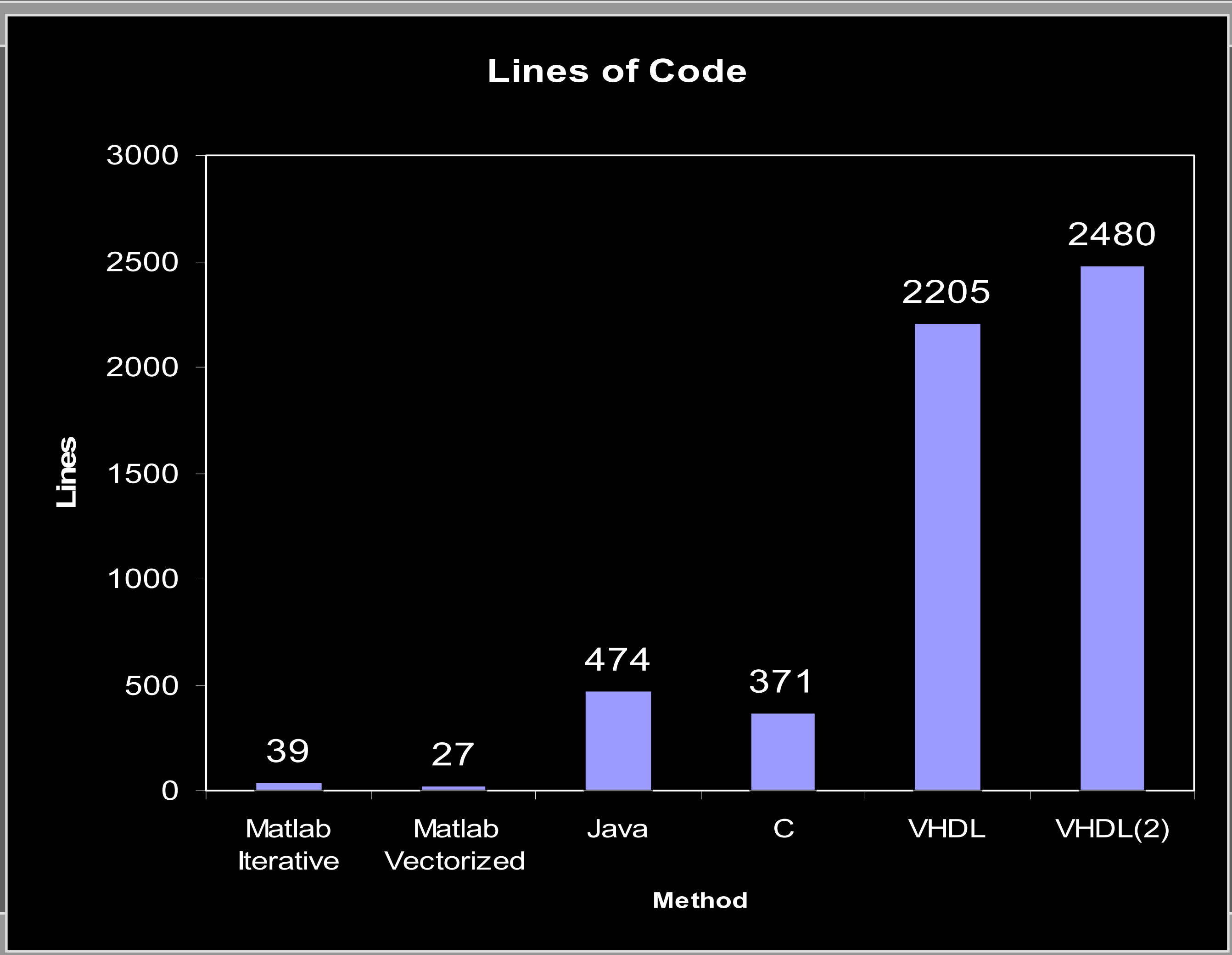
Results (Cont'd)

Reference:
MATCH Testbed

Force 5V
MicroSPARC CPU
64 MB RAM



Results (Cont'd)



Conclusions

- Adaptive computing has high performance potential
- Need better tools to take full advantage of FPGAs
- NASA will need high-performance solutions soon
- MATCH is a good solution for NASA applications
  - High processing loads and I/O requirements
  - Well-suited for acceleration using adaptive computing
  - Scientists want to write in MATLAB rather than C+VHDL
- Driver application research will lead to a better, smarter MATCH compiler