Parallel EDA,

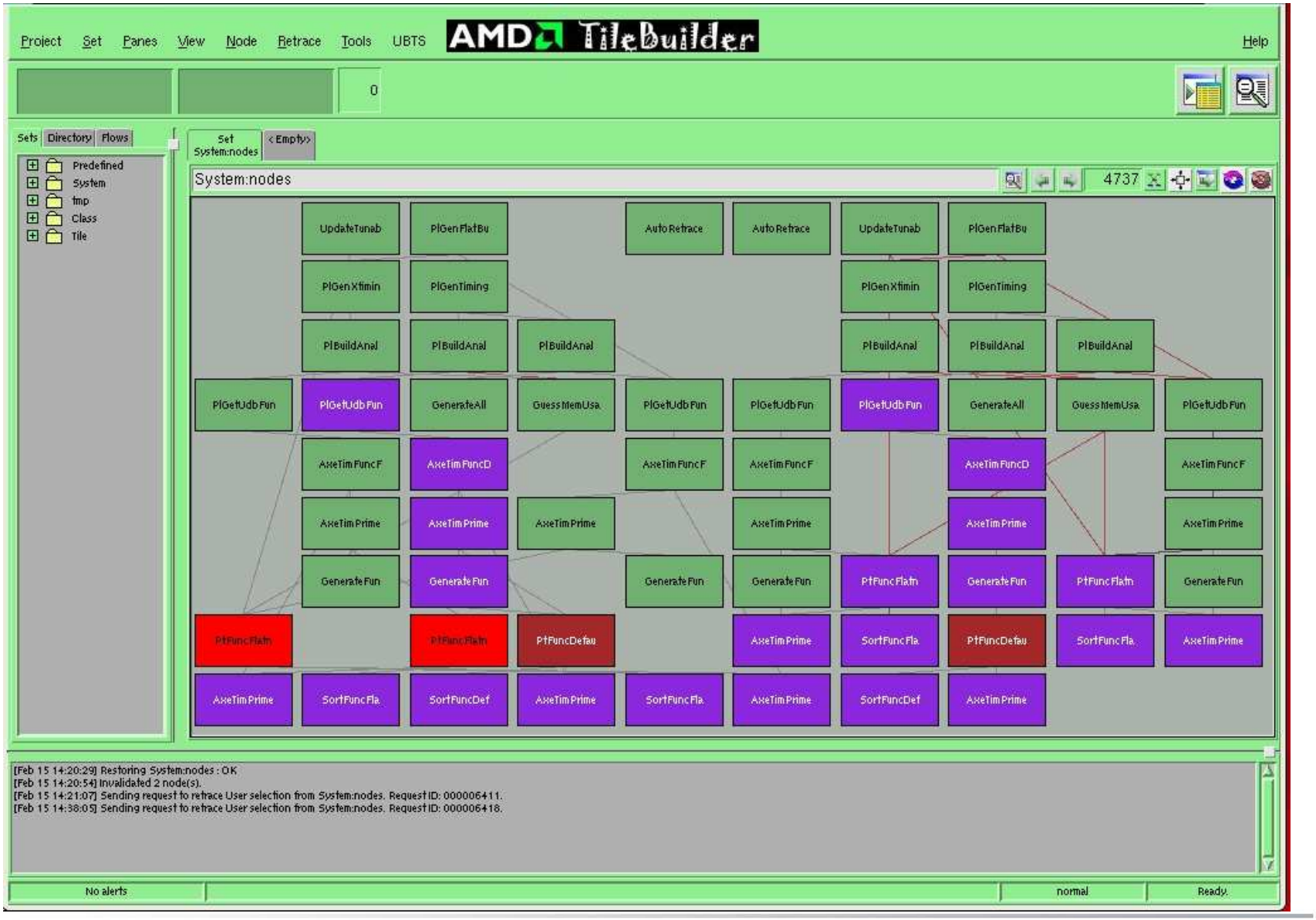
A user's perspective

Tom Spyrou
AMD Fellow
April 2012, EDPS

User's View of Parallel EDA

- Tool(s) that run for a long time
- Access to a farm of machines usually managed by lsf or something similar
- Fair share concept allocating machines across users
  - Exclusive access to machines is discouraged
  - Access to large memory machines requires long waits
- Hoping to use fair share of the farm to get the job done faster
- Cloud EDA is not yet in the mindset

AMD
The future is fusion



AMD
The future is fusion

Given the User's view these are the trade-offs

- Requiring multiple machines simultaneously does not work well.
- For example in Timing Analysis, MMMC distributed, where all timing views are needed at the same time is not feasible.
- Machines need to be used effectively as they come up, no waiting for all machines requested before starting
- If multiple machines are used and there is high memory duplication this is not efficient
- Multi-threading with minimal memory increase per thread is preferred but there is a usability issue

AMD
The future is fusion

Usability of multi-threading and LSF/others

- Tool asks user to set number of threads
- User must make sure bsub call asks for that number of threads.
- If there is a good machine with n-1 processors available it won't be chosen
- If resource managers and tool developers could coordinate it would be helpful.
  - Ask for machine with X memory and most available cpus from LSF. Tell App how many it can use via envi variable

AMD
The future is fusion

Fair Share example

- A user cannot have more than 8 machines with running jobs at the same time
- 8-way machines are the most available
- 64 cpus
- Easy to get machines with 64Gig RAM
- Hard but possible to get machines with 128Gig Ram
- Bigger requires special approval / process

AMD
The future is fusion

Using parallel processing to reduce Machine Size

- Split job into pieces where each uses less memory
- MMMC Timing analysis
- Routing
- DRC

AMD
The future is fusion

What about the Cloud

- Possible benefits
  - Peak access to machines when needed
  - **Solve the application / resource manager coordination**
  - Shared cache contention
  - **EDA Vendor easy access to data**
    - **Quick debugging of actual issue, like internal CAD**
- Questions
  - Data security real and perceived
- Cost predictability
  - Time based versus usage based model

AMD
The future is fusion

Methodology for deciding if and how much parallel programming to use

- When answering this question we need to look at :
  - Technical issues
  - ROI issues for the expert resources usually needed to write parallel programs and make them scale

- List of questions that form a decision diagram starting at the simplest solution moving to the most complex

- I may not have the decision points right for everyone but I feel strongly about the general methodology of trying to start simple and adding complexity when there is ROI.

AMD
The future is fusion

Methodology for deciding if and how much parallel programming to use

- Can non-shared memory, coarse grained with separate processes give the needed scalability?
  - Use only processes and keep it simple when possible
  - Coarseness defined as compute time >> data transfer time
  - My Blog Post discusses pushing this to the limit :
    - http://software.intel.com/en-us/blogs/2009/09/02/parallelizing-legacy-code-using-fine-grained-distributed-processing/

AMD
The future is fusion

Methodology for deciding if and how much parallel programming to use

- If shared memory is required does the task tend to share a lot of memory for read and then generate a smaller amount of data?
  - Use copy on write fork() and keep it simple when possible
  - Generate all data before fork(). Compute generates new data versus updating existing data.
  - My blog post discusses this in detail :
    - http://software.intel.com/en-us/blogs/2009/09/25/parallelizing-legacy-unixlinux-code-using-copy-on-write-fork/

AMD
The future is fusion

Methodology for deciding if and how much parallel programming to use

- If shared memory is required for both reading and writing at a fine grained level then we need threads that share memory.
  - Can you get the needed scalability with X86 threads including using SSE?
  - Stay with X86 threads, if SSE use openCL if not use the pthread library since more people are trained on it.

AMD
The future is fusion

Methodology for deciding if and how much parallel programming to use

- Is the cost of the run in terms of number of cpu's needed too high, or is the bottleneck access to cpu's instead of scalability of the algorithm?
  - Use openCL to access GPU hardware and see if the compute power there can be utilized. Save on X86 hosts.
  - If X86 based threads won't scale, there is either a bottleneck in the divisibility of the work or the data transfer.
    - If this is the case it probably won't scale on GPU's either

AMD
The future is fusion

Trademark Attribution

AMD, the AMD Arrow logo and combinations thereof are trademarks of Advanced Micro Devices, Inc. in the United States and/or other jurisdictions. Other names used in this presentation are for identification purposes only and may be trademarks of their respective owners.

©2009 Advanced Micro Devices, Inc. All rights reserved.

AMD
The future is fusion