This paper describes a new security system for reconfigurable hardware such as an FPGA. The goal of this experiment is to create a secure implementation of multiple third party modules on the same FPGA by enforcing a memory access policy for each module. The isolation of each module will be the result of this implementation. Pending drastic complications, it is proposed that the author is hopeful to perform more research in the development of new security policies that can be created using this system. The approach described is novel, but the paper does not address issues which this reviewer feels are relevant and lacks solid results which makes acceptance of the proposed security technique difficult.

It is stated that the reason for developing a security system which performs memory-wise separation of modules on the same FPGA is used because third party modules come with different trust levels, but is there a specific application or family of applications that this technology is geared towards?

One of the questions that I had was how this security system would benefit a company who designs all of its cores in-house (doesn't the government do this for national security reasons?)? Is the interference of modules on a single FPGA a problem if the cores are not created with malicious intent?

Another question regards the security policy. When a violation occurs in the policy, what actions are taken? If the system simply reboots or shuts down, this could be viewed as an effective malicious attack on the system. If the system stays running, what happens to the memory access request? Is it simply discarded or is there a way for remapping the address to another location within the bounds of memory for that module? What would be the overhead of such an event? If a chip continuously violates is policy, how much overhead will this incur? Addressing these questions would have resulted in a higher grade for this paper.

The (seemingly) application specific design of the reference monitor brings a question regarding the man-power required to implement a reference monitor. In its current state, it seems as if the reference monitor simply checks who sent the memory request and then checks whether the memory request is within a certain range of addresses. I realize that the development of a policy is outside the scope of this project, but will it be difficult or time consuming to implement a new reference monitor (from the ground up) every time a design is changed? Can the reference monitor be modularized so that certain pieces can be reused during the creating of new reference monitors? Also, I would have liked to see a brief section on the process of validating the correct operation of the reference monitor.

Finally, by simply stating that your modifications will add just one cycle of latency to the original system is a hard sell without a more detailed description of the actions taken during a security policy violation and without talking about how often applications access
memory and how many cycles it usually takes to access a piece of memory. If you can make it clear that the overhead of one cycle is insignificant compared to the frequency of memory accesses and the number of cycles per memory request, then you make a stronger point for your technique.

There is also a typo in Section 5.2. Xiling is currently in place of what I believe should be Xilinx.

On a more positive note, the background section of this paper was written well. I felt that the existing research in the field was presented well with a brief description of each one. It was quickly apparent to this reviewer that this research is indeed novel.

The goals set for the remainder of this term are also reasonably obtainable.