

# CS615 Project Report

## Verification of Linux kernel code

Group Members:

Ajay Kumar Sarda (06305026)

Pradnya Chavarkar (06305907)

In this project we have analyze the DMA channel request code fragment of Linux kernel code using the BLAST tool. We have studied DMA code and augmented it with appropriate error location, which enables BLAST to perform predicate abstraction to decide the error reachability in the given code.

Outline of the report is as follows. We start the discussion with the description of BLAST tool and why it is appropriate tool for analyzing code in our project. The desired behavior of DMA request is presented next. The properties of the DMA request is identified and noted in section . The code for DMA request with our appended comments is presented followed by conclusion.

### **BLAST:**

BLAST tool is a model checking tool for C programs. BLAST checks whether code satisfies the behavioral requirements specified by properties of the program. BLAST implements an abstract-model check-refine loop to check for reachability of a specified label in the program. The abstract model is built on the fly using predicate abstraction. This model is then checked for reachability. If there is no (abstract) path to the specified error label, BLAST reports that the system is safe and produces a succinct proof. If there is error, BLAST checks the path by executing the program. It is driven by counter-examples.

### **DMA functions:**

Tow data structures are defined to store the properties and the channels. "dma\_chan" stores properties of a channel such as lock and the device which locked it. "dma\_cha\_busy" is a array of type "dma\_chan" where all the DMA channels are stored. This array is modified to reflect the changes made on the run. Two main functions are request\_dma and free\_dma. We have verified both these functions.

#### request\_dma:

It takes two arguments, device\_id and the channel requested. Total number of DMA channels are specified in the program.

If the channel requested is greater than the total DMA channels then return invalid.

If the channels is already busy then return busy. This is done by checking the lock on the channel if the lock is 1 then the channel is busy else it is free.

If the channel is free then it is set to 1 (busy) and the corresponding device name is also set.

#### free\_dma:

It takes one argument which is the channel to be freed. It sets the lock of that channel to 0 (free). If the channel requested is greater than the total DMA channels available then it returns invalid.

### **Properties:**

#### request\_dma:

- If the channel is busy then it should correctly return, it should not continue
- if user provides a wrong DMA channels number then the program should terminate
- Check if the program is assigning a already busy channel.

#### free\_dma:

- If the channel is freed then the lock should be set to 0 accordingly
- If the channel requested is greater than the total DMA channels available then the program should terminate.

### **Difficulties faces:**

1. Original kernel code sets the lock value of a DMA channel using a exchange function. To this function we pass the DMA channel pointer which is to be modified and the value to which the lock is set. In this function we were not able to change the value of lock through pointer variable.
2. Appropriate changes are made to the code making sure that the functionality remains the same.
3. We Had originally planned to verify the code for page allocation. But it turned out that it had quit a few outer connections. Also blast was not able to handle some of the code.