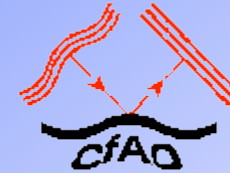




Center for Adaptive Optics
An NSF Science & Technology Center



FPGA's, What Are They and How Can We Use Them in AO

Marc Reinig

UCO/Lick Observatory Laboratory for Adaptive Optics
University of California, Santa Cruz

CfAO Summer School on Adaptive Optics
August 12, 2005



Laboratory for Adaptive Optics

*UCO/Lick Observatory
University of California, Santa Cruz*

“The key to efficiency is to take
advantage of special structures”

Kurt Vogel

Goals

- Expose you to a technology that can be used to solve many of the problems you may need to solve: both computational and general data handling
- Describe an real example of where we are applying it to one of the toughest AO problems, tomography

FPGA's

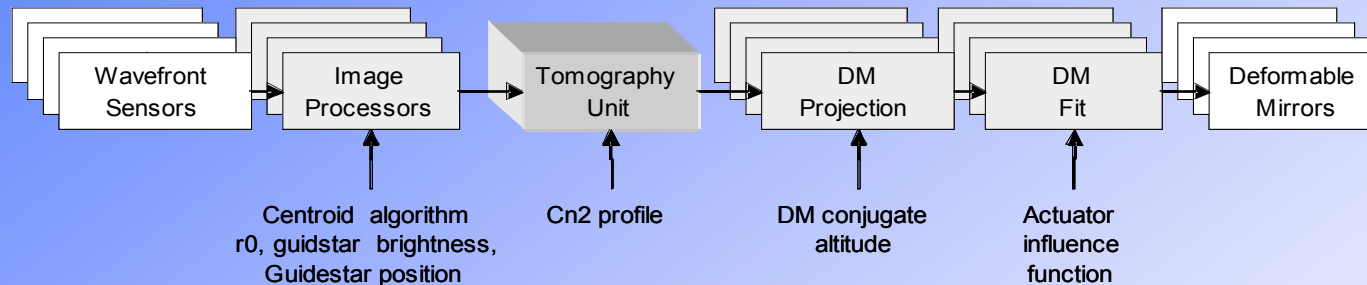
- Field Programmable Gate Arrays -

- Field Programmable – you can change the logic in the chip in the field by loading a file just as you would update code on a regular processor
- Gate Array – An array of logic gates (can be >100,000) that can be hooked up to build virtually any logic function you need

“The key to efficiency is to take advantage of special structures”

- Since the FPGA has a structure, if we can match the structure of our problem we can use it most efficiently
- The whole issue is to get your design to fit most efficiently. The better the structure of your design fits the structure and resources of the FPGA, the more efficient your solution will be.

Massively parallel processing



- **Advantages**

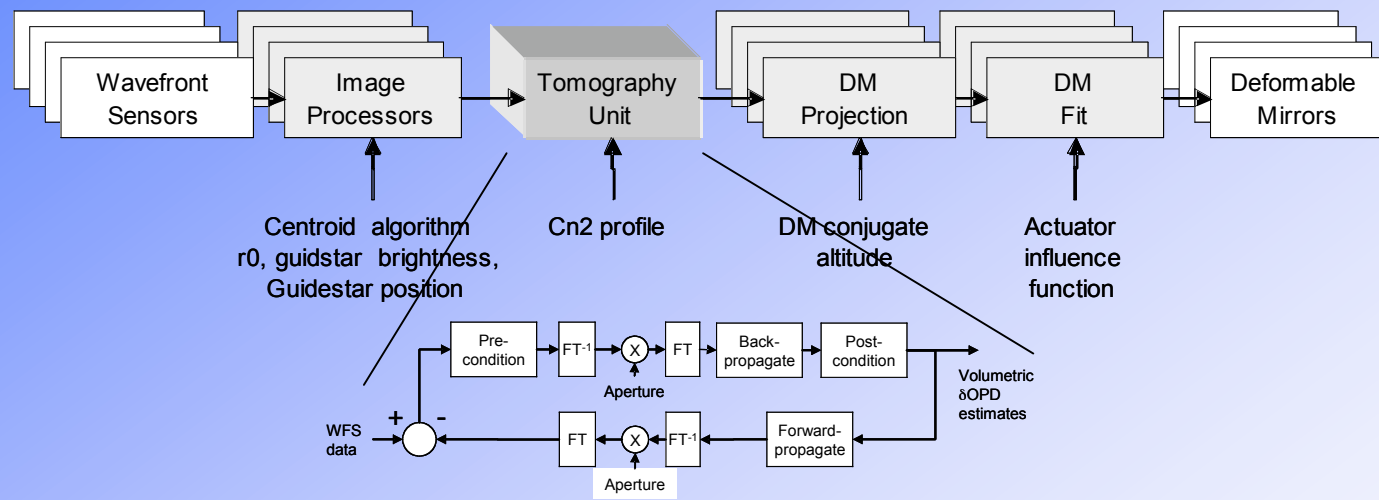
- **Many small processors** each do a small part of the task – not taxing to any one processor
- Modularity: each processor has a stand-alone task – possibly **specialized** to one piece of hardware (WFS or DM)
- Modularity makes the system **easier to diagnose** – each part has a “recognizable” task
- Modularity makes **system design easier** – each subsection depends only on parameters associated with it, as opposed to global optimization of a monolithic design

- **Requires**

- Lots of small processors, with high speed data paths
- **Iteration to solution** – but what if 1 iteration took only 1 μ s? – then we would have time for 1000 iterations per 1 ms data frame cycle!

Tomography Implementation Summary

- The architecture: **massive parallel computation**



- Conceptually **simple**
- Tested with a commercial FPGA; evaluated with simulations – *it's feasible with today's technology*
- Under study:
 - FD-PCG – extra computation per iteration traded off against faster convergence rate

Summary of the Size TMT Tomography Problem

- Tomography for a TMT sized problem
 - 128 Sub apertures (16K sub aps. per layer)
 - 8 Layers of atmosphere to estimate
 - 7 Guide stars
- >1G Floating point operations to converge to a new estimate of the atmosphere
 - We need >1KHz rate of convergence
 - => We need >1,000 G Flops per second
- **That's a huge problem to solve**

How Could We Solve This Problem?

- The problem we are solving is $y = Ax$
- One approach would be to find A^{-1} and obtain $x = A^{-1}y$
- This would take numerous Cray computers i.e. $>\$1M$; $>1,000 \text{ ft}^2$; 1 full-time operator 24/7; $>30Kw$; ...
- Reliability issues

Using Our Knowledge of the Structure of the Problems and Solutions

- Kurt and Don have pointed out the ways of using structure to improve the accuracy of our results and reduce the computational requirements, making the task more tractable.
- We can solve the problem iteratively (**speed**)
- Using our knowledge of the structure of the solution desired (Kolmogorov power spectrum) we can **post-process for a more accurate result**.
- Using our knowledge of the type of Eigen Values we need for rapid convergence (low Eigen number and/or clustered Eigen Values) we can **pick an appropriate pre-conditioning matrix and reduce the number of iterations needed for our solution to converge**.

Tomography solutions

- Least squares

$$\mathbf{x} = \mathbf{A}^T (\mathbf{A}\mathbf{A}^T)^{-1} \mathbf{y}$$

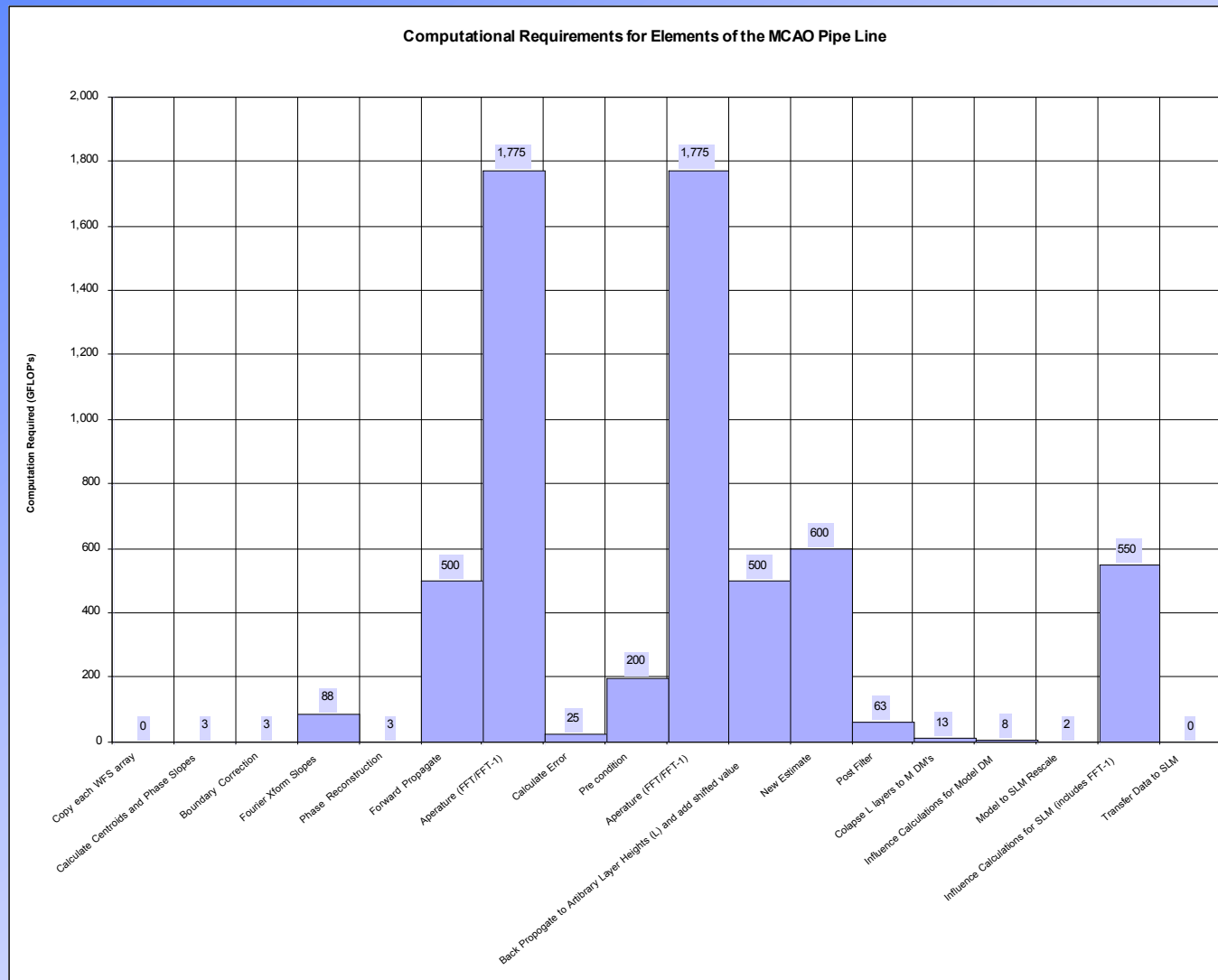
- Minimum variance

$$\mathbf{x} = \mathbf{P}\mathbf{A}^T (\mathbf{A}\mathbf{P}\mathbf{A}^T)^{-1} \mathbf{y}$$

- Minimum variance with measurement noise

$$\mathbf{x} = \mathbf{P}\mathbf{A}^T (\mathbf{A}\mathbf{P}\mathbf{A}^T + \mathbf{N})^{-1} \mathbf{y}$$

Where Do The CPU Cycles Go?



A Few GFlops Here, a Few There, Pretty Soon You're Talking Some Serious Compute Power

- The previous has reduced our computational requirements to something over 1,000 GFlops
- Continuing along in the same vane, now that we have an optimized algorithm, how can we use our knowledge of the structure of the algorithm, the problem or the solutions to gain more efficiency?
Because God knows we need it!

How Do We Solve The Compute Problem?

- At one end of the spectrum we could have a single powerful computer go through each ray sequentially, forward propagating, comparing to measured, back propagating the error, computing new estimated values based on the back propagation, and continuing until the overall error is reduce to our desired level.
- **Unfortunately, such a single computer doesn't exist**, or at least is not available to us mortals.

Traditional Parallel Processing Approaches

- Since the problem is highly parallelizable, we could make a cluster of powerful computers (Beowolf, etc.)
- Numerous (>20) powerful computers i.e. >\$1M; >1,000 ft²; 1 full-time operator 24/7; >30Kw; ...
- **This is an attainable solution, but an extremely distasteful one** to put on an instrument in the dome
- So, there we have one end of the spectrum.

The other End of the Spectrum

- What if we had one processor per voxel (~100,000 of them)?
- They could be very simple
 - Tailored specifically for this type of problem
 - With a minimum set of operations
 - Minimum set of registers
- But, **100,000 of them? The mind boggles!**

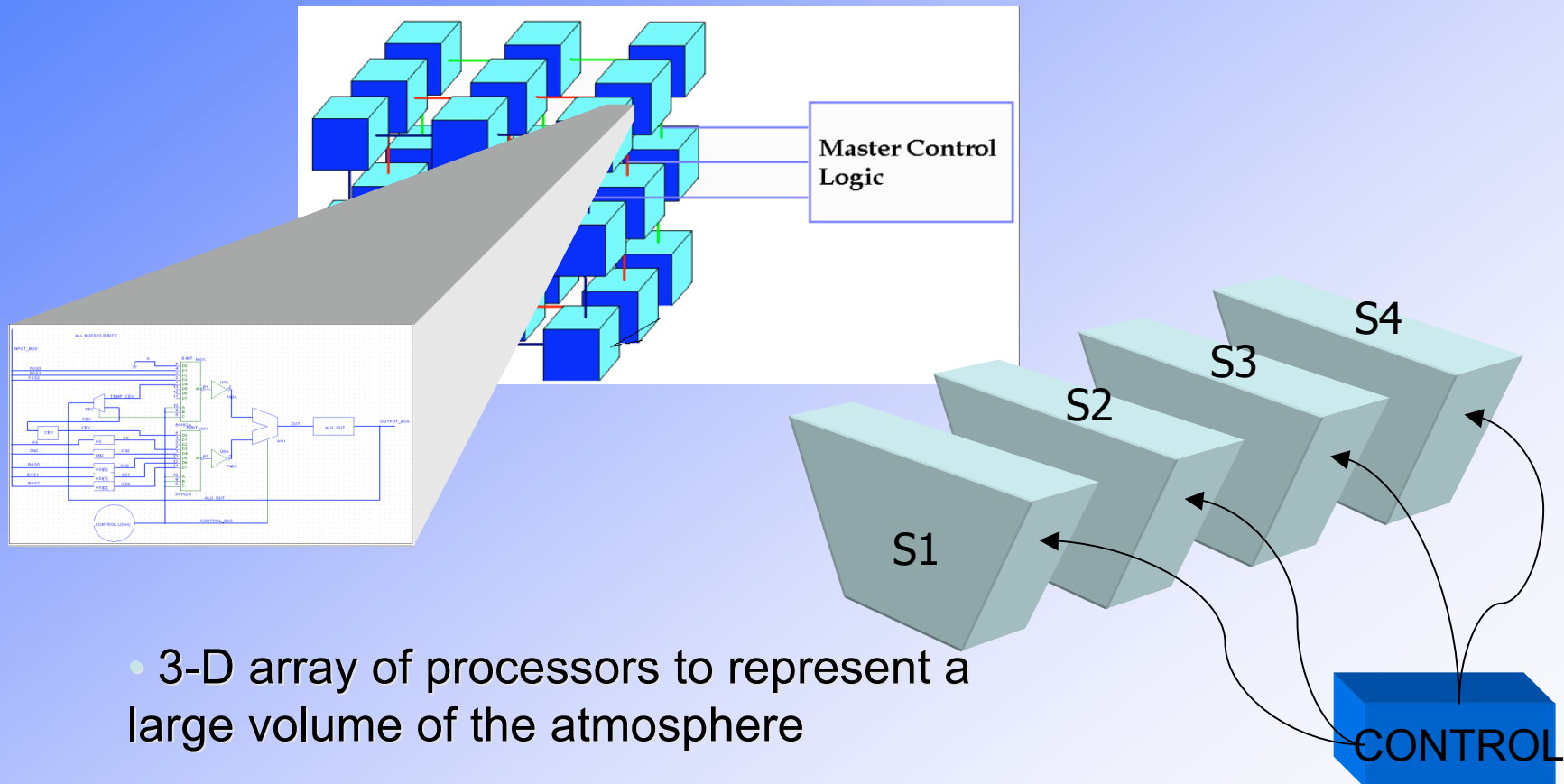
Sometimes It is Good To Examine the Ridiculous to See How Ridiculous it Really Is

- What if we could put 500 of these things on a chip, all properly interconnected
- We would only need 200 chips and our solution would look like:
 - ~1, 000W (5W/chip); 10 ft³; <\$250,000; no operators
- OK, Great! Let's do it!
- Unfortunately, **there are no chips like this that meet our needs.**

OK. I Must Be Going Somewhere With This, Right?

- CPU vendors use state of the art semiconductor processes to make CPU's with 100's of Millions of transistors for their complex processors
- Other companies use similar processes to give us the ability to put 500 very, very simple specialized processors on a single chip.
- They also give us the tools to do it ~easily
- One of the key things that make this possible is not just the amount of raw logic and tools to which they give us access, but the fact that the structure of these chips is a natural fit to the way that our problem is structured!!

Matching Problem Structure to Solution Structure



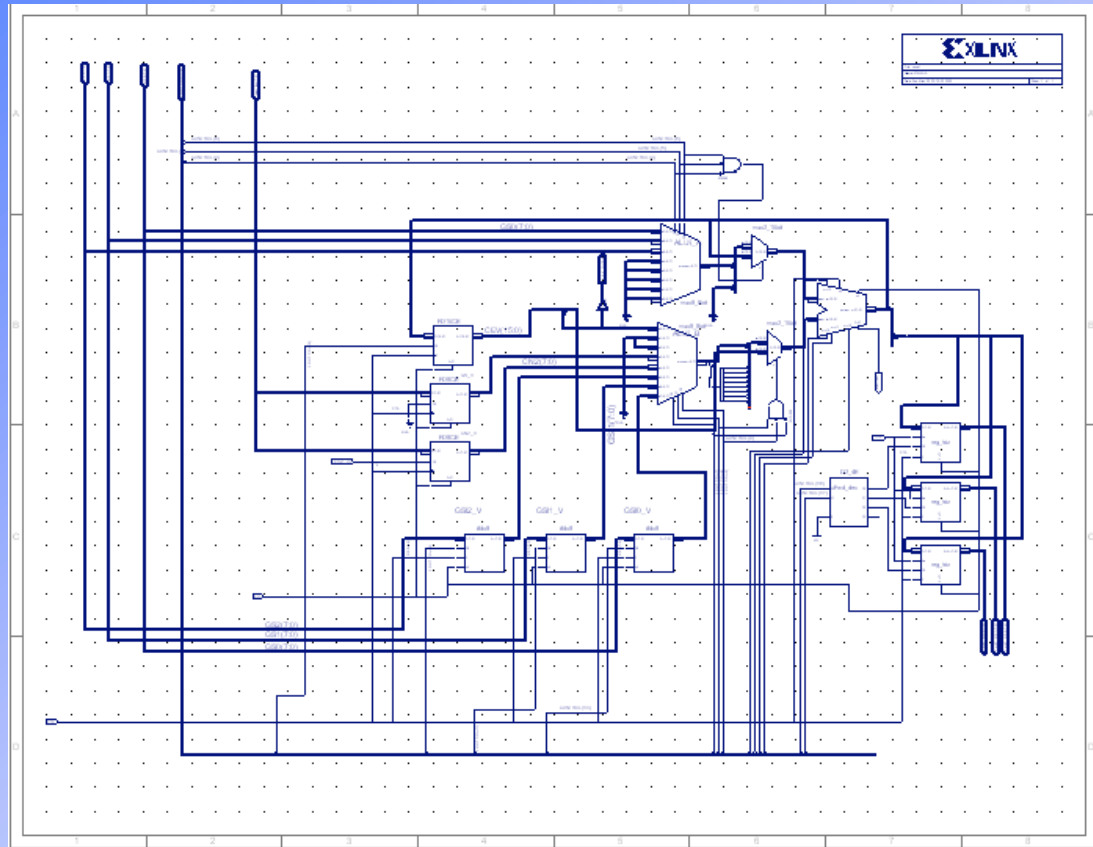
- 3-D array of processors to represent a large volume of the atmosphere
- Each processor represents one voxel of the represented atmosphere

Our ALU Design

- Code is missing in action, but will be included in the online version

```
Case: Operation
{
    (Mult)      out = In1 * In2
    (Add)       out = In1 + In2
}
```

Our Voxel Processor



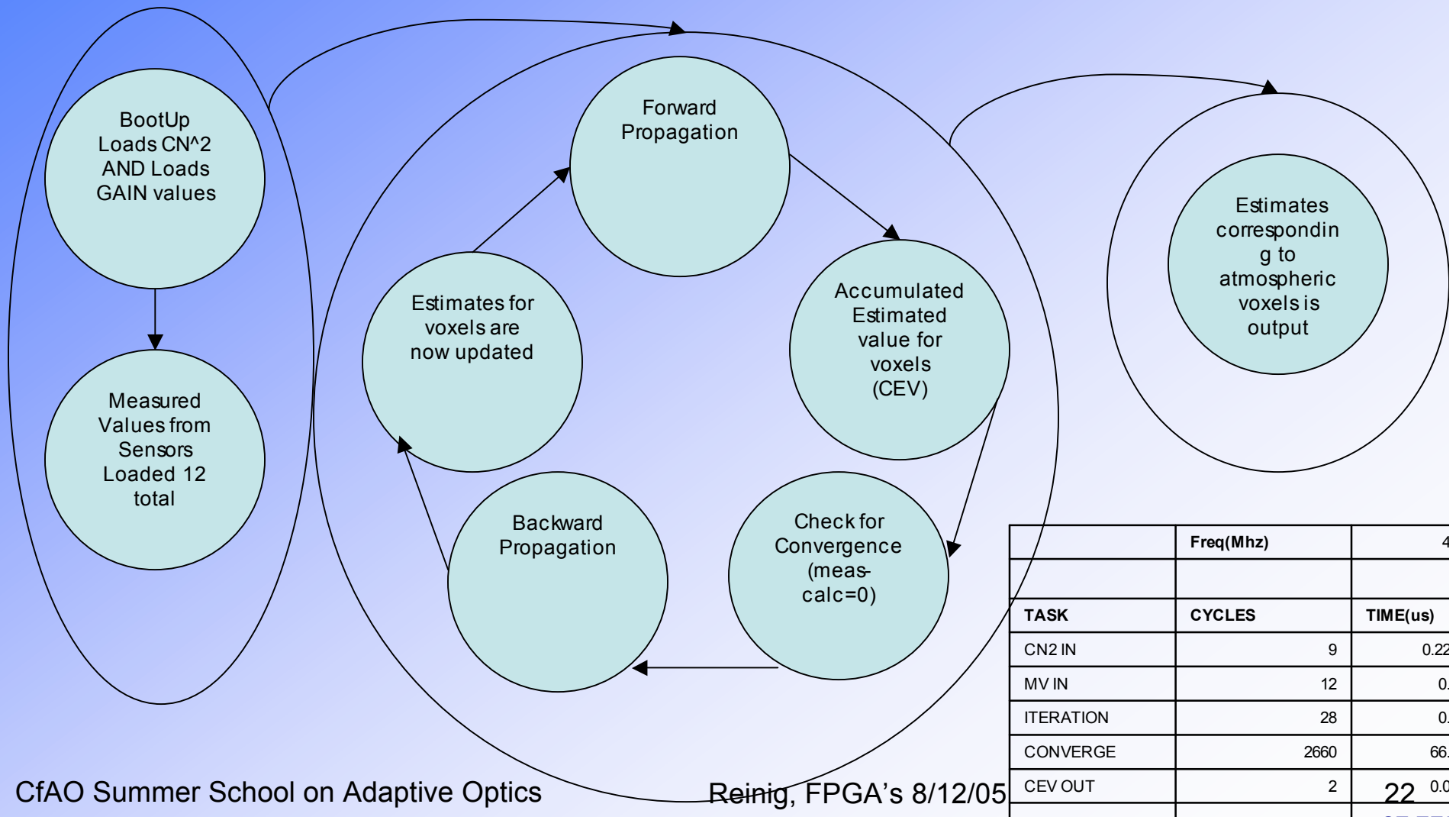
- Control

- Implement multiplication and division of integer values

- ALU

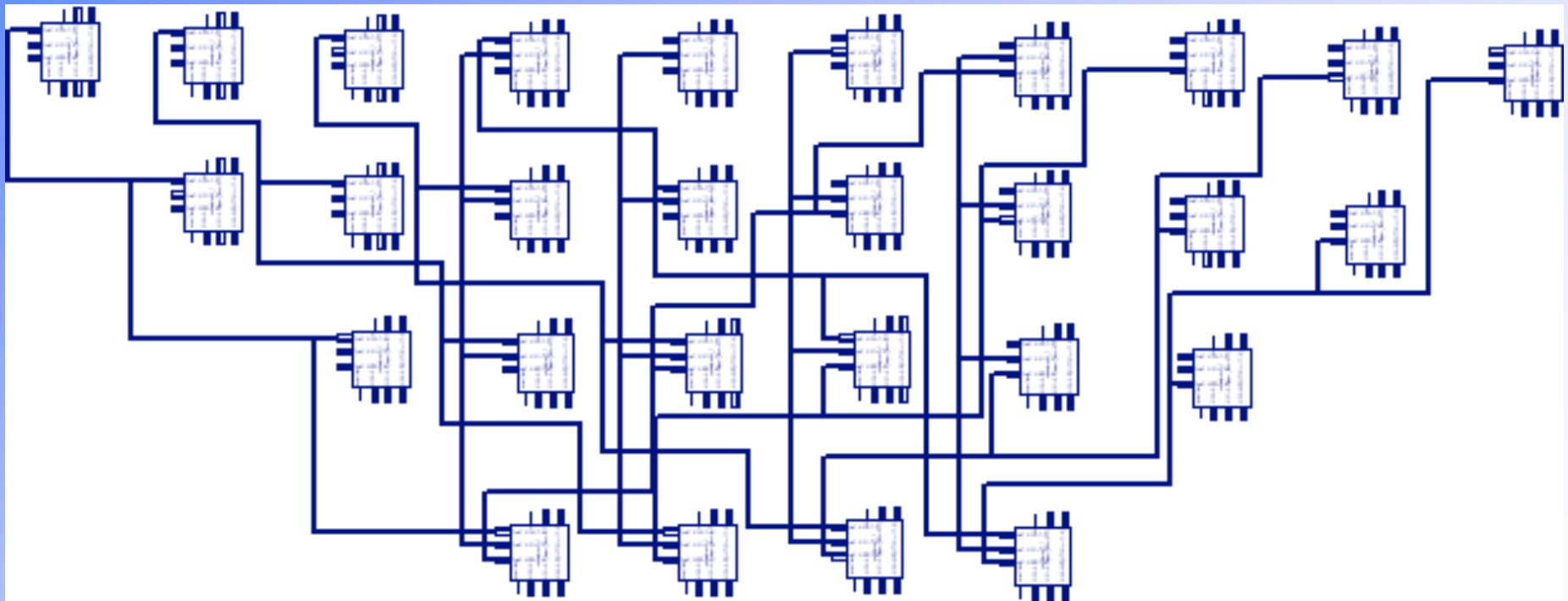
- Capable of +, -, *, <<
- Outputs OF, Z

Our Control Logic

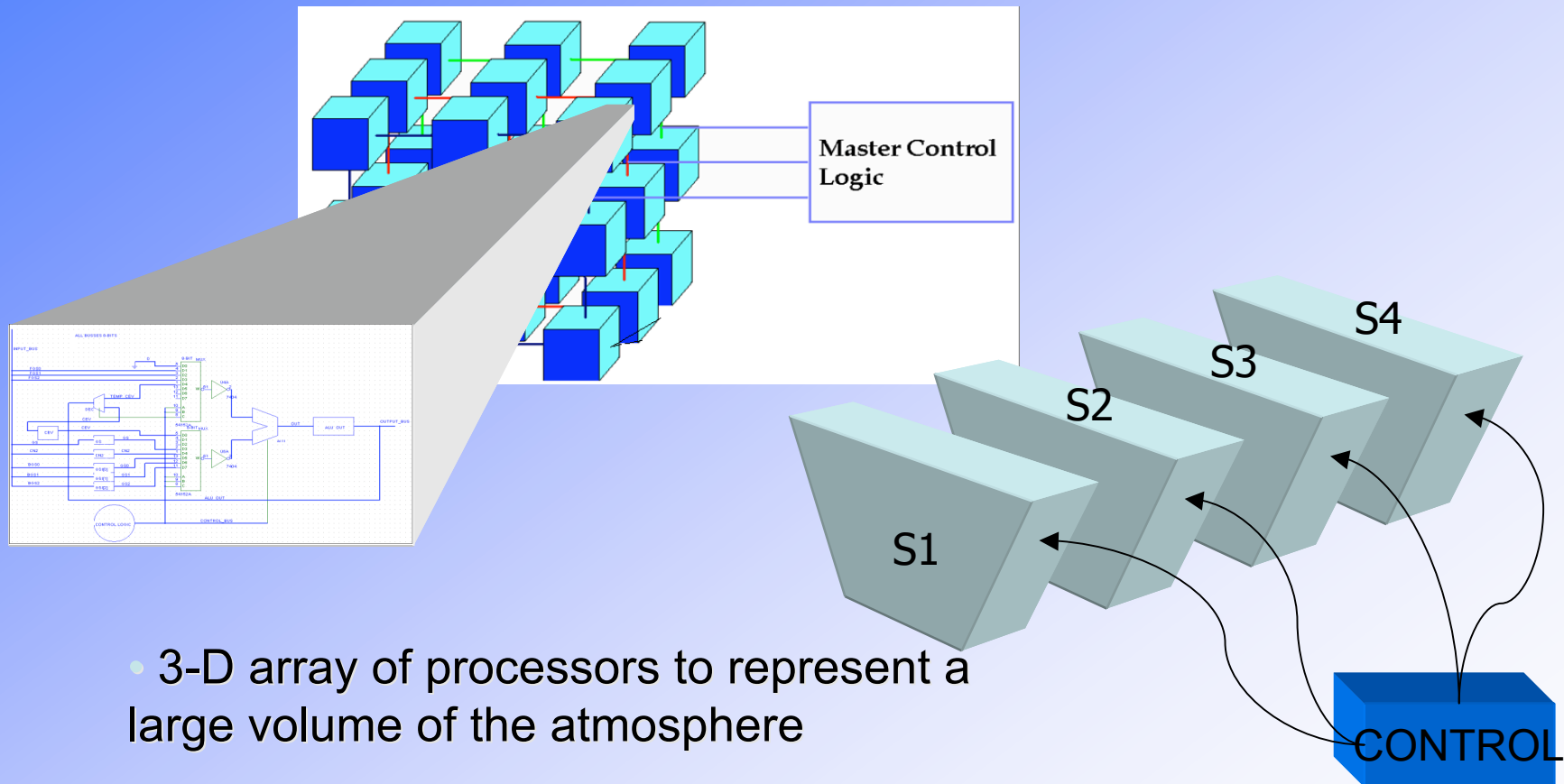


	Freq(MHz)	
		4
TASK	CYCLES	TIME(us)
CN2 IN	9	0.22
MV IN	12	0.
ITERATION	28	0.
CONVERGE	2660	66.
CEV OUT	2	22 0.0

A Partial Slice of the Atmosphere in the Tomography Engine Using Multiple Voxel Processors



Matching Problem Structure to Solution Structure

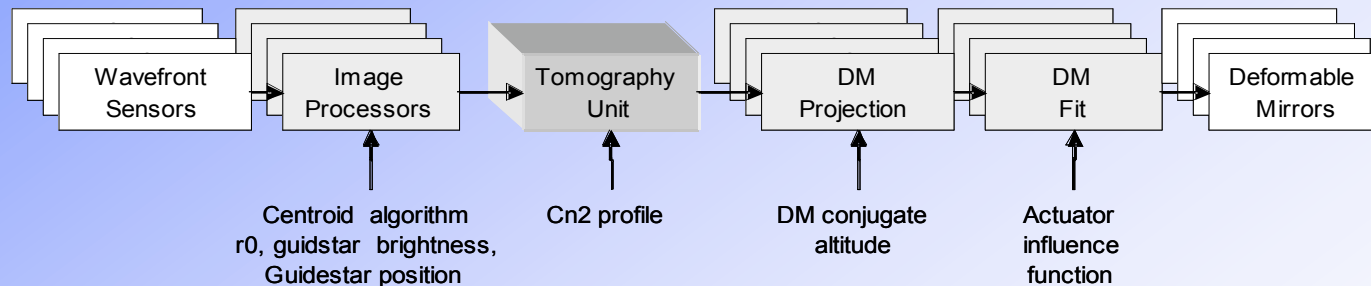


- 3-D array of processors to represent a large volume of the atmosphere
- Each processor represents one voxel of the represented atmosphere

Guess what!

It Actually Works for All the Blocks In the Pipe Line, Not just the Tomography Portion!

- Just Add More State Machines that Describe the New Algorithms -



- The Tomography Engine is Really a General Purpose Iterative Solver -

Barriers to Using FPGA's

- Fear of the Unknown -

- “But I’m not a logic designer, I’m barely a programmer and I wouldn’t know how to start designing a processor!”
- Right! For the final design of an instrument you would bring in a professional. But many of us, though not professionals, will do basic optical designs and setups, or do some hacking at code to accomplish things and prove concepts, or do basic hardware hacking.
- This is really no different. Well it is but you get the idea.
- You can do this to prove concepts or examine possibilities. It is not a black art, nor does it take Giga Bucks.

FPGA Conclusions

These opinions are my own and do not reflect the Opinions of the University of California or its Regents and does not bind the University etc. etc. ...

- TMT sized problem in less than 10 medium sized boards and <<\$1M
>7 GS, >10 layers, >100x100 sub aps
- Keck sized problem in a few boards
5 GS, 10 layers, >20x20 sub aps
- Lick sized problem in 1 board
3 GS, 10 layers, 10x10 sub aps

FPGA Conclusions (Cont.)

- Yes, it's something new.
- No, it's not trivial
- Yes, it's within all of your grasp
- The goal here is not to convert you to using FPGA's all over the place or in your next design, but to let you know they do exist and they could make a significant impact in some of your projects
- You should consider them as part of your tool kit for solving your problems the way you would consider a PC, a single board computer, a servo controller, a code library, etc.
- Current FPGA vendors, Xilinx and Altera

Thank You!