

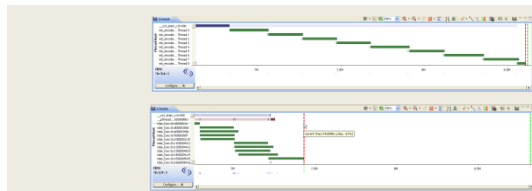
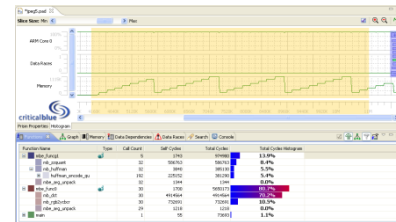


# Software Analysis Tool For Multicore

*Analyze existing sequential single core software*  
*Explore benefits of migration to parallel implementation*  
*Tune and verify safety of multithreaded code*

Wondering what benefits would be delivered if you multithreaded your existing sequential software and ran it on multicore platforms? You are not alone. Software teams resist parallelizing their applications because they can't risk potentially complex and uncertain development cycles without real assurance that they can achieve high quality code, on time, with predicted power and performance improvements. By working closely with multithread/multicore engineers, CriticalBlue developed Prism™ to take developers from 'what-if' to 'requirements met', streamlining sequential to parallel programming in five best practice steps:

**1. Analyze** – characterize a serial or parallel application on real workloads. Identify program hotspots, call trees, and data dependencies which will shape achievable parallelism.

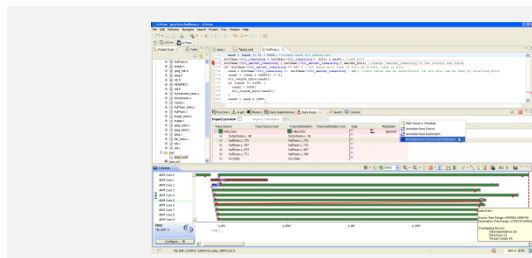


**2. Explore** – before modifying any code, explore different parallel scenarios. What if this function ran in a separate thread? What if these dependencies were removed? Understand the benefits and select the best strategy.

**3. Code** – Implement the chosen parallelization strategy. Use existing development tools without change.

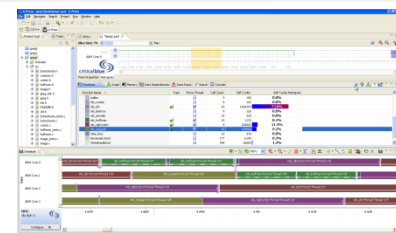
```

// Example C code snippet showing parallelization
#include <pthread.h>
void *thread_func(void *arg) {
    // Parallel execution logic
}
int main() {
    pthread_t t1, t2;
    pthread_create(&t1, NULL, thread_func, NULL);
    pthread_create(&t2, NULL, thread_func, NULL);
    pthread_join(t1, NULL);
    pthread_join(t2, NULL);
}
    
```



**4. Verify** – Confirm that the implementation safely achieved the desired results. Functions properly threaded and dependencies removed? Any potential data races? Synchronization bottlenecks?

**5. Tune** – Analyze opportunities for further parallelization by reapplying the previous steps.



## Benefits

**Gain more insight, waste less time** – analyze multiple parallelization scenarios before making any code changes. Verify parallel objectives are properly achieved.

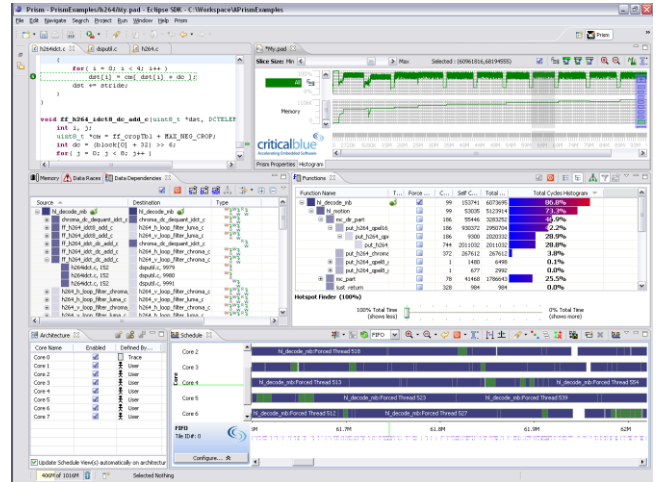
**Augment the way you already work** – there are no new languages or proprietary extensions to learn. Start with existing serial or parallel code – mix standard C, C++, assembly, or even binary objects. Work within your own code base. Use familiar development tools and libraries. Prism integrates with Eclipse for maximum portability.

**Visualize schedules and pinpoint dependency relationships** – parallel programming is about relationships. Understand synchronization between threads, and go directly to the source to see dependency relationships which must be properly maintained.

**Design for scalability and reuse** – develop parallelization which scales across varying number of cores. Eliminate potential race conditions which could fail as data sources and processing resources change.

**Apply precision when needed** – platform support ranges from instruction-accurate application analysis to highly specialized and system-accurate chip and board support packages.

**Boost your parallel IQ** – Prism includes integrated documentation and practical case studies. Supplemental services are available to instill practical parallel practices or augment development teams.



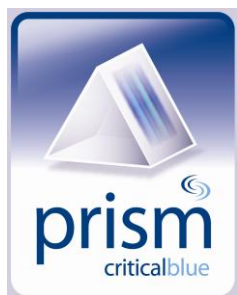
## Platform Support Packages

Prism is engineered to work with different processors, programming models, and system environments. Prism capabilities are enabled through platform support packages (PSPs). Instruction PSPs provide instruction-accurate analysis for specific processor families and are intended for quick application parallelization. Core PSPs improve the accuracy of results by adding core awareness such as data cache coherency and miss impact analysis, hardware multithreading, branch misprediction and pipeline interlock analysis. Most mainstream embedded processors are supported at one of these levels. System PSPs enhance Prism further for production processors, chips, and boards. Platform analysis extensions might include configurable full system modeling, power analysis, custom scheduling, specialized parallel programming models, as well as operating system and board level integration.

## Pricing

Prism pricing starts at a few hundred dollars per month for a minimum one year license with one Instruction PSP or one Core PSP. Additional open source or commercial simulators, or development systems may be required. System PSPs are developed in collaboration with chip and system providers and are priced individually.

Prism is available directly from CriticalBlue, via distributors and through platform providers.



## Evaluation

Wondering what multithreading and multicore could do for your software? Have a project ready to consider for multicore? Need to learn how to go parallel?

Evaluate Prism on your favorite platform free for 30 days at:

<http://www.criticalblue.com/prism/eval>

## System Requirements

- Linux or bare metal target operating system
- Host PC with at least 2GB memory
- Windows or Linux host operating system