Abstract
With the widespread use of GPUs, it is important to ensure that programmers have a clear understanding of their shared memory consistency model i.e. what values can be read when issued concurrently with writes. GPUs present very different memory and concurrency systems from traditional CPUs and have not been the subject of any published study we know yet. We propose a collection of litmus tests that illustrate interesting visibility and ordering properties. We establish a model using intuitive data structures and implement our model in the Murphi modeling language. As a preliminary study, we restrict our model to Load (Ld), Store (St), Thread Fence (TF) and Thread Fence Block (TFB) instructions across global and shared memory.

Motivation
Lack of Clarity on Fences
Memory fences can have many different properties, including static (ordering within a thread) or dynamic (memory visibility to other threads) properties. Fences can also be cumulative which requires ensuring visibility of values the calling thread did not author. The current CUDA documentation does not mention any static or cumulative properties at all.

Performance Increase
Well-synchronized code hides weak memory model issues from the programmer; however, synchronization operations can be expensive. Cutting edge algorithms bypass these expensive operations to obtain significant increases in performance1 at the cost of exposing the memory model, which must be understood for correct implementation.

Looming Bugs
If developers write unsynchronized code assuming certain instruction orderings or memory visibilities which are not in line with the memory consistency model, then their code is buggy. We have found instances of such assumptions in unsynchronized real world code, instances in real-world linear algebra libraries, and graph traversal algorithms.

Murphi Implementation
We implemented our model in the Murphi modeling language. The model is available at: http://www.cs.utah.edu/~tylers/CUNAM.
It is easily modifiable to run custom litmus tests and verify assertions regarding the test. A flowchart of the process and possible outcomes is shown below:

Litmus Tests and Results

<table>
<thead>
<tr>
<th>Relaxed Coherence</th>
<th>Classical Coherence</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>T0</strong>: St(a,1); ...;</td>
<td><strong>T0</strong>: St(a,1); St(a,2)</td>
</tr>
<tr>
<td><strong>T1</strong>: St(a,2); ...;</td>
<td><strong>T1</strong>: Ld(a,1); Ld(a,2)</td>
</tr>
<tr>
<td><strong>T2</strong>: Ld(a,1); TFB; Ld(a,2)</td>
<td><strong>T2</strong>: Ld(a,2); Ld(a,1)</td>
</tr>
<tr>
<td><strong>T3</strong>: Ld(a,2); TFB; Ld(a,1)</td>
<td><strong>T3</strong>: ...;</td>
</tr>
</tbody>
</table>

- **Simple Global/Shared Visibility across threads**
  - **T0**: St(a,1); St(b,2); ...; 
  - **T1**: Ld(b,2); Ld(a,1); ...; 
- **T0 and T1 in the same block**
  - **T0**: St(a,1); TFB; St(b,2); ...; 
  - **T1**: Ld(b,2); TFB; Ld(a,1); ...; 
- **T0 and T1 not in the same block**
  - **T0**: St(a,1); TFB; St(b,2); ...; 
  - **T1**: Ld(b,2); TFB; Ld(a,1); ...; 
  - **T0**: St(a,1); TF; St(b,2); ...; 
  - **T1**: Ld(b,2); TF; Ld(a,1); ...; 

Write Atomicity Relaxation

- **T0**: St(a,1); ...; ...; 
- **T1**: St(b,2); ...; ...; 
- **T2**: Ld(a,1); TFB; Ld(b,0); 
- **T3**: Ld(b,2); TFB; Ld(a,0); 

Model

Conclusions/References
This model is currently being reviewed by industry experts and is expected to grow and change based on feedback. Future work includes: a more complete treatment of PTX, a level-language model for CUDA, an axiomatic model with an equivalence proof and a contrast with observable behavior on GPUs.